On the computation of the Möbius transform

Morgan Barbier, Hayat Cheballah, Jean-Marie Le Bars

On the computation of the Möbius transform

Morgan Barbier[a], Hayat Cheballah[a], Jean-Marie Le Bars[a]

[a] *Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France*

Abstract

The Möbius transform is a crucial transformation into the Boolean world; it allows to change the Boolean representation between the True Table and Algebraic Normal Form. In this work, we introduce a new algebraic point of view of this transformation based on the polynomial form of Boolean functions. It appears that we can perform a new notion: the Möbius computation variable by variable and new computation properties. As a consequence, we propose new algorithms which can produce a huge speed up of the Möbius computation for sub-families of Boolean function. Furthermore we compute directly the Möbius transformation of some particular Boolean functions. Finally, we show that for some of them the Hamming weight is directly related to the algebraic degree of specific factors.

Keywords: Boolean functions, Möbius transform, Butterfly algorithm

1. Introduction

Numerous studies of Boolean functions have been conducted in various fields like cryptography and error correcting codes [7], Boolean circuits and Boolean Decision Diagram (BDD) [3], Boolean logic [1] or constraint satisfaction problems [11]. There are many ways to represent a Boolean function which depends of the domain. For instance, on propositional logic one usually uses the conjunctive normal form or the disjunctive normal form, while we often use the BDD in Boolean circuits.

Email addresses: morgan.barbier@ensicaen.fr (Morgan Barbier), jean-marie.lebars@unicaen.fr (Jean-Marie Le Bars)

The various criteria of a Boolean function lead us to bring them together in numerous classes of Boolean functions which share some set of requirements and basic operations involved in studies mentioned above consists to build a Boolean function in a class or to check if a Boolean function belongs to a class.

Most of the time, practical applications involve several properties which require different representations. For instance, the (algebraic) degree and the (Hamming) weight are crucial criteria in Cryptography but these basic criteria are efficiently managed by distinct representations.

Indeed, the best representation for the degree is the Algebraic Normal Form (the characteristic function of monomials), while the weight requires the truth table (the characteristic function of minterms). Both ANF and truth table representation require a binary word of length 2^n, where n is the number of variables.

Thus the Reed-Muller decomposition (or expansion) allows us to perform recursive decomposition , enumeration and random generation among the degree whereas the Shannon decomposition (or expansion) does the same task among the weight [25] shows the switching network interpretation of this identity, but Boole will be the first to mentioned it [2].

As its name implies, Reed-Muller decomposition is applied in error correcting codes for Reed-Muller codes[17], but also various other fields, for example to implemente circuits with AND/OR gates [19]. Furthermore it is often used to construct classes of boolean functions. One example is the Maiorana-McFarlands functions where Boolean functions are obtained by expansions of affine functions (see [12, 18] for the first studies and [6] for the use of this class for cryptography).

Shannon decomposition is very often applied in cryptography, especially when we want to maintain a condition over the Hamming weight. However the name is not explicitly mentioned, less specific terms like concatenation or construction are rather used [26, 7, 8]. Furthermore it occurs in various other fields like Ordered Binary Decision Diagrams (OBDD) [21] or Modal Logics [23].

These decompositions allow us to rewrite a Boolean function with n variables into two Boolean functions with $n-1$ variables, while the expansions perform the same acts in reverse, they allows us to build a Boolean function with n variables with two Boolean functions with $n-1$ variables.

Since these decompositions appear to be orthogonal, it seems unreachable to consider them simultaneously or to perform enumeration or random generation with both criteria.

The Möbius transform allows to pass from one to the other [7, 15, 24]. The Butterfly algorithm appears as the best known algorithm which performs this transformation. It was invented by Gauss in 1805 and Cooley and Tukey independently rediscovered this algorithm for the Fast Fourier Transform (FFT) (CooleyTukey FFT algorithm [14, 10, 16]). This is a divide and conquer algorithm which may be implemented in recursive or iterative form. It has quasi-linear complexity with respect to representation length, $n\,2^{n-1}$ in term of number of XOR operations $\oplus$. However some Boolean functions have compact representation with monomials sum or conjunctive or disjunctive normal form and we may expect to get more efficient Möbius transform algorithm for these functions. On the other hand, the Möbius transform is not necessary when we want to answer the two following problems : finding the Hamming weight from the ANF and finding the algebraic degree from the truth table. The aim of our work is to characterize classes for which we have algorithms to answer these two problems more efficient than Butterfly algorithm.

The key ingredient of this work is to manipulate polynomials with Möbius transform operators instead of Boolean functions. Different works in pure Mathematics, as for example complex variable, are provide interesting new results with the polynomial approach [20, 27]. These polynomials are not Boolean functions, they contain indeterminates (defined by indices involved in monomials) instead of variables. It is possible to go from one world to another by fixing the number of variables of Boolean functions. We prove that this new approach provides better algorithms to perform Möbius transform (from ANF

to truth table) when we have few monomials in the ANF and monomials of high degree. For instance, with $2^{n/2}$ monomials of degree greater or equal to $2^{n/2}$, our method has a complexity 2^n; thus we speed up butterfly algorithm with a $\frac{n}{2}$ factor.

Section 2 provides the different representations of Boolean functions and exhibit the function to change a representation from another one. In Section 3, we discuss on the Möbius transformation and its first properties. Section 4 is dedicated to reformulate the Möbius transform for the polynomial form of the Boolean function, thus we deduce faster algorithm to compute it. Finally, Section 5 shows how to compute directly the Möbius transform and the Hamming weight of simple and more complicated families of Boolean functions, we conclude with a speed up of greater of 10% on Achterbahn-128.

2. Representations of a Boolean functions

A Boolean function is a mathematical object which is used in different domains: error correcting code, cryptography, constraint satisfaction problems, boolean circuits, etc... Most of time, each of the previous domains use a particular point of view of Boolean functions, thus it exists different representations of Boolean functions. Each point of view make easier to study specific properties of Boolean functions. In this work, we regularly switch between representations. We propose, to give a brief overview of three following representations: Algebraic Normal Form (ANF), truth table, and polynomial form of Boolean functions.

2.1. Based table representations

2.1.1. Monomials and Minterms

Let $\mathcal{F}_n$ be the set of Boolean functions with n variables $x_1, \ldots, x_n$. Monomials and minterms play a role of canonical element in the different writings.

Let us to denote $x = (x_1, \ldots, x_n)$. For any $u = (u_1, \ldots, u_n) \in \mathbb{F}_2^n$, x^u will be denoted the monomial $x_1^{u_1} \ldots x_n^{u_n}$. The minterm M_u is the Boolean function

with n variables defined by its evaluation

$$M_u(a) = \begin{cases} 1, & \text{if } u = a; \\ 0, & \text{otherwise.} \end{cases}$$

Let $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n) \in \mathbb{F}_2^n$, we will write $u \preceq v$, a partial order when $u_i \leq v_i$, for any $i \in \{1, \ldots, n\}$.

A minterm (resp. a monomial) may be written as a sum of monomials (resp. minterms).

$$\begin{cases} M_u &= \bigoplus_{u \preceq v} x^v; \\ x^u &= \bigoplus_{u \preceq v} M_v. \end{cases}$$

2.1.2. Characteristic functions of monomials and minterms

Let $f \in \mathcal{F}_n$ be a Boolean function, f may be viewed as a sum of of minterms

$$f = \bigoplus_{u \in \mathcal{F}_2^n} \theta_u M_u, \text{ with } \theta_u \in \mathbb{F}_2.$$

Its Truth Table is the characteristic function of minterms, that is:

$$T(f) = t_1 \ldots t_{2^n},$$

where $t_k = \theta_u$, with $k = \sum_{i=1}^{n} u_i \, 2^{i-1}$. Moreover, f may be also viewed as a sum of monomials

$$f = \bigoplus_{u \in \mathcal{F}_2^n} \alpha_u x^u, \text{ with } \alpha_u \in \mathbb{F}_2.$$

Its ANF (Algebraic Normal Form) is the characteristic function of monomials, that is:

$$A(f) = a_1 \ldots a_{2^n},$$

where $a_k = \alpha_u$, with $k = \sum_{i=1}^{n} u_i \, 2^{i-1}$.

Example 1. *Let $f = x_1 \oplus x_1 x_2 \in \mathcal{F}_2$ be a Boolean function with two variables. Then its truth table and its ANF are represented by four long bit sequences, and we have:*

- $T(f) = 0100$*;*
- $A(f) = 0101$.

Obviously, we may choose in both cases other orders to encode the characteristic function, we may for instance permute the order of variables.

2.2. Polynomial representation

Let $n \in \mathbb{N}$ and $i_1, \ldots, i_n \in \mathbb{N}$, we will denote by $\mathbb{F}_2[X_{i_1}, \ldots, X_{i_n}]$ the set of polynomials over the field $\mathbb{F}_2$ with the indeterminates $X_{i_1}, \ldots, X_{i_n}$.

