Boolean Functions: Cryptography and Applications BFCA'05
Fonctions Booléennes : Cryptographie & Applications

ON THE DECOMPOSITION OF BOOLEAN FUNCTIONS

Gérard H. E. Duchamp[1], Hatem Hadj Kacem[2] and Éric Laugerotte[2]

Abstract. The minimization of a weighted automaton given by its linear representation (λ, μ, γ) taking its letters in an alphabet A and its multiplicities in a (commutative or not) field k, due to Schützenberger, provides the construction of a suffix set P such that the orbit $(\mu(p)\gamma)_{p\in P}$ is a basis of the k-space $\mu(k\langle A\rangle)\gamma$. This allows to study algorithmically the $\mathfrak{S}_n$-module $\mathbb{Z}/2\mathbb{Z}[\mathfrak{S}_n].f$ where $\mathfrak{S}_n$ is the symmetric group which acts on the unknowns $x_1, \ldots, x_n$ by change of variables, and $f(x_1, \ldots, x_n)$ is a boolean function. In this work, we present an algorithm which computes the possible decompositions of f with respect to this action. In case the function f is indecomposable the algorithm gives a proof of indecomposability.

1. Introduction

This contribution is intended to tackle the multifaceted problem of decomposing the Boolean Functions (BF in the sequel). By boolean function we here mean any function $\{0,1\}^n \mapsto \{0,1\}$ which, in the language of Computer Science, is just any function taking a n-bits word as argument and returning a boolean value. These functions are efficiently represented by a BDD (a *Binary Decision Diagram*). This representation can be traced back as far as in the late fifties [14] and was exploited extensively (for the first

[1] LIPN, Université de Paris-Nord, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France. email: gerard.duchamp@lipn.univ-paris13.fr
[2] LIFAR, Université de Rouen, place Émile Blondel, 76821 Mont-Saint-Aignan, France. email: eric.laugerotte@univ-rouen.fr

developpements see [1,2]). The great merit of this coding is that it is extremely concise and also compatible with boolean automata theory [12] to such a point that a measure of hadness has been derived from the consideration of a minimal automaton associated to the BDD of a boolean function [5].

As a BDD is variable-order dependant, we would like here to study the orbit of a boolean function under the action of the (algebra of the) symmetric group on the variables. The set of boolean functions of n-variables is naturally a $\mathbb{Z}_2$ $(= \mathbb{Z}/2\mathbb{Z})$ vector space. Thus, the action of the symmetric group given by permutation of variables can be at once extended by linearity to the algebra $\mathbb{Z}_2[\mathfrak{S}_n] = \mathfrak{A}_n$ and, by Krull-Schmitt's theorem, we get that the orbit of f can be split (uniquely, up to isomorphism) as a direct sum of $\mathfrak{A}_n$ indecomposable submodules. The interest of such a splitting is that the components are monogenous (i.e. generated by a single element). The decomposition reads $\mathfrak{A}_n.f = \oplus \mathfrak{A}_n f_i$ and this yields a decomposition of f using $\mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$ idempotents

$$f = \pi_1.f_1 + \pi_2.f_2 \cdots \pi_k.f_k \tag{1}$$

Surprizingly, a suitable adaptation of Schützenberger's algorithm [16] for the minimization of automata with multiplicities (here with coefficients in $\mathbb{Z}_2$) makes all this computable. We use here half of the minimization process, keeping a note of the relators appearing and then getting a minimal presentation of the module $\mathfrak{A}_n.f$. This process is reminiscent of the theory of noncommutative Gröbner bases [11], but here we need more. We need also to compute idempotents in the transfer algebra $\mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$, which can be done using the reduced basis of the module $\mathfrak{A}_n.f$ previously computed. All the process has been implemented in MuPAD.

The structure of the contribution is the following. In Section 2, we present the main aspects of weighted automaton minimization. In Section 3, we deal with the splitting of modules. After, in Section 4, we present the algorithmic of the decomposition of boolean functions. At the end, in Section 5, an example is given with the numbers of decomposable functions for the first values of n.

