

On Maximal Boolean Complexity of Boolean Functions^{*}

Jean-Francois Michon¹, Jean-Baptiste Yunes², and Pierre Valarcher¹

¹ Université de Rouen, LIFAR, 75821 Mont Saint Aignan Cédex, France
jean-francois.michon@univ-rouen.fr pierre.valarcher@univ-rouen.fr

² Université Paris 7, LIAFA, 175 rue du Chevaleret, 75013 Paris, France
Jean-Baptiste.Yunes@liafa.jussieu.fr

Abstract. We investigate the structure of “worst-case” quasi reduced ordered decision diagrams (or boolean graphs) and boolean functions whose truth tables are associated to: we suggest different ways to count and enumerate them. We, then, introduce a notion of complexity which leads to the concept of “hard” boolean functions as functions whose boolean graphs are “worst-case” graphs. So we exhibit the surprising relation between hard functions and the Storage Access function (also known as Multiplexer). We also show some interesting properties of the hard functions and their graphs like the degree of the polynomial representation or the preservation of the hardness nature of the graph through variable permutations.

1 General background of the problem

The complexity of boolean functions is a central subject of information theory. In theoretical computer science, the term complexity usually refers to the size of a chosen representation of an object (or even some part of this description). There is a lot of such representation for boolean functions like: truth table, boolean circuit, binary decision diagrams, normal disjunctive and conjunctive forms, etc. We focus here on QROBDD and ROBDD representations.

The binary decision diagram (BDD) representation was introduced by Lee in 1959 but the rise of its success began with Bryant’s thesis [Bry86] giving theorems and good algorithms to handle them. It’s now a widespread tool for boolean function manipulation with many application areas such as verification or reliability studies and today the term BDD covers a large family of different representations: QROBDD, ROBDD, FBDD, ZBDD, etc. We refer the reader to Bryant’s web site and Wegener’s book [Weg00] for exhaustive study.

We are concerned here with the most primitive BDD: the quasi reduced ordered BDD (QROBDD) and reduced ordered BDD (ROBDD). They are directed acyclic graphs canonically associated to a given boolean function and we quickly sketch the theoretical construction of the QROBDD graph:

^{*} Work partially supported by "ACI Cryptologie - Ministère de la Recherche (France)"

Starting from the truth table of f (canonically associated to f), we construct the binary tree associated to f whose leaves are labeled by the values (0 or 1) of f , that is to say the data contained in the truth table of f . Then, identifying all the isomorphic subgraphs of this tree (this process is sometimes called the *merging rule*) we get a directed acyclic graph called the QROBDD of f . Bryant work (or the minimal automata theorem) shows that the order in which the identifications of subtrees are made have no impact on the final result: it's a canonical graph associated to f .

Now, let \mathcal{B}_n the set of boolean functions in n boolean variables. The number of vertices of the QROBDD graph of $f \in \mathcal{B}_n$ is called the QROBDD-complexity of the function, and denoted $c_{\text{QROBDD}}(f)$ (or $c(f)$ if the context is clear) in the following. The reader will then find immediately the trivial bound:

$$n + 1 \leq c_{\text{QROBDD}}(f) \leq 2^n + 1$$

The lower bound is obviously exact but the upper bound is not. The true upper bound, say $W(n)$ (for “worst-case”), can be effectively computed and was studied by many people until recently (see the papers of Gröpl, Prömel and Srivastav [GPS98,Grö99,GPS01]).

Our goal is the description of the family \mathcal{H}_n of boolean function in n variables achieving this maximal QROBDD-complexity. We shall often use the term of “hard” functions for them.

The main result is that \mathcal{H}_n can be precisely described and enumerated for any n . This is very unexpected: for example nothing similar occurs for circuit complexity! Another surprise is the appearance of the Storage Access (SA) function (see [Weg87] p. 76). It has been used by W. Paul [Pau77] to prove the first linear lower bound in circuit complexity of explicitly given families of boolean functions. It appears as the simplest among the hard functions and we are able to show that, for some special values of n , \mathcal{H}_n is exactly the family of “twisted” SA functions by a whole symmetric group. We also study the effect of the ordering of variables on \mathcal{H}_n and give elementary properties.

The work of Gröpl (cited above) presented in STACS'98, is centered around the “Shannon effect” for QROBDD-complexity. The well known theorem of Shannon (1949) says that random boolean functions in n variables have almost maximal boolean circuit complexity. They study the analog theorem for QROBDD-complexity and use statistical tools. Our results bring very complementary informations on the worst case QROBDD investigations by combinatorial means.