Notation 1. *Let $n \in \mathbb{N}^*$ and $u = (u_1, \ldots, u_n) \in \mathbb{F}_2^n$. Recall that x^u is the monomial $x_{u_1} x_{u_2} \ldots x_{u_n}$. In order to distinguish a monomial over Boolean functions and a monomial over polynomials, we use the respective notations x^u and X^u.*

Definition 1 (Polynomial form)**.** *Let $f \in \mathcal{F}_n$ such that*

$$f = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u x^u.$$

We call the polynomial form of the Boolean function f, the polynomial in $\mathbb{F}_2[X_1, \ldots, X_n]$:

$$\pi_n(f) = \sum_{u \in \mathbb{F}_2^n} \alpha_u X^u.$$

Since an indeterminate $X_j \in \{X_{i_1}, \ldots, X_{i_n}\}$ could not occur in $P \in \mathbb{F}_2[X_{i_1}, \ldots, X_{i_n}]$, we have $\mathbb{F}_2[X_{i_1}, \ldots, X_{i_n}] \subset \mathbb{F}_2[X_{j_1}, \ldots, X_{j_m}]$ if $\{i_1, \ldots, i_n\} \subset \{j_1, \ldots, j_m\}$. Conversely $P \in \mathbb{F}_2[X_{i_1}, \ldots, X_{i_n}]$ means that any indeterminate X_j which occurs in P belongs to $\{X_{i_1}, \ldots, X_{i_n}\}$. Thus the polynomial $X_1 + X_1X_2$ belongs to $\mathbb{F}_2[X_1, X_2]$ but also belongs to $\mathbb{F}_2[X_1, X_2, X_3]$. We will use the term indeterminate instead of variable to notice that we manipulate formal terms X_{i_j} without notion of evaluation.

Let $P \in \mathbb{F}_2[X_{i_1}, \ldots, X_{i_n}]$, for any $i \in \mathbb{N}$, we will consider the following decomposition

$$P = X_i P_i^0 + P_i^1,$$

where the second part contains all the monomials without the indeterminate X_i and the first one contains all the other monomials. Obviously, the case $P_i^0 = 0$ means the indeterminate X_i does not occur in P.

Example 2 (Example 1 continued)**.** *Let $f = x_1 \oplus x_1x_2 \in \mathcal{F}_2$. We define $f_3 \in \mathcal{F}_3$ and $f_4 \in \mathcal{F}_4$ such that $\pi_2(f) = \pi_3(f_3) = \pi_4(f_4)$. Then*

$\pi_2(f)$	=	$X_1 + X_1X_2$			
$A(f)$	=	0101	$T(f)$	=	0100
$A(f_3)$	=	01010000	$T(f_3)$	=	01000100
$A(f_4)$	=	0101000000000000	$T(f_4)$	=	0100010001000100

Although the polynomial form seems to be identical to the ANF, we can see in Example 2, that the size of the ANF representation fixed the number of variables.

2.3. Differences and similarities between representations

2.3.1. Hamming weight and algebraic degree

Let $f \in \mathcal{F}_n$ be a Boolean function, we will write $w_H(f)$ the (Hamming) weight of f, *ie* the number of 1 of $T(f)$ and $\deg(f)$ the (algebraic) degree of f, *ie* the maximal degree of the monomials in the polynomial or ANF of f.

2.3.2. Shannon and Reed-Muller decompositions

While the Reed-Muller decomposition is related to the algebraic normal form, the Shannon one is associated to truth table. Indeed, let $f \in \mathcal{F}_n$, the *Reed-Muller decomposition*, consists in rewriting the Boolean function as

$$f = f_R^0 \oplus x_n f_R^1,$$

where $f_R^0, f_R^1 \in \mathcal{F}_{n-1}$ and are unique. Clearly, the part $x_n f_R^1$ correspond exactly at all monomials of f where x_n is, and f_R^0 the part of f where x_n is not. Let $\|$ be the concatenation over words, then

$$A(f) = A(f_R^0) \parallel A(f_R^1),$$

$A(f_S^0)$ (resp. $A(f_S^1)$) contains all the monomials x^u of the ANF of f, where $u_n = 0$ (resp. $u_n = 1$).

The *Shannon decomposition*, consists in rewriting the Boolean function as

$$f = (1 \oplus x_n) f_S^0 \oplus x_n f_S^1,$$

where $f_S^0, f_S^1 \in \mathcal{F}_{n-1}$ and are unique. Clearly the part $(1 \oplus x_n) f_S^0$ gives the part of f when $x_n = 0$, and $x_n f_S^1$ the part of f when $x_n = 1$. Thus

$$T(f) = T(f_S^0) \parallel T(f_S^1),$$

$T(f_S^0)$ (resp. $T(f_S^1)$) contains all the minterms M_u of the ANF of f, where $u_n = 0$ (resp. $u_n = 1$).

Remark 1. *Let $f \in \mathcal{F}_n$, we have by a trivial identification $f_R^0 = f_S^0$ and $f_R^1 = f_S^0 \oplus f_S^1$.*

Remark 2. *The Shannon decomposition is the natural decomposition for manipulating the minterms since $T(f) = T(f_S^0) \parallel T(f_S^1)$. This trivially implies*

$$w_H(f) = w_H(f_S^0) + w_H(f_S^1).$$

On the other hand the Reed-Muller decomposition is the natural decomposition for manipulating the monomials; since $A(f) = A(f_R^0) \parallel A(f_R^1)$, this implies

$$\deg(f) = max\left(\deg(f_R^0), \deg(f_R^1) + 1\right).$$

3. Möbius transform: operator relating the representations

Since the polynomial and the ANF representation of a Boolean function involving the presence of monomials, it is easy to see these two representations are in direct connection. Moreover, it is a lot more difficult to see that the truth table and the ANF of a Boolean function are connected by a transformation, called the *Möbius transform*. We noted it μ and is defined by the following bijection

$$\begin{array}{rcl} \mu : \mathcal{F}_n & \longleftrightarrow & \mathcal{F}_n \\ f & \longmapsto & \mu(f), \end{array}$$

such that, for any $f \in \mathcal{F}_n$ and $a \in \mathbb{F}_2^n$

$$f(a) = \bigoplus_{u \in \mathbb{F}_2^n} \mu(f)(u) a^u. \tag{1}$$

The Möbius transform allows us to compute the truth table representation from ANF one and vice versa. Let f and $g \in \mathcal{F}_n$, the following assertions are equivalent:

$$\begin{cases} \mu(f) = g; \\ \mu(g) = f; \\ A(f) = T(g); \\ T(f) = A(g). \end{cases}$$

We propose to present a known result [24, Theorem 5, page 5] in a different usual way. Thus we easy make the link between the Reed-Muller parts f_R^0 and f_R^1 with the Möbius transform.

Proposition 1. *Let $f \in \mathcal{F}_n$ and $f_R^0,\ f_R^1 \in \mathcal{F}_{n-1}$ be the Reed-Muller decomposition of f, ie $f = f_R^0 \oplus x_n f_R^1$. Then*

$$\mu(f) = (1 \oplus x_n)\mu(f_R^0) \oplus \mu(f_R^1).$$

Proof. Let $a = (a_1, \ldots, a_n)$ and $u = (u_1, \ldots, u_n) \in \mathbb{F}_2^n$, $b = (a_1, \ldots, a_{n-1})$ and $v = (u_1, \ldots, v_{n-1})$. we will write $a = ba_n$ and $u = vb_n$. It follows $a^u = b^v\ a_n^{u_n}$. Since $a_n^{u_n} = 0$ if and only if $a_n = 0$ and $u_n = 1$, we have $a^u = 0$ if $a_n = 0$ and $u_n = 1$ and $a^u = b^v$ otherwise.

The relation (1) implies

$$\begin{array}{rcl} f(b0) & = & \bigoplus_{v\in\mathbb{F}_2^{n-1}} \mu(f)(v0)b^v 0^0 \bigoplus_{v\in\mathbb{F}_2^{n-1}} \mu(f)(v0)b^v 0^1, \\ & = & \bigoplus_{v\in\mathbb{F}_2^{n-1}} \mu(f)(v0)b^v; \\ f(b1) & = & \bigoplus_{v\in\mathbb{F}_2^{n-1}} \mu(f)(v1)b^v 1^0 \bigoplus_{v\in\mathbb{F}_2^{n-1}} \mu(f)(v1)b^v 1^1, \\ & = & \bigoplus_{v\in\mathbb{F}_2^{n-1}} (\mu(f)(v1) \oplus \mu(f)(v1))b^v. \end{array}$$

We deduce

$$\begin{array}{rcl} \mu(f)(v0) & = & \mu(f_R^0)(v) \\ \mu(f)(v1) & = & \mu(f_R^0)(v) \oplus \mu(f_R^1)(v) \end{array}$$

Thus $\mu(f) = (1 \oplus x_n)\mu(f_R^0) \oplus \mu(f_R^1)$. □

In the following, we propose a new operator, which is related to the Möbius transform, which is dedicated to manipulate indeterminate one by one.