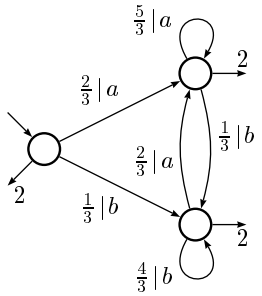

FIGURE 1. A $\mathbb{Q}$-automaton.

2. Minimization of weighted automata

Let us give here a short review of the minimization algorithm from the theory of automata with multiplicities (see also [4, 16] for fields and domains and [10] for a detailed algorithm and an extension to skew fields). An *automaton with multiplicities* $\mathcal{A}$ is a structure equivalent to a triplet (λ, μ, γ) called *linear representation* which is defined by:

- an alphabet (of commands, say) A
- a (finite) set of states Q
- a (semi)ring k of scalars
- an input vector $\lambda \in k^{1\times Q}$
- an output vector $\gamma \in k^{Q\times 1}$
- a mapping $\mu : A \to k^{Q\times Q}$

These data are usually represented as a valued graph (see Figure 1). The mapping μ is at once extended to a morphism from (A^*, conc) to $(k^{Q\times Q}, \cdot)$ where conc stands for the binary operator of concatenation of words and $\cdot$ for the usual matrix multiplication. The number of states of the weighted automaton $\mathcal{A}$ is its *dimension* noted dimension($\mathcal{A}$). Therefore $\mathcal{A}$ is a finite state machine taking words and providing coefficients (called also costs or weights) which are provided by $\lambda\mu(w)\gamma$ for a word $w \in A^*$. The function $A^* \to k$, given by $w \mapsto \lambda\mu(w)\gamma$, can more conveniently be written as a noncommutative series

$$\mathrm{behaviour}(\mathcal{A}) = \sum_{w\in A^*} \lambda\mu(w)\gamma w \tag{2}$$

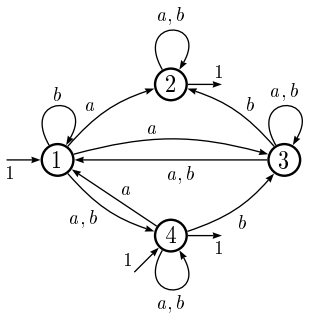

FIGURE 2. A $\mathbb{Z}_2$-automaton

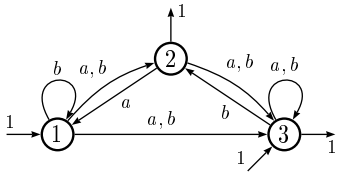

FIGURE 3. A $\mathbb{Z}_2$-automaton

which is called the *behaviour of* $\mathcal{A}$. Such series are just functions $A^* \mapsto k$ called *rational* [4, 8, 15, 17].

The whole set of functions k^{A^*} (noncommutative series) is often denoted $k\langle\langle A\rangle\rangle$ and a function $S \in k\langle\langle A\rangle\rangle$, written in the style of (2) reads

$$S = \sum_{w \in A^*} \langle S|w\rangle w \tag{3}$$

so that $S(w)$ (i.e. the coefficient of w in S) will be denoted as the scalar product $\langle S|w\rangle$. The *behaviour* of $\mathcal{A}$ thus determines the weight of w for the automaton $\mathcal{A}$.

The aim of minimization is to construct an automaton

$$\mathcal{A}_{\min} = (\lambda_{\min}, \mu_{\min}, \gamma_{\min})$$

with the same behaviour and of smallest dimension.

From now on, we set once for all $\mathbb{Z}_2 = \mathbb{Z}/\mathbb{Z}_2$.

The $\mathbb{Z}_2$-automaton given in Figure 2 is minimized in Figure 3.