The most standard representation of a boolean function f is the boolean circuit representation (see [Weg87] or [Spi97]). It permits to define the circuit complexity of f as the minimal size (in gates) of the circuits computing f and denoted $c_{\text{circuit}}(f)$. Unlike the truth table representation, the circuit representation gives a fundamental tool for discriminating between boolean functions. Unfortunately no algorithm is known to compute in reasonable time the circuit complexity of a random f . Nevertheless, one can show that:

$$c_{\text{circuit}} \leq (1 + o(1)) \frac{2^n}{n}$$

and a classical theorem [Sha49] says that for almost every $f \in \mathcal{B}_n$ we have $c_{\text{circuit}}(f) \geq \frac{2^n}{n}$. This statistical behaviour, is referred as the “Shannon effect” by Lupanov: if we randomly take a boolean function, it has nearly maximal circuit complexity.

2 Canonical reduced graphs of boolean functions (QROBDD)

2.1 Definitions

Definition 1 *A boolean function (in n variables, $n \geq 0$) is any:*

$$f : \{0, 1\}^n \rightarrow \{0, 1\}.$$

We call their set \mathcal{B}_n : it has 2^{2^n} elements. We use also $\mathcal{B}_{m,n}$ as the set of n -uples of boolean functions in m variables: it is also the set of all functions from $\{0, 1\}^m$ to $\{0, 1\}^n$.

We must immediately point out that the variables of a boolean function have an implicit natural ordering, say the order given in the definition of $f(x_0, \dots, x_{n-1})$, if $n \geq 1$.

From the truth tree representation of f one can deduce another more compact canonical graph representation:

Definition 2 *A boolean graph is a graph G satisfying the following properties:*

1. G is finite and directed.
2. Any vertex is reachable from a unique one called the “root” of G .
3. Each leaf (vertex with no successor) is labelled “0” or “1” (values).
4. From each non-leaf vertex outcome exactly two edges labeled “0” (commonly called the left edge) and “1” (right edge) (vertices of outdegree 2).
5. All paths from the root to a vertex have the same length (the graph is said to be complete).

As a consequence of these properties the graph is **acyclic**. The number of vertices is called the **size** of the boolean graph. The distance from a vertice to the root is called the **height** of the vertice. The set of all the vertices of height d , ($0 \leq d \leq n$), is called a **level** of the boolean graph.

To any boolean function in n variables one can associate an *unique* boolean graph whose core structure is a binary tree and in which the value of a leaf is exactly the value of the function at the point corresponding to the unique n -uple labelling the path from the root to that leaf. That graph is exactly the **truth tree** of the function and its size obviously is $2^{n+1} - 1$.

We can compare our structure to the “function graphs” used by Bryant. The only difference is that Bryant need an additional label (called variable index) on the vertices. This comes from the fact that our edges can only connect vertices on consecutive levels (a consequence of the fifth property of boolean graphs),

but Bryant's allows some kind of *shortcuts* for paths in which some variables are not significant. A consequence is that boolean graphs are bigger (in number of vertices) than BDD.

The same remark gives small differences with automata in the management of a sink state. But these rather small differences allow us to give nice formulas for enumeration of "hard" boolean functions.

Let G a boolean graph, a boolean subgraph of G is a subgraph of G which is boolean.

A morphism from the boolean graph G to a boolean graph G' is a morphism of graph from G to G' respecting the labelling of edges and the values.

Definition 3 *A reduction is a morphism from G to G' with $size(G') \leq size(G)$. We say that G' is a reduction of G .*

We say that the reduction is **strict** if $size(G') < size(G)$. The composition of reductions is a reduction. An easy way to understand what a reduction is is to think of them as identifications of isomorphic boolean subgraphs. A boolean graph is called **irreducible** if no strict reduction can be defined on it.

It is well known that:

Theorem 1 *For any boolean graph G there exists a unique irreducible boolean graph which is a reduction of G .*

One can even say more: reduction is a confluent operation. This signifies that if G' and G'' are reductions of G there exist a common reduction G''' of G' and G'' .

Definition 4 *The reduced graph of a boolean function f is the irreducible boolean graph associated to its truth tree. We call it the **reduced boolean graph** of f . Its size is called the **(binary) complexity** of f and written $c(f)$.*

The reduced boolean graph of f is also called the Quasi Reduced Ordered Binary Decision Diagram (QROBDD) of f .

It's clear that we can recover the truth tree of f from its reduced boolean graph. So reduced graphs of distinct boolean functions are distinct.

The reduction process may identify only some of the vertices having same height. For instance the 2^n leaves of the binary tree will be identified in (at most) 2 leaves labelled 0 and 1 in the reduced graph of f .

Then we have:

Proposition 1 *Let $r_i(f)$ the number of vertices in the level i ($0 \leq i \leq n$) of the reduced graph of f , then:*

$$1 \leq r_i(f) \leq 2^i$$

$$r_0(f) = 1, \quad r_n(f) = 2 \text{ or } 1$$

$$c(f) = r_0(f) + \dots + r_n(f)$$

We give the distribution of complexities $c(f)$ for $n \leq 4$ in Appendice A.

2.2 Reduced graphs and ROBDD

We refer again to the classical papers [Bry86] and [Weg00] for a complete presentation of ROBDD.