Definition 2. *Let $f \in \mathcal{F}_n$ and $P = \pi_n(f) \in \mathbb{F}_2[X_1, \ldots, X_n]$ its polynomial form. Assume that $P = P_i^0 + X_i P_i^1$. We define the operator μ_{X_i} by*

$$\mu_{X_i}(P) = P_i^0 + X_i(P_i^0 + P_i^1).$$

In particular, if $i \notin \{i_1, \ldots, i_n\}$, $\mu_{X_i}(P) = (1 + X_i)P$ and if $P = X_i P_i^1$, $\mu_{X_i}(P) = P$.

Proposition 2. *The operators μ_{X_i} are commutative, that is*

$$\mu_{X_i}(\mu_{X_j}(P)) = \mu_{X_j}(\mu_{X_i}(P)).$$

Proof. Let P_1, P_2, P_3 and P_4 the four polynomials without the variables X_i and X_iX_j such that

$$P = P_1 + X_i P_2 + X_j P_3 + X_i X_j P_4.$$

Thus

$$\begin{array}{rcl} \mu_{X_i}(P) & = & P_1 + X_i(P_1 + P_2) + X_j(P_3 + X_i P_4 + X_i P_3); \\ \mu_{X_j}(\mu_{X_i}(P)) & = & P_1 + X_i(P_1 + P_2) + X_j(P_1 + P_3 + X_i(P_1 + P_2 + P_3 + P_4)), \\ & = & P_1 + X_j(P_1 + P_3) + X_i(P_1 + P_2 + X_j(P_1 + P_2 + P_3 + P_4)), \\ & = & \mu_{X_i}(\mu_{X_j}(P)). \end{array}$$

□

Notation 2. *Let $k \in \mathbb{N}^*$ and $i_1, \ldots, i_k \in \mathbb{N}$. Let P be a polynomial over $\mathbb{F}_2$. We denote the operator $\mu_{X_{i_1} \ldots X_{i_k}}$ by*

$$\mu_{X_{i_1} \ldots X_{i_k}}(P) = \mu_{X_{i_1}}(\mu_{X_{i_2}}(\ldots \mu_{X_{i_k}}(P) \ldots).$$

We may extend the previous Proposition for any permutation σ of $\{1, \ldots, k\}$,

$$\mu_{X_{i_1} \ldots X_{i_k}}(P) = \mu_{X_{i_{\sigma(1)}} \ldots X_{i_{\sigma(k)}}}(P).$$

Hence $\mu_{X_{i_1} \ldots X_{i_k}}(P)$ depends only of the set of indexes $N = \{i_1, \ldots, i_k\}$.

Notation 3. *We will write $\mu_N(P)$ instead of $\mu_{X_{i_1} \ldots X_{i_k}}(P)$. Moreover, let $n \in \mathbb{N}^*$, we will denote by $[n]$ the set $\{1, \ldots, n\}$.*

Example 3 (Example 2 continued)**.** *Let $f \in \mathcal{F}_2$ such that its polynomial form is $X_1 + X_1X_2$. Then*

$$\begin{aligned}
\mu_{[2]}(X_1 + X_1X_2) &= \mu_2(\mu_1(X_1 + X_1X_2)) \\
&= \mu_2(X_1 + X_1X_2) \\
&= X_1X_2 + (1 + X_2)X1 \\
&= X_1 \\
\mu_{[3]}(X_1 + X_1X_2) &= \mu_{\{3\}}(\mu_{[2]}(X_1 + X_1X_2)) \\
&= \mu_{\{3\}}(X_1) \\
&= (1 + X_3)X_1 \\
&= X_1 + X_1X_3 \\
\mu_{[4]}(X_1 + X_1X_2) &= (1 + X_4)(X_1 + X_1X_3) \\
&= X_1 + X_1X_3 + X_1X_4 + X_1X_3X_4
\end{aligned}$$

The following proposition explains how the previous operator is related to the Möbius transform.

Proposition 3. *Let $n \in \mathbb{N}^*, f, g \in \mathcal{F}_n$ with polynomial forms $P = \pi_n(f)$ and $Q = \pi_n(g)$. The following assertions are equivalent:*

(a) $\mu(f) = g$;

(b) $\mu_{[n]}(P) = Q$.

Which yields the following commutative diagram:

$$\begin{array}{ccccc}
 & f & \xrightarrow{\mu} & g & \\
\pi_n & \downarrow & & \downarrow & \pi_n \\
 & P & \xrightarrow{\mu_{[n]}} & Q &
\end{array}$$

Proof. We only proof that $(a) \implies b$. The other implication is similar.

We use a induction on n. For $n = 1$, we have by disjunction

f	P	$\mu(f)$	$\mu_{X_1}(P)$
0	0	0	0
1	1	$1 \oplus x_1$	$1 + X_1$
x_1	X_1	x_1	X_1
$1 \oplus x_1$	$1 + X_1$	1	1

Since for all $f \in \mathcal{F}_1$, we have $\mu(f) = \mu_{X_1}(P)$, the induction holds for $n = 1$.

Assume now this is true for $n > 1$:

$$\pi_n(\mu(f)) = \mu_{[n]}(\pi_n(f)).$$

Let $f \in \mathcal{F}_{n+1}$ be a Boolean function and $f_R^0, f_R^1 \in \mathcal{F}_n$ such that $f = f_R^0 \oplus x_{n+1} f_R^1$. Thus with the induction assumption and Proposition 1:

$$\begin{aligned}
\pi_{n+1}(\mu(f)) &= (1 + X_{n+1}) \times \pi_n\left(\mu\left(f_R^0\right)\right) + \pi_n\left(\mu\left(f_R^1\right)\right), \\
&= (1 + X_{n+1}) \times \mu_{[n]}\left(\pi_n\left(f_R^0\right)\right) + \mu_{[n]}\left(\pi_n\left(f_R^1\right)\right), \\
&= \mu_{X_{n+1}}\left(\mu_{[n]}\left(\pi_n\left(f_R^0\right) + X_{n+1}\pi_n\left(f_R^1\right)\right)\right), \\
&= \mu_{[n+1]}\left(\pi_{n+1}\left(f_R^0 + x_{n+1} f_R^1\right)\right), \\
&= \mu_{[n+1]}\left(\pi_{n+1}(f)\right).
\end{aligned}$$

□

Directly, the operator on monomials inherits of Möbius transform properties.

Proposition 4. *Let X_i be an indeterminate. The automorphism μ_{X_i} is involutive:*

$$\mu_{X_i}^2 = id.$$

Proof. Let $P = P_i^0 + X_i P_i^1$ the Reed-Muller decomposition of polynomial P, we denote $Q = \mu_{X_i}(P)$. By definition of μ_{X_i}, $Q = P_i^0 + X_i(P_i^0 + P_i^1)$, thus $\mu_{X_i}(Q) = P^0 + X_i(P_i^0 + P_i^0 + P^1) = P$. □

Propositions 3 and 4 imply the Corollary below

Corollary 1. *Let $N \subset \mathbb{N}$ be a subset, then μ_N satisfies*

$$\mu_N^2 = id.$$

Let $f \in \mathcal{F}_n$ be a Boolean function and $P = \pi_n(f)$ its polynomial form; Corollary 1 provides an alternative proof that μ is an involutive automorphism, since combined with Proposition 3 it implies $\mu_{[n]}^2 = id$.

Notation 4. *Let $I \subset [n]$, we define $M^I = \prod_{i \in [n] \setminus I} \left(1 + X^i\right)$ which is the polynomial form of the minterm M_u, where $I = I_u$.*

Proposition 5. *Let $I \subset [n]$, then*

$$\mu_{[n]}(X^I) = X^I \times \prod_{i \in [n] \setminus I} \left(1 + X^i\right) = M_I.$$

Moreover since $\mu_{[n]}$ is an involutive function $\mu_{[n]}(M_I) = X^I$.

Proof. Thanks to Definition 2, we obtain by recurrence

$$\begin{aligned} \mu_{[n]}(X^I) &= X^I \times \mu_{\{X_i \,:\, i \in [n] \setminus I\}}(X^I) \\ &= X^I \times \prod_{i \in [n] \setminus I} \left(1 + X^i\right). \end{aligned}$$

And finally Proposition 4 holds the last statement. □

This provide an alternative proof $\mu(x^u) = M^u$ and $\mu(M^u) = x^u$.