Minimization is obtained by a left and a right *reduction*. In fact, let $\circ$ be the left action defined for all formal series $S \in k\langle\langle A\rangle\rangle$ and all word $w \in A^*$ by $w \circ S = \sum_{x \in A^*} \langle S|xw\rangle x$. If S is rational, there

exists a finitely generated submodule of $k\langle\langle A\rangle\rangle$ stable for $\circ$ which contains the formal series behaviour($\mathcal{A}$) (this is even a criterium of rationality, see [4,9]). The generators S_i ($i = 1, \ldots,$ dimension($\mathcal{A}$)) may be explicitly given by

$$S_i = \sum_{w \in A^*} (\lambda_i \mu_i(w) \gamma_i) w$$

but in general it is not a family of smallest rank.
Finding algorithmically such a minimal family goes as follows [10]. Call *suffix set* a subset P of the free monoid A^* such that, if a word w belongs to P then every suffix of w belongs to P.
Left reduction of $\mathcal{A}$ allows to construct a suffix set P such that $(\mu(p)\gamma)_{p\in P}$ is a basis of the space of columns $\mu(k\langle A\rangle)\gamma$. The family $(p \circ \text{behaviour}(\mathcal{A}))_{p\in P}$ generates a stable submodule of $k\langle\langle A\rangle\rangle$ which contains behaviour($\mathcal{A}$) and whose the dimension is smaller or equal to dimension($\mathcal{A}$). Indeed, it is the smallest possible among the stable submodules containing behaviour($\mathcal{A}$). More precisely, let $p \in P$ and $a \in A$,

$$a \circ (p \circ S) = \begin{cases} ap \circ S & \text{if } ap \in P, \\ \sum_{q\in P} \alpha^a_{pq}\ q \circ S & \text{if } ap \notin P. \end{cases}$$

To each formal series $p \circ S$ is associated a state in the reduced automaton. The weight of a transition $p \to q$ is the scalar α^a_{pq}, the transition label being a. After left reduction, right reduction is applied and returns the minimized automaton $\mathcal{A}_{\min}$ because $\dim(\lambda_{\min}\mu_{\min}(k\langle A\rangle)) = \dim(\mu_{\min}(k\langle A\rangle)\gamma_{\min})$.

3. Splitting modules

In what follows, we consider the algebra $\mathfrak{A}_n = \mathbb{Z}_2[S_n]$ (we omit the subscript as it is fixed once for all) of the symmetric group S_n over $\mathbb{Z}_2$ [13]. It is generated by the simple transpositions $\sigma_1, \ldots, \sigma_{n-1}$ (σ_i is the transposition of i and $i+1$) and therefore can be presented by generators $(s_i)_{1\le i\le n-1}$ and the relations (the symbol s_i standing for σ_i)

$$\begin{cases} {s_i}^2 = 1, \\ s_i s_j = s_j s_i & \text{if } |i-j| > 1, \\ s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}. \end{cases}$$

(called Moore-Coxeter relations [7, 13]). The algebra $\mathfrak{A}_n$ acts on the left on $\mathbb{Z}_2\langle x_1, \ldots, x_n\rangle$ by change of variables which are morphisms $s_i : A^* \mapsto A^*$ defined on the letters by

$$\begin{cases} s_i x_i = x_{i+1}, \\ s_i x_{i+1} = x_i, \\ s_i x_j = x_j \qquad \text{if } j \neq i, i+1. \end{cases}$$

Let $S = \{s_1, \ldots, s_{n-1}\}$ be the set of symbols of these (simple) transpositions. The morphism $\mathbb{Z}_2\langle S\rangle \to \mathfrak{A}_n$ is onto and then the notions of submodule and decomposition are the same for the action of $\mathfrak{A}_n$ and the action of $\mathbb{Z}_2\langle S\rangle$. For this reason, we will denote similarly (and with no risk of confusion) the two actions. Let $\mathcal{F}_n$ be the left $\mathfrak{A}_n$-module of boolean functions with n variables (it is a finite dimensional $\mathbb{Z}_2$-vector space). We consider the submodule $\mathfrak{A}_n.f$ where $f \in \mathcal{F}_n$ is a single generator. Krull-Schmidt's theorem [6] implies that there exists (unique up to isomorphisms) a splitting of the module $\mathfrak{A}_n.f$ into a direct sum $\mathfrak{A}_n.f = M_1 \oplus \cdots \oplus M_l$ of indecomposable $\mathfrak{A}_n$-submodules M_i. The aim of the algorithm below is to compute a splitting of $\mathfrak{A}_n.f$ by the knowledge of a complete family of orthogonal projectors $\pi_i \in \mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$ $(i = 1, \ldots, l)$ i.e. which satisfy:

