RAIRO-Inf. Theor. Appl. **39** (2005) 677-686
DOI: 10.1051/ita:2005036

ON MAXIMAL QROBDD'S OF BOOLEAN FUNCTIONS*

Jean-Francis Michon[1], Jean-Baptiste Yunès[2] and Pierre Valarcher[1]

Abstract. We investigate the structure of "worst-case" quasi reduced ordered decision diagrams 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 QROBDD are "worst-case" ones. So we exhibit the relation between hard functions and the Storage Access function (also known as Multiplexer).

Mathematics Subject Classification. 06E30, 68Q15, 94C10, 94C15.

1. Introduction

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 representations for Boolean functions like: truth tables, Boolean circuits, binary decision diagrams, normal disjunctive and conjonctive forms, etc. We focus here on Quasi Reduced Ordered Binary Decision Diagrams.

The binary decision diagram (BDD) representation was introduced by Lee in 1959 but the rise of its success began with Bryant's thesis [1] 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

Keywords and phrases. Boolean functions, Boolean complexity, Boolean graphs, binary decision diagrams, BDD, OBDD.

* *Work partially supported by "ACI Cryptologie – Ministère de la Recherche (France)".*

[1] Université de Rouen, LIFAR, 75821 Mont Saint Aignan Cedex, France;
jean.francis.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

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 [9] for exhaustive study.

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

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 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 $C(n)$, can be effectively computed and was studied by many people until recently (see [2–4]).

Our goal is the description of the family $\mathcal{H}_n$ of Boolean functions in n variables achieving this maximal QROBBD-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. Functions in $\mathcal{H}_n$ are all related to the Storage Access (SA) function (see [7, 8]). 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.

2. Canonical reduced graphs of Boolean functions

Definition 2.1. A Boolean function (in n variables, $n \geq 0$) is any:

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

We call their set $\mathcal{B}_n$: it has 2^{2^n} elements.

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 $f(x_0, ..., x_{n-1})$, if $n \geq 1$. Then the order $x_i \prec x_{i+1}$ will be implicitly used in the rest of the paper.

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

Definition 2.2. 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 leaves 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 vertex to the root is called the **height** of the vertex. 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 a *unique* graph whose core structure is a binary tree and in which the label of a leaf is exactly the value of the function at the point corresponding to the unique n-uple (respecting the previously defined *natural* order of the variables) labelling the path from the root to that leaf. This graph is exactly the **truth tree** of the function and its size obviously is $2^{n+1} - 1$.

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 leaves. A (resp. strict) reduction is a morphism from G to G' with $size(G') \leq size(G)$ (resp. $size(G') < size(G)$). We say that G' is a reduction of G: one can think of a reduction as identification of isomorphic subgraphs. It is known (see [1] and [9]) that the composition of reductions is a reduction and that operation is confluent. A Boolean graph is called **irreducible** if no strict reduction can be defined on it. By confluence, one can even say more: for any Boolean graph G there exists a unique irreducible Boolean graph which is a reduction of G.

Definition 2.3. The reduced Boolean graph of a Boolean function f is the irreducible Boolean graph associated to its truth tree. We call it the **QROBDD** of f. Its size is called the **(binary) complexity** of f and written $c(f)$.

It's clear that we can recover the truth tree of f from its QROBDD. So QROBDD 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.

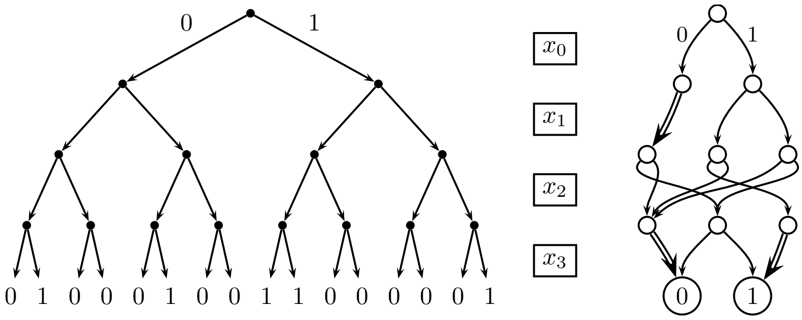

Figure 1. A truth tree and its QROBDD.

Then we easily deduce:

Proposition 2.4. *Let $r_i(f)$ be the number of vertices at level i ($0 \le i \le n$) of the reduced Boolean graph of f, then:*

1. $r_0(f) = 1$ *and* $r_n(f) = 1$ *or* 2*;*
2. $r_{i+1}(f) \le 2r_i(f)$*;*
3. $r_i(f) \le r_{i+1}(f)^2$*;*
4. $c(f) = \sum_{i=0}^{n} r_i(f)$.

We can then deduce that $r_i \le 2^i$ and $r_i \le 2^{2^{n-i}}$, which leads to the well known property (see [2] or [5]):

Proposition 2.5. $\forall i \in [0, n]$, $r_i(f) \le \inf(2^i, 2^{2^{n-i}})$.