4. A new method to compute the Möbius transform

We have introduced the Möbius transform over polynomials and show that it is possible to perform the computations in several steps with various orders thanks to the partial operators μ_{X_i}. We propose to firstly reformulate the Möbius transform over polynomials in order to introduce two new algorithms based on this reformulation.

4.1. Reformulation of Möbius transform

To introduce our reformulation let us to present a new operator given in the following definition.

Definition 3 (Exclusive multiplication)**.** *Let P be a polynomial over $\mathbb{F}_2$ and $i \in \mathbb{N}$, P_i^0 and P_i^1 such that $P = P_i^0 + X_i P_i^1$. We define the* exclusive multiplication*, noted $\otimes$, as*

$$P \otimes X_i = X_i P_i^0.$$

Let I be a finite subset of $\mathbb{N}$, we generalize the definition for a monomial X^I.

$$P \otimes X^I = X^I P_{\not| I},$$

where $P_{\not| I}$ is formed with the monomials of P which contain no variables X_i, with $i \in I$.

We may now generalize for any polynomial Q. Let $\mathcal{I}$ be a set of finite subsets of $\mathbb{N}$ and $Q = \sum_{I \in \mathcal{I}} X^I$,

$$P \otimes Q = \sum_{I \in \mathcal{I}} P \otimes X^I.$$

Proposition 6. *Let $I = \{i_1, \ldots, i_k\}$ be a finite subset of $\mathbb{N}$.*

$$P \otimes X^I = (\ldots (P \otimes X_{i_1}) \otimes X_{i_2}) \otimes \ldots X_{i_k}).$$

Thanks to the previous definition, we can reformulate the Möbius transform of the Boolean function as a multiplication; this is the result of the following proposition.

Proposition 7. *Let P be a polynomial over $\mathbb{F}_2^n$ and $i \in \mathbb{N}$.*

$$P \otimes (1 + X_i) = \mu_{X_i}(P).$$

Proof. Let P_i^0 and P_i^1 such that $P = P_i^0 + X_i P_i^1$.

$$\begin{array}{rcl} P \otimes (1 + X_i) & = & P + P \otimes X_i \\ & = & P + X_i P_0^i \\ & = & P_0^i + X_i(P_0^i + P_1^i) \\ & = & \mu_{X_i}(P). \end{array}$$

□

Thanks to the previous results, we obtain the following corollary, which supplies a new reformulation of the Möbius transform.

Corollary 2. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$. Then*

$$\mu_{[n]}(P) = P \otimes \prod_{i=1}^{n} (1 + X_i).$$

Proposition 8. *Let P be a polynomial over $\mathbb{F}_2$ and i and $j \in \mathbb{N}$.*

$$(P \otimes (1 + X_i)) \otimes (1 + X_j) = P \otimes ((1 + X_i)(1 + X_j)).$$

Proof. Let $P = P_1 + X_i P_2 + X_j P_3 + X_i X_j P_4$. By Proposition 7,

$$\begin{array}{rcl} P \otimes (1 + X_i) & = & P_1 + X_j P_3 + X_i(P_1 + X_j P_3 + P_2 + X_j P_4) \\ & = & (P_1 + X_i P_1 + X_i P_2) + X_j(P_3 + X_i P_3 + X_i P_4) \\ (P \otimes (1 + X_i)) \otimes (1 + X_j) & = & P_1 + X_i P_1 + X_i P_2 + X_j(P_1 + X_i P_1 + X_i P_2 + P_3 + X_i P_3 + X_i P_4) \\ & = & P_1 + X_i(P_1 + P_2) + X_j(P_1 + P_3) + X_i X_j(P_1 + P_2 + P_3 + P_4) \\ P \otimes ((1 + X_i)(1 + X_j)) & = & P \otimes (1 + X_i + X_j + X_i X_j) \\ & = & P_1 + P_2 + X_j P_3 + X_i X_j P_4 + X_i P_1 \\ & & + X_i X_j P_3 + X_j P_1 + X_i X_j P_2 + X_i X_j P_1 \\ & = & (P \otimes (1 + X_i)) \otimes (1 + X_j) \end{array}$$

Corollary 3. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$. Then*

$$\mu_{[n]}(P) = P \otimes (1 + X_1) \otimes (1 + X_2) \ldots \otimes (1 + X_n).$$

Now, we propose to build an algebraic structure such that the exclusive multiplication becomes the canonical multiplication in this new structure. Thus we have to create, an algebraic structure such that all monomials containing square indeterminates are projected on zero. We naturally researched a ring which is quotiented by an ideal which represent all these monomials. Thus we obtain the following proposition.

Proposition 9. *Let $\mathcal{I}_n$ be the ideal of $\mathbb{F}_2[X_1, \ldots, X_n]$ spanned by all the indeterminates with a power of two, that is*

$$\mathcal{I}_n = \langle X_1^2, \cdots, X_n^2 \rangle.$$

Then the exclusive multiplication is the natural multiplication in the ring

$$\mathcal{R}_n = \mathbb{F}_2[X_1, \ldots, X_n]/\mathcal{I}_n.$$

Proof. We propose to prove by inclusion that the ideal $\mathcal{I}_n$ is exactly all monomial with at least a square indeterminates.

Since $\mathcal{I}_n$ is an ideal, thus by the stability property, we have:

$$\forall a \in \mathcal{I}_n,\ \forall P \in \mathbb{F}_2[X_1, \ldots, X_n],\ a.P \in \mathcal{I}_n;$$

thus all monomials containing a square indeterminate is into the ideal $\mathcal{I}_n$.

Let $a \in \mathcal{I}_n$ be an element such that it does not contain any square indeterminate. Since $\mathcal{I}_n$ is spanned by $X_1^2, \cdots X_n^2$, then it exists $a_1, \cdots, a_n \in \mathbb{F}_2[X_1, \cdots, X_n]$ such that:

$$a = \sum_{i=1}^{n} a_i.X_i^2.$$

Since a does not contain any square indeterminate then $\forall i \in \{1, \cdots, n\},\ a_i = 0$; thus $a = 0$. We obtain the statement. □

Corollary 4. *Let $f \in \mathcal{F}_n$ be a Boolean function, then the computation of its Möbius transform is only a multiplication on $\mathcal{R}_n$.*

Thus we reformulate the Möbius transform such that it is equivalent to canonical multiplication into the quotient ring $\mathcal{R}_n$.

Example 4. *[Example 3 continued] With this reformulation, let us to compute again the Möbius computation of the two variables Boolean function defined by its polynomial form $P = X_1 + X_1X_2$.*

$$\begin{aligned}
\mu_{[2]}(P) &= (X_1 + X_1X_2) \otimes (1 + X_1) \otimes (1 + X_2) \\
&= (X_1 + X_1X_2) \otimes (1 + X_2) \\
&= X_1 + X_1X_2 + X_1X_2 \\
&= X_1.
\end{aligned}$$

We find exactly the same result that in Example 3.

Proposition 10. *The exclusive multiplication is commutative.*

Proof. The exclusive multiplication is only the canonical multiplication in $\mathcal{R}_n$, moreover $\mathbb{F}_4[X_1, \ldots, X_n]$ is a commutative ring, then $\mathcal{R}_n$, that is the exclusive multiplication, also is. □

4.2. Algorithms to compute the Möbius transform

We propose in this Section an algorithm which compute the Möbius transform with the multiplication $\otimes$. Firstly we will see that it is exactly the same than the iterative version of Butterfly algorithm when the algorithm is applied on a 2^n long bit vector which encodes which monomials occur in P (which corresponds to the ANF of $\pi_n^{-1}(P)$). Thus the complexity is $n2^{n-1}$. Secondly, we consider P as a list of monomials. In this case, we show that this algorithm is better than Butterfly algorithm over large classes of Boolean functions.

In the first hand, we propose to revisit the Butterfly algorithm and recall a previous improvement. And the other hand, we propose new algorithms from our previous results.

4.2.1. Butterfly algorithm

There exists a simple divide-and-conquer butterfly algorithm to perform the Möbius transform, called the Fast Möbius Transform. We work over A, a vector of size 2^n which encodes the ANF of a Boolean function f. Algorithm 1 gives the recursive version of the Fast Möbius Transform.

Algorithm 1: Recursive butterfly algorithm RBM(A,n)

Input: A be the ANF (or truth table) of a Boolean function with n variables.
Output: the truth table (or ANF) corresponding to A.

if $n = 1$ **then**
  if $A = 00$ *or* $A = 01$ **then**
    return A
  if $A = 10$ **then**
    return 11
  if $A = 11$ **then**
    return 10
else
  $A^0 \leftarrow RBM(A[0] \ldots A[2^{n-1} - 1], n - 1)$
  $A^1 \leftarrow RBM(A[2^{n-1}] \ldots A[2^n - 1], n - 1)$
for $i = 0$ **to** $2^{n-1} - 1$ **do**
  $A^1[i] \leftarrow A^1[i] \oplus A^0[i]$
return $A^0||A^1$

We may directly apply the modifications over $A = A(f)$ without recursive calls. For $i = 1$ to n, we split the string A in 2^{n-i} pairs of strings (A_1, A_2) of size 2^{i-1} and we replace A_2 by $A_1 \oplus A_2$, where $\oplus$ is here the bit-wise modulo 2 sum. Thus it provides a butterfly algorithm working with the memory in place; that is no need extra memory and copy results. It result the Algorithm 2 which gives this iterative version of the Fast Möbius Transform. It is quite the same algorithm introduced in [9], replacing plus operation by XOR.