$$\begin{cases} \pi_i \circ \pi_i = \pi_i, \\ \pi_i \circ \pi_j = 0 \quad \text{if } i \neq j, \\ \pi_1 \oplus \cdots \oplus \pi_l = 1_{\mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)}. \end{cases}$$

Therefore, the module $\mathfrak{A}_n.f$ is the direct sum of submodules given by $\pi_i(\mathfrak{A}_n.f)$ which are generated by a single element. If π_i is the projector which carries out $\mathfrak{A}_n.f$ to M_i $(M_i = \pi_i(\mathfrak{A}_n.f))$ then π_i must be $\mathfrak{A}_n$-linear.
Let then $\varphi \in \mathfrak{End}_k(\mathfrak{A}_n.f)$. Whether $\varphi \in \mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$ is algorithmically decidable thanks to the fact that the ideal of annihilators of $\mathrm{ann}(f)$ is finitely generated. We explain now how this can be done.

One can construct a suffix set (see below or [10]) $P \subset S^*$ such that $P.f$ is a basis of $\mathfrak{A}_n.f$. Let $E := \{(\sigma_i, \sigma) \in S \times P \mid \sigma_i\sigma \notin P\}$. For $(\sigma_i, \sigma) \in E$, one has:

$$\sigma_i\sigma.f = \sum_{\sigma' \in P} \alpha_{\sigma_i\sigma,\sigma'}\sigma'.f \tag{4}$$

and the differences $R := \left\{\sigma_i\sigma - \sum_{\sigma'\in P}\alpha_{\sigma_i\sigma,\sigma'}\sigma'\right\}_{(\sigma_i,\sigma)\in E}$ are a complete set of generators of $\mathrm{ann}(f)$. The construction of idempotents will rely on the following lemma:

Lemma 3.1. *Let $\varphi \in \mathfrak{End}_k(\mathfrak{A}_n.f)$ and set $f_\varphi = \varphi(f)$. Then the linear transformation φ belongs to $\mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$ iff:*
i) for all $\sigma \in P$, one has $\varphi(\sigma.f) = \sigma.\varphi(f)$
ii) for all $(\sigma_i, \sigma) \in E$, the difference $\left(\sigma_i\sigma - \sum_{\sigma'\in P}\alpha_{\sigma_i\sigma,\sigma'}\right)$ annihilates f_φ.

Thus, it suffices to compute a basis of $\mathfrak{A}_n.f$, keeping track of the relators appearing, to obtain a test which allows to select the idempotents of $\mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$ among the projectors of $\mathfrak{End}_k(\mathfrak{A}_n.f)$.

4. Computation of endomorphisms and projectors

We can transfer the half minimization process to $\mathfrak{A}_n.f$ and also take care of keeping trace of the relators appearing. The following algorithm allows us to find a suffix set of S^* and the corresponding set of relators:

algorithm *suffix*
input the set S
a boolean function $f \in \mathcal{F}_n$
output a suffix set $P \subset S^*$
a set of relators R
$(P, Y, R) := (\emptyset, \{\varepsilon\}, \emptyset)$
while $Y \neq \emptyset$
do take $y \in Y$
  if $my \notin \mathrm{span}(mp : p \in P)$
  then $(P, Y) := (P \cup \{y\}, (Y - \{y\}) \cup yS)$
  else there exits a relation $my = \sum_{p\in P}\alpha_p mp$
    $(P, Y, R) := (P, (Y - \{y\}), R \cup \{y - \sum_{p\in P}\alpha_p p\})$
  end_if
end_while
return(P, R)
end