Figure 1 is a picture of a truth tree and its associated QROBDD of a Boolean function in 4 variables (x_0, x_1, x_2, x_3).

3. Connecting consecutive levels

Counting the number of different ways to connect k vertices at level i to m vertices at level $i + 1$ according to Definition 2.2 can be restated in the more manageable following combinatorial problem:

Problem 3.1. Consider a grid of $m \times m$ checkable boxes. In how many ways can we check off k different boxes so that, for any s so that $1 \le s \le m$, a row or a column of index s has at least a checked box?

We call any solution to that problem an (m, k)**-correct configuration**. For example,

×	×	
	×	

is $(3, 3)$-correct but

×	×	
	×	

is not.

Theorem 3.2. *For all $k, m \in \mathbb{N}$, let $C(m,k)$ be the number of (m,k)-correct configurations. It satisfies:*

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

$$C(m,k) = \binom{m^2}{k} - \sum_{j=1}^{m} \binom{m}{j} C(m-j,k) > 0.$$

Proof. If $\frac{m}{2} \leq k \leq m^2$ then there exists at least one configuration, this proves the inequality.

Then consider all possible k-checked configurations in the $m \times m$ grid and subtract all incorrect configurations. First subtract those missing exactly one index: the ones that 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. This leads us to the equality. □

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

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

Theorem 3.3. *For all $m, k \in \mathbb{N}$, $C(m,k)$ satisfies:*

(1) $C(0,0) = 1$, $\forall\, p > 0$, $C(0,p) = C(p,0) = 0$;
(2) $C(1,1) = 1$, $\forall\, p > 1$, $C(1,p) = 0$;
(3) $\forall\, m > 1, \forall\, k > 0,$ *then*

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

Proof. We first multiply the two sides by $(k-1)!$ and get in the left handside the number of correct configurations of k ordered 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 ordered 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 orderely 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 $m(m-1)(k-1)!C(m-2,k-1)$. □

Explicit values of $C(m,k)$ can be found in [6].

4. Hard Boolean functions

Definition 4.1. 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.

We define the height of inflexion (the "critical point" in [2]) the unique integer $h(n) < n$ so that:

$$2^{h(n)-1} < 2^{2^{n-(h(n)-1)}} \text{ and } 2^{h(n)} \geq 2^{2^{n-h(n)}}.$$

We define $h(0) = 0$ and $h(1) = 1$.

It is known (see [2]) that:

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

Definition 4.2. For all integer $n \geq 1$ we call $(r_{h(n)}(f), r_{h(n)-1}(f))$ the **inflexion pair** of a n variables hard Boolean function f.

When n varies, the inflexion pair evolves regularly:

Theorem 4.3. *Let (m,k) be the inflexion pair of n, then, when n takes on all values between $a+2^a$ and $a+2^{a+1}$, i.e. $n = a+2^a+b$ with $0 \leq b \leq 2^a$ ($a,b \in \mathbb{N}$), we have:*

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

We can now conclude from these results:

Proposition 4.4. $\mathrm{Card}(\mathcal{H}_n) = k!C(m,k)$ *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. □

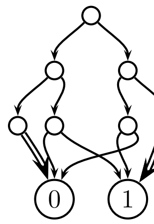

Figure 2. A QROBDD of an hard Boolean function.

Figure 2 is an example of a reduced Boolean graph of a hard Boolean function in 3 variables, $f(x_0, x_1, x_2) = x_0 + x_0x_2 + x_1x_2$ (symbol + denote the addition modulo 2).

5. Some properties of hard functions and $\mathcal{H}_n$

Proposition 5.1. *A hard function depends on its n variables except when n is of the form* $1 + a + 2^a$. *In this case, there exist hard functions in n variables which essentially depend on only* $n - 1$ *variables. Their form is:*

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

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 on the variable $x_{h(n)-1}$.

Conversely, if the function does not depend on x_j, 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 $j = h(n) - 1$. □

Corollary 5.2. *There are* $2^a!$ *hard functions in* $n = 1 + a + 2^a$ *variables which depends in only* $n - 1$ *variables.*

For example and using a polynomial representation of the Boolean functions (symbol + denote the xor operation):

(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 5.3. *The number of hard functions in* $a + 2^a$ *variables is* $2^{2^a}!$

Proof. From theorem 4.3, 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 QROBDD we know that there are $\frac{m}{2}$ pairs of m vertices to construct so 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. □

Then from Stirling formula we deduce that:

Corollary 5.4. *The asymptotic density of $\mathcal{H}_{a+2^a}$ in $\mathcal{B}_{a+2^a}$, is 0 when $a \to \infty$.*

Proposition 5.5. *For $n = a + 2^a + b$, with $0 \le b \le 2^a$, $Card(\mathcal{H}_n) \le 2^{2^a+b}!$.*

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. □

6. Storage access functions SA_k

Storage Access functions are well known in Boolean complexity theory (see [8]). 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 radix 2 is $y_0 \dots y_{a-1}$.