Algorithm 2: Iterative butterfly algorithm *IBM*(A,n)

Input: A be the ANF (or truth table) of a Boolean function with n variables.

Output: the truth table (or ANF) corresponding to A.

for $i = 1$ **to** n **do**
  for $k = 0$ **to** $2^{n-i} - 1$ **do**
    for $l = 0$ **to** $2^{i-1} - 1$ **do**
      $A[k * 2^i + l + 2^i] \leftarrow A[k * 2^i + l + 2^i] \oplus A[k * 2^i + l]$

return A

4.2.2. Optimisation by isolated monomials

In 2012, Calik Cagdas and Doganaksoy Ali, compute the Hamming weight of Boolean functions from the ANF [5]. More exactly, a deep reading of this work shows that they compute the Hamming weight of Boolean functions from its polynomial form. Moreover, it provides a new algorithm which can be faster than the butterfly one over a subclass of Boolean function. The previous subclass is mainly defined by they called isolated monomials. That is they rewrite the polynomial form in isolating a monomial, and they take advantage to compute the Hamming weight, their method can be fully detailed in [5, Algo. 4.1]. An implementation is even available in [4].

4.2.3. Algorithm with the exclusive multiplication

From Corollary 2, we obtain directly the following algorithm to compute the Möbius transform.

Algorithm 3: Möbius transformation by the exclusive multiplication.

Input: P be a polynomial form of a Boolean function.

Output: Q be the polynomial such that $\mu_{[n]}(P) = Q$.

$P_0 \leftarrow P$
for $i = 1$ **to** n **do**
  $P_i \leftarrow P_{i-1} \otimes (1 + X_i)$;
return P_n

We change the point of view of the Algorithm 3, in order to make the relation with the Butterfly algorithm. We encode P by a array $A = A(\pi_n^{-1}(P)))$ of length 2^n such that for each $j = u_1 + u_2 2 \oplus \ldots + u_n 2^{n-1} \in \{0, \ldots, 2^n - 1\}$

$$A[j] = 1 \iff \text{the monomial } X^{I_u} \text{ occurs in the ANF of } f.$$

At the step i, $(P = P \otimes (1 + X_i))$, we consider all the $j = a_1 + a_2 2 \oplus \ldots + a_n 2^{n-1}$ such that $a_i = 0$ and we modify the value of $A[j + 2^i]$ when $A[j] = 1$.

Algorithm 4 gives the iterative version of Algorithm 3 over the vector A which encodes the monomials.

Algorithm 4: Reformulation of Algorithm 3.

Input: A be the ANF (or truth table) of a Boolean function with n variables.

Output: the truth table (or ANF) corresponding to A.

```
A ← A(π_n^{-1}(P))
for i = 1 to n do
    for every j = a_1 + a_2 2 ⊕ ... + a_n 2^{n-1}, where a_i = 0 do
        A[j + 2^i] ← A[j + 2^i] ⊕ A[j]
return A
```

We obtain exactly the same that algorithm 2. Indeed, let $i \in \{1, \ldots, n\}$ and $j = a_1 + a_2 2 \oplus \ldots + a_n 2^{n-1}$, where $a_i = 0$. Let $l \in \{0, \ldots, 2^i - 1\}$ and $k \in \{0, \ldots, 2^{n-i-1} - 1\}$ such that

$$\begin{cases} l &= a_1 + a_2 2 + \ldots a_{i-1} 2^{i-2} \\ k &= a_{i+1} + a_{i+2} 2 + \ldots + a_n 2^{n-i-1} \end{cases}$$

It follows $j = l + 2^i k$ and the instruction $A[j + 2^i] \leftarrow 1 - A[j + 2^i]$ is equivalent to $A[k * 2^i + l + 2^i] \leftarrow 1 - A[k * 2^i + l + 2^i]$.

4.2.4. Algorithm for list representation

In this section, we manipulate Boolean function by its polynomial form given by the list of involved monomials. Hence we can avoid useless computation, as for example a XOR bit with zero. However, this representation suffers an extra memory cost compare to the vector representation.

Proposition 11. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ be a polynomial form of the Boolean function with n variables. We denote by P_i, $i \in \{1, \ldots, n\}$ the polynomial involved in Algorithm 3 and $N(P_i)$ their number of monomials. Then Algorithm 3 uses $\sum_{i=1}^{n} N(P_i)$ XORs.*

Proof. This is a direct implication of the equality $P_{i-1} \otimes (1 + X_i) = P_{i-1} + P_{i-1} \otimes X_i$ (see Proposition 7). □

With Proposition 11, we note that the number of monomials in the list representation is essential for the complexity.

Corollary 5. *Let $N = \max\{N(P_i) \mid i \in \{1, \ldots, n\}\}$. Algorithm 3 uses at most n N XORs.*

Notation 5. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ be a polynomial form of the Boolean function with n variables. We denote $\bar{P}$ the polynomial form of the complementary Boolean function associated at polynomial P, that is*

$$P + \bar{P} = \prod_{i=1}^{n}(1 + X_i).$$

Then we propose the following result in order to improve the complexity of our algorithm.

Proposition 12. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ be a polynomial form of the Boolean function with n variables. Then*

$$\begin{array}{lcl} \mu_{[n]}(\bar{P}) & = & \mu_{[n]}(P) + 1; \\ \mu_{[n]}(P+1) & = & \mu_{[n]}(P) + \prod_{i=1}^{n}(1+X_i) = \overline{\mu_{[n]}(P)}. \end{array}$$

Proof. Let us to compute

$$\begin{array}{rcl} \mu_{[n]}(\bar{P}) & = & \left(P + \prod_{i=1}^{n}(1+X_i)\right) \otimes \prod_{i=1}^{n}(1+X_i) \\ & = & \left(P \otimes \prod_{i=1}^{n}(1+X_i)\right) + \left(\prod_{i=1}^{n}(1+X_i) \otimes \prod_{i=1}^{n}(1+X_i)\right) \\ & = & \mu_{[n]}(P) + 1 \\ \mu_{[n]}(P+1) & = & (P+1) \otimes \prod_{i=1}^{n}(1+X_i) \\ & = & \left(P \otimes \prod_{i=1}^{n}(1+X_i)\right) + \prod_{i=1}^{n}(1+X_i) \\ & = & \mu_{[n]}(P) + \prod_{i=1}^{n}(1+X_i). \end{array}$$

□

Then if the list representation of the Boolean function is dense, we can take advantage and work on the complementary, which have a sparse representation. Thus mixing with previous results, we improve the complexity for the list representation.

Corollary 6. *Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$. We may perform Algorithm 3 with $\min\left(\sum_{i=1}^{n} N(P_i), \sum_{i=1}^{n} N(\bar{P}_i)\right)$ XORs.*

Proposition 12 is useful in our context, however this result is not dedicated to our reformulation, it is also true with truth table and ANF.

We remark that the order of the multiplication by the affine polynomial plays an important role since we involved different polynomials P_i when we change the order. To illustrate our claim, we propose to make again Example 4 by multiplying with another order.

Example 5 (Example 4 continued)**.** *Let $f \in \mathcal{F}_3$ be the Boolean function in Example 4, with the list representation we can see that we need only to add 3 monomials, that is*

$$P = [X_3, X_1X_2, X_1X_3].$$

After the multiplication by affine polynomials, we obtain

$$\begin{aligned} P \otimes (1 + X_1) &= [X_3, X_1X_2]; \\ P \otimes (1 + X_1) \otimes (1 + X_2) &= [X_3, X_1X_2, X_2X_3]; \\ P \otimes (1 + X_1) \otimes (1 + X_2) \otimes (1 + X_3) &= \mu_{[3]}(P) = [X_3, X_1X_2, X_2X_3, X_1X_2X_3]. \end{aligned}$$

If we process the exclusive multiplication in the different order the number of operation in the list, that is add or remove, will considerably increase:

$$\begin{aligned} P \otimes (1 + X_2) &= [X_3, X_1X_2, X_1X_3, X_2X_3, X_1X_2X_3] \\ P \otimes (1 + X_2) \otimes (1 + X_1) &= [X_3, X_1X_2, X_2X_3] \\ P \otimes (1 + X_2) \otimes (1 + X_1) \otimes (1 + X_3) &= \mu_{[3]}(P) = [X_3, X_1X_2, X_2X_3, X_1X_2X_3]. \end{aligned}$$

We obtain the same result with 5 modifications, while Example 4 obtain the same result with only 3.