The set P is a suffix set and the algorithm terminates. In fact, we show that the set P is suffix at each step of the algorithm. This is clear from the beginning when $P = \{\varepsilon\}$. Now, if $y \in Y \subseteq S^*$ is

accepted it must have been so of every suffix of it before. Let $|\sigma|$ denote, as usual, the length of a word $\sigma \in S^*$. As the space $\mathfrak{A}_n.f$ has a finite dimension, it exists a non-negative integer l such that:

$$\mathrm{span}(\sigma.f : \sigma \in S^*) = \mathrm{span}(\sigma.f : \sigma \in S^* \text{ and } |\sigma| < l).$$

One has $P \subseteq \{\sigma \in S^* : |\sigma| < l\}$ and $Y \subseteq \{\sigma \in S^* : |\sigma| < l+1\}$. Then the set Y becomes empty during the algorithm and then the algorithm terminates.

Lemma 4.1. *The set $P.f = \{\sigma.f : \sigma \in P\}$ is a basis of the space $\mathfrak{A}_n.f$.*

Proof. Let $C = SP \setminus P$ be the complete suffix code associated to P [3]. One has the decomposition $S^* = P \sqcup S^*CP$ of the free monoid. Let $\sigma_i\sigma \in C$. Then there exists a relator $\sigma_i\sigma - \sum_{\sigma' \in P} \alpha_{\sigma'}\sigma' \in R$. Now let $\sigma_i \in S$ and $\sigma_i\sigma \in CS^*S$. One has by induction:

$$\begin{aligned}\sigma_i\sigma f &= \sum_{\sigma' \in P} \alpha_{\sigma'}\sigma_i\sigma' f \\ &= \sum_{\sigma_i\sigma' \in P} \alpha_{\sigma'}\sigma_i\sigma' f + \sum_{\sigma_i\sigma \notin P} \sum_{\sigma'' \in P} \alpha_{\sigma'}\beta^{\sigma''}_{\sigma_i\sigma'}\sigma'' f.\end{aligned}$$

And then $\sigma_i\sigma f \in \mathrm{span}(\sigma f : \sigma \in P)$ which ends the proof. □

By Lemma 4.1, Algorithm *suffix* computes a complete description of the space $\mathfrak{A}_n f$. We can observe that the set R of relators depends on the choice of words $y \in Y$. Let $f = x_1x_2 + x_1 \in \mathcal{F}_3$. The set of relators are different if the words are choosen with the graded or with the usual lexicographic order. In fact, the element $\sigma_2\sigma_2 + \varepsilon \in S^*$ is a relator with the use of the second order but not with the first. See Figure 4 where a full transition means an action giving an new element of the basis, a dotted transition giving a relator.

Lemma 4.2. *The ideal generated by the set of relators R is then* $\mathrm{ann}(f)$.

In [11], the tools for the proof of Lemma 4.2 are presented. Therefore we associate at $\varphi \in \mathfrak{End}(\mathfrak{A}.f)$ an unique element $f_\varphi = (\sum_{\sigma \in P} \alpha_{\sigma,\varepsilon}\sigma f)$ [6]. By Lemma 4.1, it is easy to determine algorithmically the endomorphism φ in the basis $(\sigma f)_{\sigma \in P}$ of $\mathfrak{A}f$. In fact,