A ROBDD is obtained from the preceding reduced graph by applying another kind of reduction (formerly called *shortcuts*): if edges outcoming from a vertex V of height k points to the same vertex W of height $l > k$ then the vertex V and its two outcoming edges are removed, and all the previously incoming edges of V are redirected to vertex W (shortcutting all the paths formerly passing through V). The important fact is that this new graph structure must keep a record of the height of the vertice W . Then the structure one deals with, in the theory of ROBDD, is not our boolean graph but boolean graph with height for each vertex (Bryant call this graphs function graphs, and the height parameter is called index as it stands for variable index).

The complexity of the ROBDD is defined as the number of vertices of the ROBDD and we denote this number with $c_{\text{ROBDD}}(f)$. So, we have:

$$c_{\text{ROBDD}}(f) \leq c(f)$$

3 Hard boolean functions

Definition 5 *A boolean function f in n variables is hard if $c(f)$ is maximal among the complexities of all functions in \mathcal{B}_n . Let \mathcal{H}_n be the set of all hard boolean functions in n variables and $C(n) = \max_{f \in \mathcal{B}_n} c(f)$ the complexity of those hard functions.*

The number $C(n)$ is easily computable as follows.

For each integer n , let $R_n : [0..n] \rightarrow \mathbb{N}$ defined by

$$0 \leq k \leq n, R_n(k) = \inf(2^k, 2^{2^{n-k}})$$

It's easy to see that R_n is a function which first increase from 1 up to some point where it then decrease to 2. Then we define:

Definition 6 *The heigth of inflexion of $n \geq 2$, as the unique integer $i \leq n$ such that:*

$$2^{i-1} < 2^{2^{n-(i-1)}} \quad \text{and} \quad 2^i \geq 2^{2^{n-i}}$$

We denote $h(n)$ such i and we define $h(0) = 0$ and $h(1) = 1$.

Theorem 2 [CP89] $C(n) = \sum_{k=0}^n R_n(k)$

Proof: There is exactly 2^{2^k} possible distinct truth trees of height k (the number of distincts boolean functions in k variables). In a binary tree of height n the subtrees of height k are those which root is a vertex of height $n - k$ then $c(n) \leq \sum_{k=0}^n R_n(k)$.

The equality will follow from the construction of an irreducible boolean graph of complexity exactly equal to the right hand side.

If $n = 1$ it's clear.

If $n > 1$, we first construct a binary tree of height $h_1 = h(n) - 1$. We then add $R_n(h_2) = 2^{2^{n-h_2}}$ vertices of height $h_2 = h(n)$ and connect vertices at level h_1 to vertices at level h_2 . Such a connection must verify the following three conditions:

1. two edges exactly come out from each vertex of height h_1 (by definition of a boolean graph),
2. two vertices of height h_1 have distinct (ordered) pairs of vertices (left son, right son) (the two boolean subgraphs are irreducible one to the other),
3. each vertex of height h_2 receives at least one edge from a vertex of height h_1 (each vertex at h_2 is reachable from the root).

We can always construct such edges because the number of ordered pairs of vertices we can choose with height h_2 is:

$$R_n(h_2)^2 = 2^{2^{n-(h_2-1)}} = 2^{2^{n-h_1}} \geq R_n(h_1)$$

We then iterate the construction with a new level connecting $R_n(h_1 + 1 = h_2)$ vertices to $R_n(h_2 + 1)$ new vertices, and so on, until height n . One can note that $R_n(h_2)^2 = R_n(h_1)$ holds whenever $h_1 > h(n)$ and that $R_n(h_2)^2 > R_n(h_1)$ when $h_1 = h(n)$.

This boolean graph is clearly irreducible. \square

Corollary 1 $\forall n \geq 0, C(n) = 2^{h(n)} - 1 + \sum_{i=0}^{n-h(n)} 2^{2^i}$

Definition 7 For all integer $n \geq 1$ we call $(R_n(h(n)), R_n(h(n) - 1))$ the **inflexion pair**. The **upper part** (resp. **inflexion zone**, **lower part**) of the reduced boolean graph of a hard function is the subgraph consisting of nodes of height $\leq h(n) - 1$ (resp. $h(n) - 1$ and $h(n)$, $\geq h(n)$) and edges between them.

When n varies the inflexion pair evolves regularly:

Theorem 3 Let (m, k) the inflexion pair of n , then, when n takes all values between $a + 2^a$ and $a + 2^{a+1}$, i.e. $n = a + 2^a + b$ with $0 \leq b \leq 2^a$ ($a, b \in \mathbb{N}$):

1. $h(n) = 2^a + b = n - a$
2. m stays constantly equals to 2^{2^a}
3. $k = 2^{2^a+b-1} = 2^{n-a-1}$
4. $\frac{m}{2} \leq k < m^2$

Proof: Clear from the preceding theorem and definitions. \square

The asymptotic behaviour of $C(n)$ is easily derived from the theorem due to Champarnaud and Pin [CP89] who studied the case of finite minmax automata:

Theorem 4 Every real value in $[1, 2]$ is the limit of a convergent subsequence of $s_n = \frac{C(n)n}{2^n}$.