We show that in Example 5 the order of the affine polynomials is really important on the number of list modifications. We propose a strategy to minimize the number of modifications: we propose to multiply by $(1 + X_{i_0})$, where i_0 is the indeterminate which occurs the most of time in the intern representation. Hence we maximize the number of monomials for which one, we do not perform modification. In this way, we propose Algorithm 5 which manage a good order to perform successive exclusive multiplications to obtain the Möbius transform.

Where:

- *occurrence* computes a table of size n where the i-th component-wise gives the number of occurrences of X_i;
- $remove(L, M)$ and $add(L, M)$ modify the list L with the monomial M;
- $update(O, M, value)$ modifies the occurrence table O for all variables into the monomial M adding $value$.

In Table 1, we compare our proposed algorithms with the literature. We see that the list representation is only valuable for really sparse Boolean functions, or thanks to the complementary property, Proposition 12, and really dense ones.

Algorithm 5: Reformulated Möbius transformation for the list representation

Input: L be the list representation of $f \in \mathcal{F}_n$.
Output: Mu be the list representation of the Möbius transform of f.

```
Mu ← L;
O ← occurrence(L);
for i = 1 to n do
    i_0 ← argmax(O);
    Mui ← Mu;
    for M ∈ Mu do
        if not (X_{i_0} ∈ M) then
            if X_{i_0} ∈ Mu then
                remove(Mui, X_{i_0}M);
                update(O, X_{i_0}M, −1);
            else
                add(Mui, X_{i_0}, M);
                update(O, X_{i_0}M, 1);
    Mu ← Mui;
    O[i_0] ← −∞;
return Mu
```

Table 1: Number of XORs or list modifications needed to compute the Möbius transform in worst case or for the special case $f = x_3 \otimes x_1x_2 \otimes x_1x_3 \in \mathcal{F}_3$.

	Butterfly	Algorithm in [5]	Algorithm 5
Complexity	$n2^{n-1}$		$\min\left(\sum_{i=1}^{n} N(P_i), \sum_{i=1}^{n} N(\overline{P_i})\right)$
$x_3 \oplus x_1x_2 \oplus x_1x_3$	12	10	3

5. Direct Möbius computations for some Boolean functions

We have proposed a reformulation of the Möbius transform which produces two new algorithms: one for the vector representation and the other for the polynomial form. The worst case of these algorithm happen when a variable x_i does not appear. Hence this section is dedicated to directly compute the Möbius transform and the Hamming weight of a Boolean function for the worst cases of proposed algorithms.

Please note that for the following propositions, we give the Möbius transform for some families of Boolean functions. Thus, the computation cost of these Boolean functions is only their Hamming weight for simply write the result into the memory.

Proposition 13. *Let $I \subset [n]$; then*

$$\mu_{[n]}(X^I) = X^I \prod_{j \in [n] \setminus I} (1 \oplus X_j),$$

and we have $w_H(\pi_n^{-1}(X^I)) = 2^{n-|I|}$.

Proof. It is sufficient to combine Proposition 5 and Proposition 3. This result could be also proved with the relation $M_u = \bigoplus_{u \preceq v} x^v$. □

We consider the following basic algorithm to compute $\mu_{[n]}(P)$ which involves the monomial X^I. We began with the word $w = (0, \ldots, 0)$ of length 2^n; then for each monomial X^I, we flip the corresponding bits in w, hence the complexity depends on $|\bar{I}|$. For instance, if $P = X^I$, we obtain a complexity $2^{n-|I|}$.

For all $i \in [n]$, we find again that the Boolean functions of $\mathcal{F}_n$ given by the polynomial form X_i are balanced functions.

Definition 4 (Valuation)**.** *Let P be a polynomial defined over a ring $\mathcal{R}$. The valuation of P is the smallest degree of the set of its monomials.*

Example 6. *Let $P(X_1, X_2, X_3) = X_1 + X_2X_3$ and $Q(X_1, X_2, X_3) = 1 + X_2X_3 + X_1X_2X_3$ be polynomials over $\mathbb{F}_2[X_1, X_2, X_3]$, then*

$$val(P) = 1,\ val(Q) = 0.$$

Moreover, in order to the valuation has order property, it is frequently assumed that $val(0) = -\infty$.

Proposition 14. *Let $P = \sum_{I \in \mathcal{I}} X^I \in \mathbb{F}_2[X_1, \ldots, X_n]$ and $M = |\mathcal{I}|$. Then the Möbius transform of P and the Hamming weight of $\pi_n^{-1}(P)$ can be computed with a complexity $\sum_{I_i n \mathcal{I}} 2^{n-|I|}$, with upper bound $M \ . \ 2^{n-val(P)}$.*

Proof. Let $P = \pi_n(f)$ and $\mathcal{I}$ such that $P = \sum_{I \in \mathcal{I}} X^I$.

$$\mu_{[n]}(P) = \bigoplus_{I \in \mathcal{I}} \big(X^I \prod_{j \in [n] \setminus I} (1 + X_j)\big).$$

We conclude by observing that each factor of the sum contains $2^{n-|I|} \leq 2^{n-val(P)}$ terms. □

For example, if $val(P) = n/2$ and $M = 2^{n/2}$, we obtain an upper bound of the complexity $2^{n/2} \cdot 2^{n/2} = 2^n$ which is better than the complexity of butterfly algorithm which is $n2^{n-1}$.

Proposition 15. *Let P be the polynomial form of a Boolean function $f \in \mathcal{F}_{n-1}$. Then Möbius transform of the polynomial $X_n + P$ with n indeterminates is*

$$\mu_{[n]}(X_n + P) = \mu_{[n-1]}(P) + X_n \mu_{[n-1]}(P + 1).$$

Moreover the Boolean function $f' = \pi_n^{-1}(X_n + P)$ is a balanced one, that is:

$$w_H(f') = 2^{n-1}.$$

Proof. Let us to develop the computation thanks to Definition 2:

$$\begin{aligned}\mu_{[n]}(X_n + P) &= \mu_{[n]}(X_n) + \mu_{[n]}(P) \\ &= X_n\mu_{[n-1]}(1) + (1 + X_n)\mu_{[n-1]}(P) \\ &= \mu_{[n-1]}(P) + X_n\mu_{[n-1]}(P + 1).\end{aligned}$$

Moreover, applying Proposition 12:

$$\mu_{[n-1]}(P + 1) = \mu_{[n-1]}(P) + \prod_{i=1}^{n-1}(1 + X_i) = \overline{\mu_{[n-1]}(P)};$$

thus

$$\begin{aligned}w_H(f') &= w_H(f) + 2^{n-1} - w_H(f); \\ &= 2^{n-1}.\end{aligned}$$

□

Proposition 16. *The Möbius transform of the sum of all monomials of degree one is the sum of all monomials of odd degree; that is*

$$\mu_{[n]}\left(\sum_{i\in[n]} X_i\right) = \sum_{J\subset[n],\ st\ |J|\ is\ odd} X^J.$$

Thus $w_H\left(\pi_n^{-1}\left(\sum_{i\in[n]} X_i\right)\right) = 2^{n-1}$.

Proof.

$$\begin{aligned}\mu_{[n]}\left(\sum_{i\in[n]} X_i\right) &= \sum_{i\in[n]} \mu_{[n]}(X_i) \\ &= \sum_{i\in[n]} X_i \prod_{j\in[n]\setminus\{i\}} (1 + X_j) \\ &= \sum_{J\subset[n],|J| \text{ is odd}} X^J.\end{aligned}$$

□

Remark 3. *Let* $f = \bigoplus_{i=1}^{n} x^i \in \mathcal{F}_n$ *be the Boolean function which is the sum of all monomials of degree 1. Since* $w_H(f) = N(\mu_{[n]}(\sum_{i\in[n]} X_i))$*, Proposition 16 provides an alternative proof that* f *is a balanced Boolean function.*

The following Proposition shows that we may improve the complexity by a factorization.

Proposition 17. *Let $I \subset [n], J \subset [n]$ be two subsets such that $I \cap J = \emptyset$ and $n_1 = |I|$. Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ be a polynomial such that $P = X^I \left(\sum_{j \in J} X_j\right)$. Then*

$$\mu_{[n]}(P) = \left(\sum_{I \subset L \subset [n] \setminus J} X^L\right) \left(\sum_{K \subset J, |K| odd} X^K\right);$$

and $w_H\left(\pi_n^{-1}(P)\right) = 2^{n-n_1-1}$.