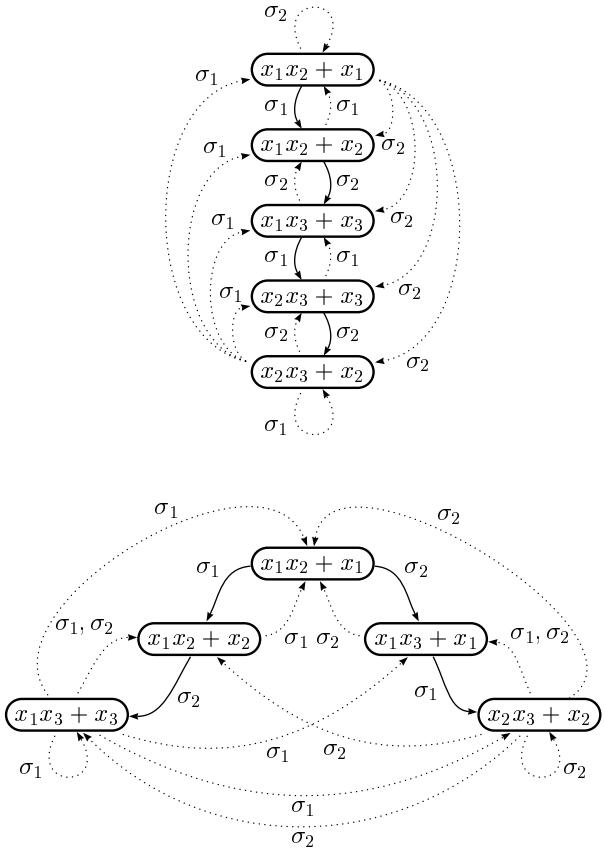

FIGURE 4. Suffix sets and relators for $f = x_1x_2 + x_1$

Algorithm *suffix* allows to compute the suffix set P and the relator set R. By linearity, for any element $\sigma' \in P$, the endomorphism φ depends only to the unknowns $\alpha_{\sigma,\varepsilon}$. In fact, one has:

$$\varphi(\sigma' f) = \sum_{\sigma \in P} \alpha_{\sigma,\varepsilon} \sigma' \sigma f = \sum_{\sigma'\sigma \in P} \alpha_{\sigma,\varepsilon} \sigma'\sigma f + \sum_{\sigma'\sigma \notin P} \alpha_{\sigma,\varepsilon} \sum_{\sigma'' \in P} \beta^{\sigma''}_{\sigma'\sigma} \sigma''.$$

The scalars $\beta^{\sigma''}_{\sigma'\sigma}$ are given by the relators. The order of computation of vectors $\varphi(\sigma f)$ is given by the entry of σ in the set P. If $\sigma = \sigma_i\sigma'$ then $\varphi(\sigma' f)$ will be known. The morphism φ is computed by the following algorithm:

algorithm *cons_*φ
input the set of simple transitions S
the suffix set P
the set of relators R
output the morphism φ
$Y := S$
$\varphi(\varepsilon f) := \sum_{\sigma\in P} \alpha_{\sigma,\varepsilon}\sigma f$
while $Y \neq \emptyset$
do take $\sigma_i\sigma \in Y$
if $\sigma_i\sigma \notin P$
then $Y := Y - \{\sigma_i\sigma\}$
else $\varphi(\sigma_i\sigma f) := \sigma_i\varphi(\sigma f)$
$Y := (Y \cup S\sigma_i\sigma) - \{\sigma_i\sigma\}$
end_if
end_while
return(φ)
end

In order to find projectors, we must study the relation of idempotence $\varphi^2 = \varphi$. Moreover we must verify that $\varphi(\sigma f) = \sigma\varphi(f)$ for all element $\sigma \in P$ and $r\varphi(f) = 0$ for all relator $r \in R$ as $\varphi \in \mathfrak{End}_{\mathfrak{A}_n}(\mathfrak{A}_n.f)$. Therefore we obtain a system of $n \times |P|^2 + |P|$ equations in the unknowns $\alpha_{\sigma,\varepsilon}$ for all $\sigma \in P$. Each non-trivial solution φ gives a decomposition $\mathfrak{A}_n.f = \mathfrak{A}_n.f_\varphi \oplus \mathfrak{A}_n.f_{1-\varphi}$. In this case, we restart the algorithm on $\mathfrak{A}_n.f_\varphi$ and $\mathfrak{A}_n.f_{1-\varphi}$. We get by repetition a direct sum of indecomposable submodules. Otherwise, if no non-trivial solutions exists, we deduce that the module $\mathfrak{A}_n.f$ can not be written in a direct sum of submodules non-zero submodules.