4 Simple properties of hard functions

Theorem 5 *If f is a hard function so does $1 + f$.*

Proof: A reduced boolean graph of $1 + f$ is obtained from the graph of f by exchanging the two leaves labelled “0” and “1”. \square

Theorem 6 *A hard function depends on its n variables except when n is of the form $1 + a + 2^a$. In this case, there exists hard functions in n variables which depend only on $n - 1$ variables. Their form is:*

$$f(x_0, \dots, x_{n-1}) = g(x_0, \dots, x_{h(n)-2}, x_{h(n)}, \dots, x_{n-1})$$

where $g \in \mathcal{H}_{n-1}$.

Proof: If $n = 1 + a + 2^a$, $h(n) = 2^a + 1$ then the inflexion pair is $(2^{h(n)-1}, 2^{h(n)-1})$. So we can choose to connect each vertex at $h(n) - 1$ to each corresponding vertex at $h(n)$ with double edges. This implies that the function does not depend of the variable $x_{h(n)-1}$.

Conversely, if the function does not depend on the variable x_j the edges between levels $j - 1$ and j are all double. From the structure of the reduced graphs of hard functions, this may only occur in inflexion zone. Then $j = h(n) - 1$ and the inflexion pair is $(2^{h(n)-1}, 2^{h(n)-1})$. \square

Corollary 2 *There is $2^a!$ hard functions in $n = 1 + a + 2^a$ variables which depends in only $n - 1$ variables.*

For example:

1. $g(x, y) = x$ is hard because $f(x) = x$ is hard and $2 = 1 + 0 + 2^0$.
2. $h(x, y, z, t) = x + yt + xt$ is hard because $f(x, y, z) = x + yz + xz$ is hard and $4 = 1 + 1 + 2^1$.

Theorem 7 *The degree of a hard function is $\geq n - h(n)$. It grows to infinity with n .*

Proof: Let f a hard function in n variables. For all boolean functions of $n - h(n)$ variables $g(x_{h(n)}, \dots, x_{n-1})$, there exist boolean values $a_0, \dots, a_{h(n)-1}$ such that $f(a_0, \dots, a_{h(n)-1}, x_{h(n)}, \dots, x_{n-1}) = g(x_{h(n)}, \dots, x_{n-1})$ (this results from the surjectivity of the connections in the inflexion zone), then choosing $g = x_i x_{i+1} \cdots x_{n-1}$ leads to $\deg(f) \geq n - h(n)$. \square

Corollary 3 *Quadratic boolean functions are not hard if $n \geq 11$.*

5 Enumeration of hard functions

We may precise now the preceding construction.

In the upper part, connections between vertices are determined as the reduced graph is a binary tree. In the inflexion zone things are more complex

as we will see in the following but before studying this, we can look at what happens in the lower part. As we already noticed, if $h(n) \leq j \leq n - 1$ the number of vertices at level j is the square of those at level $j + 1$, up to $j = n$ (by construction). All possible connections are used, and we can say that, by ordering the vertices of height j , there is only one way to connect level j to level $j + 1$.

Consequently, the number of hard functions is just the number of different connections in the inflexion zone.

To compute this number let (m, k) be the inflexion pair. The problem is to connect k vertices to m vertices with respect to the three conditions given in the proof of Theorem 2. And this can be clearly restated as the following combinatorial problem:

Problem 1 *Let a grid of $m \times m$ checkable boxes. In how many ways can we check k different boxes so that, for any s such that $1 \leq s \leq m$, a row or a column of index s has at least a checked box?*

We call such a checked boxes configuration a (m, k) -**correct** configuration. For example, $\begin{array}{|c|c|} \hline \times & \times \\ \hline \times & \times \\ \hline \end{array}$ is $(3, 3)$ -correct but $\begin{array}{|c|c|} \hline \times & \times \\ \hline \times & \\ \hline \end{array}$ is not.

Theorem 8 *Let $C(m, k)$ the number of (m, k) -correct configurations, it satisfy:*

1. if $k > m^2$ or $k < \frac{m}{2}$ then $C(m, k) = 0$
2. else

$$C(m, k) = \binom{m^2}{k} - \sum_{j=1}^m \binom{m}{j} C(m - j, k)$$

Proof: Consider all possible k -checked configuration in the $m \times m$ grid and then subtract all incorrect configurations. First subtract the ones which miss exactly one index: the ones which are $(m - 1, k)$ -correct when deleting the row and the column of index $1 \leq s \leq m$. Then subtract those missing exactly two indexes, etc. \square

Remark that if $(m - 1)^2 < k \leq m^2$ then $C(m, k) = \binom{m^2}{k}$.