Proof. Let $n_2 = |J|$, it follows

$$\begin{aligned}
\mu_{[n]}(P) &= X^I \mu_{[n]\setminus I}\left(\sum_{j \in J} X_j\right), \\
&= X^I \prod_{k \in [n]\setminus(I \cup J)} (1 + X_k)\ \mu_J\left(\sum_{j \in J} X_j\right) \\
&= \left(\sum_{I \subset L \subset [n] \setminus J} X^L\right) \left(\sum_{K \subset J, |K| odd} X^K\right).
\end{aligned}$$

Since $\prod_{k \in [n]\setminus(I \cup J)}(1 + X_k)$ gives $2^{n-n_1-n_2}$ terms and 2^{n_2-1} subsets of J has a odd cardinality, from Proposition 16, then the statement is hold. □

Example 7. *Let $P = X_1X_2(X_4 + X_5)$ be a polynomial form of a Boolean function with five variables, with calculus made in the previous proof, we directly deduce:*

$$\begin{aligned}
\mu_{[5]}(P) &= X_1X_2 \times (1 + X_3) \times (X_4 + X_5) \\
&= X_1X_2X_4 + X_1X_2X_5 + X_1X_2X_3X_4 + X_1X_2X_3X_5.
\end{aligned}$$

Thus, we can check on this example that $w_H(\pi_5^{-1}(P)) = 4 = 2^{5-2-1}$.

Proposition 18. *Let I_1 and $I_2 \subset [n], J \subset [n]$ be two subsets such that $I_1 \cap I_2 = I_1 \cap J = I_2 \cap J = \emptyset$, $|I_1| = n_1$ and $I_2 = n_2$. Let $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ be a polynomial such that $P = (X^{I_1} + X^{I_2})\left(\sum_{j \in J} X_j\right)$. Then its Möbius transform is*

$$\left(\sum_{K \subset J, |K|\ odd} X^K\right) \left(\prod_{k \in [n]\setminus(I_1 \cup I_2 \cup J)} (1 + X_k)\right) \left(X^{I_1} \prod_{k \in I_2} (1 + X_k) + X^{I_2} \prod_{k \in I_1} (1 + X_k)\right),$$

and $w_H\left(\pi_n^{-1}(P)\right) = 2^{n-n_1-1} + 2^{n-n_2-1} - 2^{n-(n_1+n_2)}$.

Proof. By Proposition 17

$$\begin{aligned}\mu_{[n]}(P) &= \left(\sum_{I_1\subset L\subset[n]\setminus J} X^L + \sum_{I_2\subset L\subset[n]\setminus J} X^L\right)\left(\sum_{K\subset J,|K|odd} X^K\right)\\ &= \left(\sum_{I_1\subset L\subset[n]\setminus J,I_2\nsubseteq I_2} X^L + \sum_{I_2\subset L\subset[n]\setminus J,I_1\nsubseteq I_2} X^L\right)\left(\sum_{K\subset J,|K|odd} X^K\right).\end{aligned}$$

Since mutual terms X^L satisfy $I_1 \cup I_2 \subset L \subset [n] \setminus J$. $L = (I_1 \cup I_2) \cup L'$, where $L' \subset [n] \setminus (I_1 \cup I_2 \cup J)$. Hence we have $2^{n-|J|-n_1-n_2}$ such L subsets. $\{K \subset J, |K|odd\}$ contains $2^{|J|-1}$ subsets. Therefore, since each mutual term is remove twice, we have $2 \cdot 2^{n-|J|-n_1-n_2} \cdot 2^{|J|-1} = 2^{n-(n_1+n_2)}$ terms to remove. □

Remark 4. *We may generalize this proposition with k subsets I_1, ..., I_k by using the inclusion/exclusion principle.*

We can easily see that the Boolean functions defined as Proposition 17 has an even Hamming weight. Moreover, we can notice that the size of second subset J does not act in the Hamming weight.

Example 8 (Example 7 continued)**.** *Let $Q = X_1X_2(X_3 + X_4 + X_5)$ be a polynomial form of a Boolean function with five variables, we have:*

$$\begin{aligned}\mu_{[5]}(Q) &= X_1X_2 \times (X_3 + X_4 + X_5 + X_3X_4X_5)\\ &= X_1X_2X_3 + X_1X_2X_4 + X_1X_2X_5 + X_1X_2X_3X_4X_5.\end{aligned}$$

Thus $w_H(\pi_5^{-1}(Q)) = w_H(\pi_5^{-1}(P)) = 4$.

Another important remark is that the Möbius transform of indeterminate on set J produces only monomials with odd degree. Thus we can generalize the previous result to the following proposition.

Proposition 19. *Let $I, J, I', J' \subset [n]$ be four subsets such that $I \cap J = \emptyset = I' \cap J'$, $I \cup J = [n] = I' \cup J'$ and $n_1 = |I|$, $n_1' = |I'|$, moreover n_1 and n_1' has not the same parity. Let $P, P' \in \mathbb{F}_2[X_1, \ldots, X_n]$ be two polynomials such that $P = X^I \left(\sum_{j\in J} X_j\right)$ and $P' = X^{I'} \left(\sum_{j\in J'} X_j\right)$. Then*

$$\mu_{[n]}(P + P') = \mu_{[n]}(P) + \mu_{[n]}(P'),$$

and

$$w_H\left(\pi_n^{-1}(P + P')\right) = 2^{n-n_1-1} + 2^{n-n_1'-1}.$$

Proof. Since n_1 and n_1' has different parity and $[n] \setminus (I \cup J) = \emptyset = [n] \setminus (I' \cup J')$, we can't have equal monomials in $\mu_{[n]}(P)$ and $\mu_{[n]}(P')$; then it could not have some vanishing. Thus the statement is hold. □

Example 9. *Let $f \in \mathcal{F}_5$ be a Boolean function such that its polynomial form is defined as:*

$$\begin{aligned} Q &= X_1X_2X_3X_4 + X_1X_2X_3X_5 + X_2X_4X_1 + X_2X_4X_3 + X_2X_4X_5 \\ &= \underbrace{X_1X_2X_3(X_4+X_5)}_{=P} + \underbrace{X_2X_4(X_1+X_3+X_5)}_{=P'}. \\ \mu_{[n]}(Q) &= X_1X_2X_3(X_4+X_5) + X_2X_4(X_1+X_3+X_5+X_1X_3X_5). \end{aligned}$$

Thus $w_H(\pi_5^{-1}(Q)) = 6 = \underbrace{2^{5-3-1}}_{=w_H(\pi_5^{-1}(P))} + \underbrace{2^{5-2-1}}_{=w_H(\pi_5^{-1}(P'))} = 2+4.$

We propose another generalization of the Proposition 17.

Proposition 20. *Let $I, J, K \subset [n]$ be three subsets of $[n]$ such that I, J, K is a partition of $[n]$, then*

$$\mu_{[n]}\left(X^I.\left(X^J+X^K\right)\right) = X^I\left(X^J\prod_{k\in K}\left(1+X^k\right) + X^K\prod_{j\in J}\left(1+X^j\right)\right).$$

Moreover, the Hamming weight of this associated Boolean function is $2^{|J|}+2^{|K|}$.

Proof.

$$\begin{aligned} \mu_{[n]}\left(X^I.\left(X^J+X^K\right)\right) &= X^I.\mu_{[n]\setminus I}\left(X^J+X^K\right); \\ &= X^I\left(\mu_{[n]\setminus I}\left(X^J\right)+\mu_{[n]\setminus I}\left(X^K\right)\right); \\ &= X^I\left(X^J\prod_{k\in K}\left(1+X^k\right) + X^K\prod_{j\in J}\left(1+X^j\right)\right). \end{aligned}$$

□

The first consequence of the last proposition, we are able to design balanced Boolean functions directly. Moreover, another direct consequence is that the Hamming weight of a Boolean function does not depend of its degree, but here only of the degree of its factorization.

Finally, we conclude this part with a generalization of the previous proposition.

Proposition 21. *Let $I, J, K \subset [n]$ be three subsets of $[n]$ such that $I \cap J = I \cap K = J \cap K = \emptyset$. We denote $L = I \cup J \cup K$, then $\mu_{[n]}\left(X^I.\left(X^J+X^K\right)\right)$ is*

$$\prod_{\ell\in[n]\setminus L}\left(1+X^\ell\right)X^I\left(X^J\prod_{k\in K}\left(1+X^k\right) + X^K\prod_{j\in J}\left(1+X^j\right)\right).$$

Moreover, the Hamming weight of this Boolean function is $2^{n-|L|}\left(2^{|J|}+2^{|K|}\right)$.

All propositions in this section allows us to give directly the Möbius transform and the Hamming weight of particular Boolean functions. Other similar propositions could be useful, we introduce the previous ones which seem to be the most helpful. We have few chances to exploit these propositions for a random Boolean function. However, most of Boolean functions used in practice are not random but design by specific constructions. The following example detail the Boolean function into the design of Achertbahn 128.