Theorem 4.3. *Let $f \in \mathcal{F}_n$. A finite repetition of Algorithm suffix and cons_φ decides if there exits a decomposition of $\mathfrak{A}_n.f$ in direct sum of indecomposable submodules. If the algorithm finds no non-trivial decompoition at the first step, then the module $\mathfrak{A}_n.f$ is indecomposable.*

The preceding process can be applied mutatis mutandis with $\mathfrak{A}$ any finitely generated associative algebra with unit over a field k acting on a finite dimensional (as a k-vector space) $\mathfrak{A}$-module.

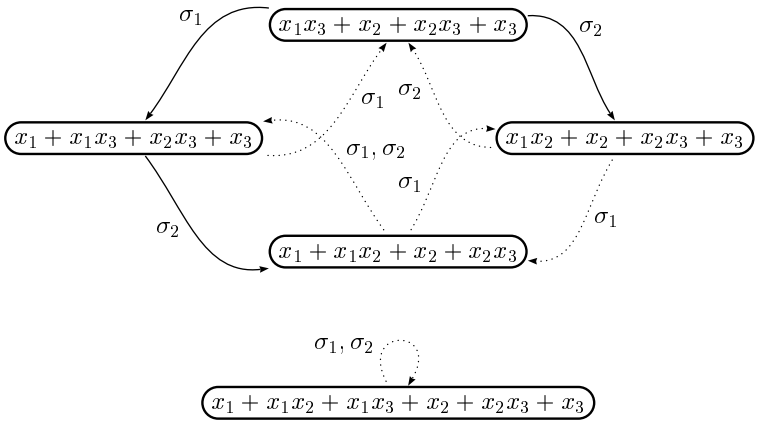

FIGURE 5. Indecomposable modules

5. Example of splitting

We consider now the boolean function $f = x_1x_2 + x_1 \in \mathcal{F}_3$. A basis of the module $\mathfrak{A}_3.f$ is presented in Figure 4 when the graded lexicographic order is choosen for the computation of the suffix set P. Let $\varphi \in \mathfrak{End}_k(\mathfrak{A}_3.f)$ and $f_\varphi \in \mathfrak{A}_3.f$ such that $f_\varphi = \alpha_\epsilon \epsilon.f + \alpha_{\sigma_1}\sigma_1.f + \alpha_{\sigma_2}\sigma_2.f + \alpha_{\sigma_2\sigma_1}\sigma_2\sigma_1.f + \alpha_{\sigma_1\sigma_2}\sigma_1\sigma_2.f$. The matrix corresponding to φ is:

$$\begin{pmatrix}
\alpha_\varepsilon & \alpha_{\sigma_1}+\alpha_{\sigma_1\sigma_2} & \alpha_{\sigma_2}+\alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_2}+\alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_1}+\alpha_{\sigma_1\sigma_2} \\
\alpha_{\sigma_1} & \alpha_\varepsilon+\alpha_{\sigma_1\sigma_2} & \alpha_{\sigma_1\sigma_2}+\alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_2} & \alpha_{\sigma_1}+\alpha_{\sigma_2} \\
\alpha_{\sigma_2} & \alpha_{\sigma_1\sigma_2}+\alpha_{\sigma_2\sigma_1} & \alpha_\varepsilon+\alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_1}+\alpha_{\sigma_2} & \alpha_{\sigma_1} \\
\alpha_{\sigma_1\sigma_2} & \alpha_{\sigma_1\sigma_2} & \alpha_{\sigma_1}+\alpha_{\sigma_2\sigma_1} & \alpha_\varepsilon+\alpha_{\sigma_2} & \alpha_{\sigma_1}+\alpha_{\sigma_2\sigma_1} \\
\alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_2}+\alpha_{\sigma_1\sigma_2} & \alpha_{\sigma_2\sigma_1} & \alpha_{\sigma_2}+\alpha_{\sigma_1\sigma_2} & \alpha_\varepsilon+\alpha_{\sigma_2}
\end{pmatrix}$$