We can now conclude from these results:

Theorem 9 *The number of hard boolean functions in n variables is $k! C(m, k)$ where (m, k) is the inflexion pair associated to n .*

Proof: The order with which we check the boxes must be considered as it is the order used to connect vertices at level $h(n) - 1$ with k pairs of vertices at level $h(n)$. There is $k!$ possible such permutations. \square

We now give another formula for the $C(m, k)$ computation as a kind of Pascal triangle formula.

Theorem 10 *$C(m, k)$ satisfy:*

1. if $k > m^2$ or $k < \frac{m}{2}$ then $C(m, k) = 0$

2. *else*

$$\begin{aligned} kC(m, k) = & (m^2 - k + 1)C(m, k - 1) \\ & + m(2m - 1)C(m - 1, k - 1) \\ & + m(m - 1)C(m - 2, k - 1) \end{aligned}$$

Proof: We first multiply the two sides by $(k - 1)!$ and get in the left hand side the number of correct configurations of k orderly checked boxes in the $m \times m$ grid. If we then suppress the k -th checking of this (m, k) -correct configuration, we find a $(m, k - 1)$ -configuration of orderly checked boxes. But one can see that this configuration has one of the following three exclusive types:

1. $(m, k - 1)$ -correct.
2. not $(m, k - 1)$ -correct but $(m - 1, k - 1)$ -correct.
3. not $(m, k - 1)$ -correct, not $(m - 1, k - 1)$ -correct but $(m - 2, k - 1)$ -correct.

In the first case, we can check an arbitrary k -th box in the remaining unchecked boxes (there is $m^2 - (k - 1)$ such boxes) to obtain a (m, k) -correct configuration of orderly checked boxes. The number of these configurations is $(m^2 - k + 1)(k - 1)!C(m, k - 1)$.

In the second case, we just need to add one more arbitrarily chosen row and its associated column (there is m such possible choices) and then check an arbitrary k -th box in one of those new boxes (there is $2m - 1$ such new boxes). The number of these configurations is $m(2m - 1)(k - 1)!C(m - 1, k - 1)$.

In the last case, we need to add one more arbitrarily chosen row (m possible choices) and one more arbitrarily chosen column of a different index ($m - 1$ possible choices) and then check the box at the crossing (only one choice). The number of these configurations is $m(m - 1)(k - 1)!C(m - 1, k - 1)$. \square

6 Permutations of variables and complexity

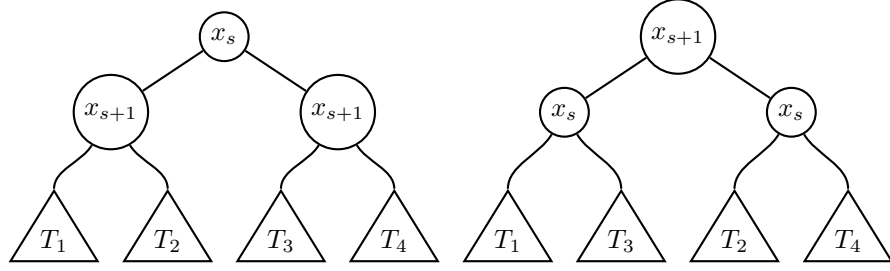
Any permutation $\sigma \in \mathfrak{S}_n$ induces an automorphism of the \mathbb{F}_2 -space \mathcal{B}_n , defined by $f \rightarrow f^\sigma$ where:

$$f^\sigma(x_0, \dots, x_{n-1}) = f(x_{\sigma(0)}, \dots, x_{\sigma(n-1)})$$

Unfortunately, the boolean graph complexity of f is not invariant under this action. For instance, if $f(x, y, z) = x$ then $c(f) = 7$ but if $g(x, y, z) = z$, $c(g) = 5$. Note that this phenomenon exists also for c_{ROBDD} .

Theorem 11 *Let f be a hard function in n variables x_0, \dots, x_{n-1} . If we transpose two consecutive variables x_s and x_{s+1} , provided that $s + 1 \leq h(n) - 2$ or $s \geq h(n)$, the function is still hard.*

Proof: If we permute to successive variables x_s et x_{s+1} with $0 \leq s < s + 1 \leq n - 1$, the effect induced on the binary tree of f is easy to describe as in the following:



Subtrees T_2 and T_3 are exchanged in all subtrees whose root is a vertex of height s . It is clear that no new subtree with root of height $> s + 1$ appears after such a permutation. New subtrees may appear only with roots of height $\leq s + 1$, and new reductions may then be applicable.

Now, suppose that we apply permutation $(x_s x_{s+1})$ with $s + 1 \leq h(n) - 2$ of a hard function. The initial reduced graph upper part is a binary tree (between the height 0 and $h(n) - 1$), thus new subtrees appearing cannot be isomorphic because it would imply that subtrees with root of height $h(n) - 2$ would have been isomorphic but they aren't by definition.