Example 10. *Achterbahn 128 is a a synchronous stream cipher algorithm developed by Berndt Gammel, Rainer Göttfert and Oliver Kniffler[13, 22]. It involves a Boolean function f_A with 13 variables, which has good cryptographic properties: balanced, its algebraic degree is 4, correlation immunity of order 8, nonlinearity 3584 and algebraic immunity 4. The polynomial form of f_A may be written with the following factorization*

$$
\begin{array}{l}
X_0 + X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8 + X_9 + X_{10} + X_{11} + X_{12} + \\
X_0X_5 + X_2(X_{10}X_{11}) + X_6(X_5 + X_8 + X_{10} + X_{11} + X_{12}) + X_8(X_4 + X_7 + X_9 + X_{10}) + \\
X_9(X_{10} + X_{11} + X_{12}) + X_{10}X_{12} + X_{12}X_4 + X_0X_5(X_8 + X_{10} + X_{11} + X_{12}) + \\
X_1X_2(X_8 + X_{12}) + X_1X_4(X_{10} + X_{11}) + X_1X_9(X_8 + X_{10} + X_{11}) + \\
X_2X_4(X_8 + X_{10} + X_{11} + X_{12}) + X_2X_7(X_8 + X_{12}) + X_2X_8(X_3 + \\
X_7 + X_{10} + X_{11}) + X_3X_8(X_4 + X_9) + X_4X_7(X_8 + X_{12}) + X_4X_8X_9 + \\
X_4X_{12}(X_3 + X_9) + X_5X_6(X_8 + X_{10} + X_{11} + X_{12}) + \\
(X_1X_2X_3 + X_4X_7X_9)(X_8 + X_{12}) + (X_1X_2X_7 + X_3X_4X_8)(X_8 + X_{12}) + \\
X_1X_3X_5 + X_2X_4X_7)(X_8 + X_{12}) + (X_1X_3X_8 + X_2X_5X_7)(X_8 + X_{12}) + \\
(X_1X_7X_9 + X_2X_5X_7)(X_8 + X_{12}) + (X_1X_5X_7 + X_2X_3X_4)(X_8 + X_{12}) + \\
X_6X_8(X_{10} + X_{11}) + X_6X_{12}(X_{10}X_{11}) + X_8X_9(X_7 + X_{10} + X_{11}) + \\
(X_0X_5X_8 + X_1X_4X_{12})(X_{10} + X_{11}) + (X_0X_5X_{12} + X_2X_3X_9)(X_{10} + X_{11}) + \\
(X_2X_4X_{12} + X_5X_6X_8)(X_{10} + X_{11}) + (X_1X_9X_{12} + X_2X_4X_8)(X_{10} + X_{11}) + \\
(X_1X_8X_9 + X_5X_6X_{12})(X_{10} + X_{11}) + (X_1X_4X_8 + X_2X_9X_{12})(X_{10} + X_{11})
\end{array}
$$

Butterfly algorithm performs the computation of Möbius transform in $13 \times 2^{12} = 53248$ operations. By Proposition 16, the Möbius transform of the sum of all monomials of degree one is the sum of all monomials of odd degree. Then $\mu_{[13]}(\sum_{i=0}^{12} X_i)$ is compute in 2^{12} operations. Concerning monomials of degree 2, we have 7 terms P of the form X_i $(\sum_{j\in J} X_j)$, where $i \notin J$. By Proposition 17, each $\mu_{[13]}(P)$ is compute in 2^{11} operations. Then for monomials of degree 3, we have 18 terms P of the form $X_{i_1}X_{i_2}$ $(\sum_{j\in J} X_j)$, where $i_1, i_2 \notin J$. By again Proposition 17, each $\mu_{[13]}(P)$ is compute in 2^{10} operations. Finally for monomials of degree 4, we have 12 terms $(X_{i_1}X_{i_2}X_{i_3} + X_{i_4}X_{i_5}X_{i_6})(X_{j_1} + X_{j_2})$, where $\{i_1, i_2, i_3, i_4, i_5, i_6\} \cap \{j_1, j_2\} = \emptyset$. By Proposition 18, each $\mu_{[13]}(P)$ is compute in $2^{10} - 2^7$ operations. Hence the total number of operations is $2^{12} + 7 * 2^{11} + 18 * 2^{10} + 12 * (2^{10} - 2^7) = 47616$. We gain 5632 operations, that is a reduction of 10.57%, only rewriting f_A and use previous propositions.

6. Conclusion

The major contribution of our work is to introduce a polynomial form without reference of a specific Boolean function; since the indeterminates indicate the variables which occurs in the ANF and not the number of variables. Which allow us to give a new point of view of the Möbius transform and to manipulate Boolean functions of various number of variables via different Möbius transform operators. We derive from this operators two new algorithms to compute the Möbius transform, which can be view as a reformulation of the famous Butterfly one. Furthermore, after a deeper study of this reformulation, we provide a new algorithm which have a huge speed up for really sparse or dense polynomials. We also explicitly compute the Möbius transform and Hamming weight for some classes of Boolean functions. Finally, we exhibit a subfamily of Boolean functions for which ones their Hamming weight is directly related to the algebraic degree of specific factors.

[1] Georges Boole. The calculus of logic. *Cambridge and Dublin Mathematical Journal*, III:183–98, 1848.

[2] Georges Boole. *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*. George Boole's collected logical works. Walton and Maberly, 1854.

[3] Randal Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[4] Cagdas Calik. anf2weight. `https://github.com/ccalik/anf2weight`, March 2012. software.

[5] Cagdas Calik and Ali Doganaksoy. Computing the weight of a Boolean function from its algebraic normal form. In Tor Helleseth and Jonathan Jedwab, editors, *Sequences and Their Applications*, volume 7280 of *Lecture Notes in Computer Science*, pages 89–100. Springer Berlin Heidelberg, 2012.

[6] Claude Carlet. A larger class of cryptographic boolean functions via a study of the maiorana-mcfarland construction. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 549–564, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[7] Claude Carlet. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, chapter Boolean Functions for Cryptography and Error-Correcting Codes. Cambridge University Press, 2010.

[8] Claude Carlet and Aline Gouget. An upper bound on the number of m-resilient Boolean functions. In *ASIACRYPT*, pages 484–496, 2002.

[9] Claude Carlet and Philippe Guillot. A new representation of Boolean functions. In Marc Fossorier, Hideki Imai, Shu Lin, and Alain Poli, editors,

Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, pages 94–103, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[10] James W Cooley, Peter AW Lewis, and Peter D Welch. Historical notes on the fast fourier transform. *Proceedings of the IEEE*, 55(10):1675–1677, 1967.

[11] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[12] John F Dillon. *Elementary Hadamard difference sets*. PhD thesis, University of Maryland, 1974.

[13] Berndt M. Gammel, Rainer Gottfert, and Oliver Kniffler. The achterbahn stream cipher, 2005.

[14] Carl Friedrich Gauss. Nachlass: Theoria interpolationis methodo nova tractata. *Carl Friedrich Gauss Werke*, 3:265–327, 1866.

[15] Philippe Guillot. *Fonctions courbes binaires et transformation de Mobius*. PhD thesis, University of Caen Basse-Normandie, 1999.

[16] Michael Heideman, Don Johnson, and Sidney Burrus. Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, October 1984.

[17] Tadao Kasami and Nobuki Tokura. On the weight structure of Reed-Muller codes. *IEEE Transactions on Information Theory*, 16(6):752–759, 1970.

[18] Robert L. McFarland. A family of difference sets in non-cyclic groups. *J. Comb. Theory, Ser. A*, 15:1–10, 07 1973.

[19] Parag K. Lala. *Digital circuit testing and testability*. Academic Press, 1997.

[20] D. Steven Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. Möbius transformations of matrix polynomials. *Linear Algebra and its Applications*, 470:120 – 184, 2015. Special Issue In Honor of Leiba Rodman.

[21] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*. Springer Berlin Heidelberg, 1998.

[22] María Naya-Plasencia. Cryptanalyse de Achterbahn-128/80. *CoRR*, abs/cs/0611033, 2006.

[23] Alexandre Niveau and Bruno Zanuttini. Efficient representations for the modal logic S5. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1223–1229, 2016.

[24] Josef Pieprzyk, Huaxiong Wang, and Xian-Mo Zhang. Möbius transforms, coincident Boolean functions and non-coincidence property of Boolean functions. *International Journal of Computer Mathematics*, 88(7):1398–1416, 2011.

[25] Claude. E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28, Issue 1:5998, 1949.

[26] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.). *IEEE Transactions on Information Theory*, 30(5):776–780, Sep. 1984.

[27] Ricardo S. Viera and Vanessa Botta. Orthogonal polynomials and Möbius transformations. https://arxiv.org/abs/1904.10766, April 2019.