Non-trivial solutions of the system given by $\varphi^2 = \varphi$ and $\sigma\varphi(f) = \varphi(\sigma.f)$ for all $\sigma \in P$ are $f_\varphi = \sigma_2\sigma_1.f + \sigma_1\sigma_2.f$ and $f_{1+\varphi} = \varepsilon.f + \sigma_2\sigma_1.f + \sigma_1\sigma_2.f$. Or else, $f_\varphi = x_1x_3 + x_2 + x_2x_3 + x_3$ and $f_{1+\varphi} = x_1 + x_1x_2 + x_1x_3 + x_2 + x_2x_3 + x_3$. In fact, one has $\mathfrak{A}_3.f = \mathfrak{A}_3.f_\varphi + \mathfrak{A}_3.f_{1+\varphi}$, and the indecomposable modules $\mathfrak{A}_3.f_\varphi$ and $\mathfrak{A}_3.f_{1+\varphi}$ are expressed by Figure 5.

First results of experiments are presented in the following table:

Nb of unknowns	1	2	3	4
Nb of functions	1	16	256	65536
Nb of dec. functions	0	0	82	683
% of dec. functions	0	0	32.03	1.04

6. Concluding remarks

The linear representation of the action of the symmetric group by change of variables of a boolean function has been studied with respect to indecomposability and using $\mathbb{Z}_2$ coefficients. We have got a presentation of the module generated by a boolean function by means of an algorithm designed by Schützenberger for the minimization of automata with multiplicities and a suited recording of the relators appearing during the computation. The whole process has been implemented in MuPAD.

References

[1] S. B. Akers, *Binary decision diagrams*, IEEE Transactions on Computers, **27**, 1978.
[2] R. E. Bryant, *Graph-based algorithms for boolean function manipulation*, IEEE Transactions on Computers, **35**, 1986.
[3] J. Berstel, D. Perrin, *Theory of codes*, Academic Press, 1985.
[4] J. Berstel, C. Reutenauer, *Rational series and their languages*, Springer, 1988.
[5] J.-M. Champarnaud, J.-F. Michon, *Automata and binary decision diagrams*, Workshop on Implementing Automata, 178-182, 1998.
[6] C.W. Curtis, I. Reiner, *Methods of representation theory*, Wiley Interscience, 1990.
[7] G. Duchamp, D. Krob, A. Lascoux, B. Leclerc, T. Scharf, J.Y. Thibon, *Euler-Poincaré characteristic and polynomial representations of Iwahori-Hecke algebras*, Pub. of the RIMS, **31**, 1995.
[8] S. Eilenberg, *Automata, languages and machines*, vol A, Academic Press, 1974.
[9] M. Fliess, *Matrices de Hankel*, Journal of Mathematics Pures and Applied, **53**, 197-222, 1974.
[10] M. Flouret, É. Laugerotte, *Noncommutative minimization algorithms*, Information Processing Letters, **64**, 123-126, 1997.
[11] G. Melançon, *Réécritures dans l'algèbre de Lie libre, le groupe libre et l'algèbre associative libre*, Monographies du LACIM, 1991.

[12] S. Fortune, J. E. Hopcroft, E. M. Schmidt, *The complexity of equivalence and containment for free single variable program schemes*, LNCS, Proceedings of the Fifth Colloquium on Automata, Languages and Programming, **62**, 227-240, 1978.
[13] J. E. Humphreys, *Reflection groups and Coxeter groups*, Cambridge University Press, 1990.
[14] C. Y. Lee, *Representation of switching circuits by binary-decision programs*, Bell Systems Technical Journal, 38:985999, 1959.
[15] J. Sakarovitch, *Éléments de théorie des automates*, Vuibert, 2003.
[16] M. P. Schützenberger, *On the definition of a family of automata*, Inform. and Contr. **4**, 245-270, 1961.
[17] R. P. Stanley, *Enumerative combinatorics*, Cambridge University Press, 1999.