In the same way if we choose to permute two variables in the lower part, i.e. $s \geq h(n)$, we just have to remind that each function in $\mathcal{B}_{n-h(n)}$ is rooted once at level $h(n)$ and that such permutation induces an automorphism of that space. Then after permutation of the two variables the new subgraphs cannot be isomorphic. \square

Let G^+ (resp. G^-) the group of permutations of the variables $x_0, \dots, x_{h(n)-2}$ (resp. $x_{h(n)}, \dots, x_{n-1}$). Of course:

$$G^+ \simeq \mathfrak{S}_{h(n)-1} \text{ and } G^- \simeq \mathfrak{S}_{n-h(n)}$$

and the direct product $G^+ \times G^-$ can be canonically identified with a subgroup of \mathfrak{S}_n (the variable $x_{h(n)-1}$ is invariant by this subgroup). Remark that the two permutations $(x_{h(n)-2} x_{h(n)-1})$ and $(x_{h(n)-1} x_{h(n)})$ are not in $G^+ \times G^-$.

Let G^{++} be the group of permutations of the variables $x_0, \dots, x_{h(n)-1}$. We have $G^{++} \simeq \mathfrak{S}_{h(n)}$.

Theorem 12 *Let f a hard function in n variables and σ any element of the canonical subgroup $G^+ \times G^-$ of \mathfrak{S}_n , then f^σ is hard.*

If $n = a + 2^a$ (with $a \in \mathbb{N}$) then f^σ is hard for any $\sigma \in G^{++} \times G^-$.

Proof: The first assertion follows from the preceding theorem and of the fact that transpositions $(s s + 1)$ for $1 \leq s \leq n - 1$ generate the group \mathfrak{S}_n .

When $n = a + 2^a$ just remark that the reduced graph is a binary tree up to level $h(n)$ and not only up to $h(n) - 1$ as in other cases. Then arguments used in the proof of the previous theorem can be used to show that the permutation $(x_{h(n)-2} x_{h(n)-1})$ preserves hardness. \square

Theorem 13 *No hard function is invariant under one permutation of type $(i i + 1)$ in G^+ . In particular, for $n \geq 3$, no symmetric boolean function in n variables is hard.*

Proof: Invariance by such a permutation would imply that some subtrees with roots in the “binary tree” upper part of the reduced graph can be identified. This is a contradiction with the fact that the graph is reduced. \square

7 Relation with ROBDD complexity

Theorem 14 *The maximal complexity $C_{\text{ROBDD}}(n)$ of the ROBDD of a boolean function in n variables is:*

$$C_{\text{ROBDD}}(n) = 2^{h(n)} + 2^{2^{n-h(n)}} - 1$$

Proof: We first exhibit a hard function f , in n variables, whose $c_{\text{ROBDD}}(f) = 2^{h(n)} + 2^{2^{n-h(n)}} - 1$.

Let j , $h(n) \leq j < n$ (the lower part). For any hard function, among the $R_n(j)$ vertices of its boolean graph, $R_n(j+1)$ of them must be deleted to construct the ROBDD because they are connected to their descendants by double edges. It then remains $\sum_{i=h(n)}^n R_n(i) - \sum_{i=h(n)+1}^n R_n(i) = R_n(h(n)) = 2^{2^{n-h(n)}}$ vertices in the lower part.

In the upper part, there is no reduction to wait for, because the reduced graph upper part is a binary tree.

In the inflexion zone, some suppressions of vertices of height $h(n) - 1$ may occur. Obviously, the ROBDD will be less complex as there is more double edges in this zone. But we can always choose f such that no double edge is in the inflexion zone, because $k \leq m^2/2$ by definition of inflexion height, and:

$$m^2 - m \geq \frac{m^2}{2} \text{ for } m \geq 2$$

Then, for any n there exists an hard f whose ROBDD complexity verifies the equation of the theorem.

To finish the proof, we have to show that:

$$\max_{f \in \mathcal{B}_n} c_{\text{ROBDD}}(f) \leq \max_{f \in \mathcal{H}_n} c_{\text{ROBDD}}(f)$$

For any boolean function g we can easily construct its reduced graph from its ROBDD by reintroducing removed vertices and edges. This process is unique by canonicity of reduced graph and ROBDD. For i , $0 \leq i \leq n$, let $d_i(g)$ be the number of vertices of height i of the reduced graph of g which are origins of double edge then:

$$c_{\text{ROBDD}}(g) = \sum_{i=0}^n r_i(g) - d_i(g)$$

Then we have:

$$r_i(g) - d_i(g) \leq r_{i+1}(g)^2 - r_{i+1}(g) \leq r_{i+1}(f)^2 - r_{i+1}(f)$$

The first inequality comes from the definition of a ROBDD and the second from the definition of hard functions. Since f is hard, $r_{i+1}(f)^2 = r_i(f)$ and $r_{i+1}(f) = d_i(f)$, then:

$$r_i(g) - d_i(g) \leq r_i(f) - d_i(f)$$

So:

$$c_{\text{ROBDD}}(g) \leq c_{\text{ROBDD}}(f)$$

□

Calling ROBDD-hard functions the boolean functions whose ROBDD is of maximal size, we see that ROBDD-hard functions are not always hard functions but there is always a hard function which is ROBDD-hard.