Proposition 6.1. *The SA_k functions are hard.*

Proof. 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. □

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 blocks of k bits.

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

Definition 6.2. We call **Σ-twisted 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 $SA_k^{\Sigma} = SA_k$.

Then, from the structure of the reduced graph we deduce at once:

Theorem 6.3. *$\mathcal{H}_{a+2^a}$ is the set of all $SA_{2^a}^{\Sigma}$.*

For non special values of n (*i.e.* $a + 2^a < n < (a+1) + 2^{a+1}$), one can observe that any hard Boolean function is built so that:

Theorem 6.4. *A Boolean function f is in $\mathcal{H}_n$ if and only if it satisfies the following two properties:*

(1) *there is an injection $\phi : \{0,1\}^{2^a} \to \{0,1\}^{2^a+b}$ so that:*

$$f(\phi(x_0, \ldots, x_{2^a-1}), x_{h(n)}, \ldots, x_{n-1}) = SA_{2^a}(x_0, \ldots, x_{2^a-1}, x_{h(n)}, \ldots, x_{n-1});$$

(2) *there is an injection $\Phi : \{0,1\}^{2^a+b-1} \to \{0,1\}^{2^{a+1}}$ so that:*

$$f(x_0, \ldots, x_{n-1}) = SA_{2^{a+1}}(\Phi(x_0, \ldots, x_{h(n)-2}), x_{h(n)-1}, \ldots, x_{n-1}).$$

Proof. Let $f \in \mathcal{H}_n$ and G its QROBDD. We suppose, for convenience, that the vertices in the lower part of G are ordered by the "natural" order of the truth tables. Then, selecting a path $u = u_0 \cdots u_{2^a-1}$ in the graph of SA_{2^a} which lead to a Boolean function f_u, ϕ is easy to define: we associate to u the path $v = \phi(u) = v_0 \ldots v_{2^a+b-1}$ in G which leads to f_u.

To define Φ, one can observe that level $h(n) - 1$ is the root of many (exactly $2^{h(n)-1}$) different Boolean functions in $a+1$ variables but possibly not all of them. There is only one path from the root to each of them (the upper part is a binary tree). Then for each path $u = u_0 \ldots u_{h(n)-2}$ in G which defines f_u, we can associate a path $v = \Phi(u) = v_0 \ldots v_{2^{a+1}-1}$ in $SA_{2^{a+1}}$ which leads to f_u.

Conversely, the first property implies that level $h(n)$ contains 2^{2^a} distinct vertices and the second that level $h(n)-1$ is connected to the root using a binary tree so it contains $2^{h(n)-1}$ vertices. Then f is hard. □

7. Directions for future investigations

We shall explore on forthcoming papers the generating series behind the scene which are interesting to explicit and understand.

It would also be interesting to know the density of $\mathcal{H}_n$ in $\mathcal{B}_n$ (for any n).

Up to now, we based our investigations on truth tables, but different semantics (OFDD for example, see [9]) may lead to different sets of "hard" functions, studying them will certainly be interesting.

References

[1] R.E. Bryant, Graph based algorithms for boolean function manipulation. *IEEE Trans. Comput.* **C-35** (1986) 677–691.

[2] C. Gröpl, *Binary Decision Diagrams for Random Boolean Functions*. Ph.D. Thesis, Humboldt-Universität zu Berlin (1999).

[3] C. Gröpl, H.J. Prömel and A. Srivastav, Size and structure of random ordered binary decision diagrams (extended abstract), in *STACS'98*, Springer Verlag. *Lect. Notes Comput. Sci.* **1373** (1998) 238–248.

[4] C. Gröpl, H.J. Prömel and A. Srivastav, On the evolution of the worst-case obdd size. *Inform. Process. Lett.* **77** (2001) 1–7.
[5] M. Heap and M.R. Mercer, Least upper bounds on obdd sizes. *IEEE Trans. Comput.* **43** (1994) 764–767.
[6] J.F. Michon, P. Valarcher and J.B. Yunès, Integer sequence number `a100344`. stored in *The On-Line Encyclopedia of Integer Sequence*, N.J.A. Sloane, published electronically at `www.research.att.com/~njas/sequences` (2004).
[7] W. Paul, A $2.5n$ lower bound on the combinatorial complexity of boolean functions. *SIAM J. Comput.* **6** (1977) 427–443.
[8] I. Wegener, *The complexity of Boolean functions*. Wiley (1987).
[9] I. Wegener, Branching programs and binary decision diagrams. *SIAM Monogr. Discrete Math. Appl.* (2000).

Communicated by C. Choffrut.
Received March, 2003. Accepted December, 2004.