8 Values of n and Storage Access functions

Theorem 15 *The number of hard functions in $a + 2^a$ variables is 2^{2^a} !*

Proof: From theorem 3, we have $h(n) = 2^a$, $m = 2^{2^a}$ and $k = 2^{2^a-1}$ then the inflexion pair is $(m, \frac{m}{2})$.

From the construction of boolean graph we know that there is $\frac{m}{2}$ pairs of m vertices to construct such that each vertex appear in at least one pair. The only way to obtain this is to permute the ordered set of the m vertices and then pair the first vertex with the second, the third with the fourth and so on. There is $m!$ such permutations. □

Theorem 16 *For $n = a + 2^a + b$, with $0 \leq b \leq 2^a$, $\text{Card}(\mathcal{H}_n) \leq 2^{2^a+b}$!*

Proof: Permuting the multiset made of $2k/m$ copies of the m vertices, and pairing those two by two, gives us at most $2k! = 2^{2^a+b}$ possibilities. □

Then from Stirling formula we deduce that:

Corollary 4 *The asymptotic density of \mathcal{H}_{a+2^a} , is 0 when $a \rightarrow \infty$*

A boolean function is **balanced** when it takes the value 0 as often as it takes the value 1. It's easy to show that this property is independant from the ordering of the variables.

Theorem 17 *Any hard function of $a + 2^a$ variables is balanced.*

Proof: Every boolean function in a variables is rooted once and only once at level $h(n)$ (2^{2^a} such functions and $m = 2^{2^a}$ vertices). So if a function f is rooted then $\neg f$ is rooted too. □

The preceding theorem is false for non special values of n , but:

Theorem 18 *For any n , there exists an hard balanced function in n variables.*

Proof: Going back to the construction of a boolean graph of a function in $n = a + 2^a + b$ (with $0 \leq b \leq 2^a$) variables, we can always take a graph in which a balanced set (if f is in the set so is $\neg f$) of boolean function in a variables is rooted at level $h(n)$. To ensure the balancing of the whole function it is sufficient to construct the basic following $(m, m/2)$ -correct configuration:

$$\forall i \in [1, \frac{m}{2}], (\frac{m}{2} + i, i) \text{ are checked}$$

Any extension of that configuration which satisfy the following properties:

1. $\forall i \in [1, \frac{m}{2}]$ and $j \in [\frac{m}{2}, m]$, (i, j) and (j, i) are both checked or not
2. $\forall i, j \in [1, \frac{m}{2}]$, (i, j) and $(\frac{m}{2} + i, \frac{m}{2} + j)$ are both checked or not

is also correct. Now the set of boolean functions rooted at level $h(n)$ is partitioned such that if a function f has index $i \leq \frac{m}{2}$ then $\neg f$ has index $\frac{m}{2} + i$. Then the configuration balance the whole function. \square

8.1 The Storage Access functions SA_k

Storage access functions are well known in boolean complexity theory. They are fundamental in hardware design where they are called **multiplexers**. Let $k = 2^a$, the SA_k function is a function in $n = a + 2^a$ variables defined by:

$$SA_k(x_0, \dots, x_{2^a-1}, y_0, \dots, y_{a-1}) = x_m$$

where m is the integer whose development in base 2 is $y_0 \dots y_{a-1}$.

Theorem 19 *The SA_k functions are hard.*

Proof: This results directly from the proof of the preceding theorem. The truth table of SA_k consists in successive 2^k blocks of k bits. Each of these blocks, from left to right, is the binary development of the successive integers from 0 to $2^k - 1$. For example, the truth table of SA_4 is the 64 bits string coded 0123456789ABCDEF, in hexadecimal. \square

Then the SA_k functions arise in our theory as the “simplest” hard functions and the truth table of all the different hard functions are obtained as the $2^k!$ permutations of all the blocks of k bits.

Let $N = 2^k$ and $\Sigma \in \mathfrak{S}_N$. Σ induces naturally a bijection of the set of all k -uples of bits representing all the integers between 0 and $N - 1$.

Definition 8 *We call **twisted (by Σ) storage access function** :*

$$SA_k^\Sigma(x_0, \dots, x_{k-1}, y_0, \dots, y_{a-1}) = SA_k(\Sigma(X), y_0, \dots, y_{a-1})$$

where X is the integer whose binary development is $x_0 \dots x_{k-1}$.

The index k will be omitted when the context is clear. Of course, if $\sigma = Id$ then $SA_k^\Sigma = SA_k$.

From the structure of the reduced graph we deduce at once:

Theorem 20 \mathcal{H}_{a+2^a} is the set of all $SA_{2^a}^\Sigma$.

Theorem 21 Let $n = a + 2^a + b$ (where $0 < b \leq 2^a$), \mathcal{H}_n is the set of all well projected $SA_{2^{a+1}}^\Sigma$.

Proof: One can see that vertices at level $h(n) - 1$ defines $2^{h(n)-1}$ different boolean functions of $a + 1$ variables. \square

In other words, we have shown that every hard function is the composition of a SA , a permutation of \mathfrak{S}_N and possibly a projection.

9 Directions for future investigations

The form of enumeration formulas obtained on \mathcal{H}_n suggests that there are generating series behind the scene which would be interesting to explicit and understand.

The connection with the theory of modular (characteristic 2) representations of the symmetric group, and the complexity of bijection seems also of interest. In particular can we describe, or bound, the variation of complexity on the orbit of a boolean function under permutations of its variables? Note that, starting from $c(f)$, we can easily define an “invariant” (for variables permutations) measure of complexity for boolean functions:

$$\bar{c}(f) = \inf_{\sigma \in \mathfrak{S}_n} c(f^\sigma)$$

What are the \bar{c} -hardest functions of \mathcal{B}_n ?

We deal here with a model of computation called QROBDD but there is a lot of other species of BDD. For example the truth tree of a boolean function can be replaced by the tree of quotients and rests of the successive euclidean division of the boolean function by its successive variables. The process of reduction is unchanged: identify isomorphic subtrees. Then we can develop the same arguments for studying the associated complexity measure and discover another specie of “hard” functions.

Another interesting subject for cryptographers is the relation of hard functions with balanced Bent functions. We leave those subjects untouched here.

Appendix A: Complexity repartitions for $n \leq 4$

We just give the number of boolean functions of given complexity $c(f)$ for small values of n

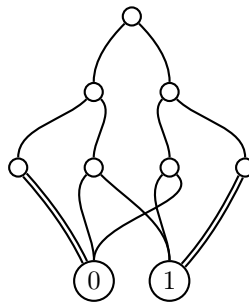
n	c(f)	number of b. f.
1	2	2
	3	2
2	3	2
	4	2
	5	12
3	4	2
	5	2
	6	12
	7	72
	8	144
	9	24

n	c(f)	number of b. f.
4	5	2
	6	2
	7	12
	8	72
	9	576
	10	1752
	11	10656
	12	20736
	13	31728

Appendix B: Characteristic datas of hard functions for small n

Example : Reduced graph of the hard function in three variables (complexity 9)

$$f(x, y, z) = x + yz + xz.$$



References

- [Bry86] R.E. Bryant, Graph Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers Vol C-35, 8, August 1986, p.677-691.
- [CP89] J.M. Champarnaud, J.E. Pin, A Maxmin problem on finite automata, Discrete Applied Mathematics, 23, 1989, 91-96.
- [Grö99] C. Gröpl: Binary Decision Diagrams for Random Boolean Functions. phd thesis, Humboldt-Universität zu Berlin, 1999.
- [GPS98] C. Gröpl, H.J. Prömel and A. Srivastav: Size and Structure of Random Ordered Binary Decision Diagrams (Extended Abstract). In: Daniel KroB, Christoph Meinel and Michel Morvan (editor): STACS 98, number 1373 in Lecture Notes in Computer Science, pages 238-248, Berlin, Heidelberg, New York, 1998. Springer Verlag.

- [GPS01] C. Gröpl, H.J. Prömel and A. Srivastav: On the Evolution of the Worst-Case OBDD Size, *Information Processing Letters*, 77: 1-7, 2001.
- [JK81] G. James, A. Kerber, The representation theory of the symmetric group, *Encyclopedia of mathematics and its applications* vol. 16, Cambridge University Press, 1981.
- [Kra02] M. Krause, BDD-based Cryptanalysis of keystream generators, *Advances in cryptology - Eurocrypt 2002*, LNCS 2332, pages 222-237, Springer, 2002.
- [LL92] H.T. Liaw and C.S. Lin, On the OBDD-representation of General Boolean Functions, *IEEE Transactions on Computers* vol 41, 7, pp 661-664, July 1992.
- [Mas96] J. Massey, The difficulty with difficulty. *Eurocrypt 96 IACR Distinguished Lecture*.
- [Pau77] Paul, A $2.5n$ lower bound on the combinatorial complexity of Boolean functions *SIAM J. on Comp.* 6, 427-443, 1977.
- [Sha49] Shannon, The synthesis of two-terminal switching circuits. *Bell Syst. Techn. J.* 28, 59-98.
- [SB00] S.A. Seshia, R.E. Bryant, The Hardness of Approximating Minima in OBDD, FBDD and Boolean Functions, CMU-CS-00-156, August 2000.
- [Spi97] M. Spiser, *Introduction to the theory of computation*, PWS Publishing Company, 1997. ISBN 0-534-94728-X
- [Weg87] I. Wegener, *The complexity of Boolean functions*. Wiley 1987. ISBN 0 471 91555 6.
- [Weg00] I. Wegener, *Branching programs and binary decision diagrams*, *SIAM Monographs on Discrete Mathematics and Applications* 4, 2000. ISBN 0-89871-458-3.