String-matching with OBDDs

Ch. Choffrut*, Y. Haddad
LIAFA, UMR 7089, Université Paris 7
2 Pl. Jussieu, Paris Cedex 75251
France

November 3, 2003

Keywords: Ordered binary decision diagrams, finite automata, string-matching problem.

Abstract

Ordered binary decision diagrams, in short OBDDs, are a very popular graph representation for Boolean functions. They can be viewed as finite automata recognizing sets of strings of a fixed length, where the letters of the input strings are read at most once in a predefined ordering. The string matching problem with string w as pattern, consists of determining, given an input string, whether or not it contains w as substring. We show that for a fraction of orderings tending to 1 when n increases arbitrarily, the minimal size of an OBDD solving the string matching problem for strings of length n has a growth which is an exponential in n.

1 Introduction

The binary decision diagrams or "ite straight line programs", also known as branching programs on a set of variables, were introduced in the late eighties as a model for computing Boolean functions, [7]. They can be traced back to the works of C. Y. Lee, [13]. The ordered binary decision diagrams, abbreviated as OBDDs, which are the subject of the present study, differ in that the Boolean variables are queried in a predetermined order with the possibility of skipping some variables. They are widely used for implementing numerous functions such as those arising in digital circuit design, signal processing and arithmetic operations. We refer to Wegener's handbook, [18], for a comprehensive presentation of the various aspects of the topic. The reader will find in particular a comparative study of the performance of the three models of binary decision diagrams, circuits and ordered binary

*email: `choffrut@liafa.jussieu.fr`

decision diagrams relative to classical issues such as satisfiability, evaluation, minimization and synthesis of one Boolean function, equivalence and operations of two Boolean functions, etc.

One of the main issues of OBDDs is the choice of an ordering of the variables minimizing the size of the diagram, which was proven to be NP-complete, [4]. Most of the literature is concerned, as with circuit theory, not with single Boolean functions, but rather with sequences of functions $(f_n)_{n>0}$ where n stands for the number of variables: arithmetic operations of two n-bit operands, different types of memory access functions on n bits, etc. Wegener studied the asymptotic sensitivity of the growth of the OBDD size relative to the choice of the variable ordering, and proposed a classification of the Booelan functions into various families: *nice*, *almost nice*, *ugly*, *very ugly* and *almost ugly*. E.g., nice functions are those for which all variable orderings lead to polynomial OBDD size, while on the opposite, very ugly functions are those for which all variable orderings lead to exponential OBDD size. Almost ugly functions, which are the purpose of this work, are between these two extremes, in the sense that there is a way of choosing an ordering for which the growth is polynomial, but almost surely, an arbitrary choice of the variables leads to a nonpolynomial growth.

Our main result makes use of this typology and applies it to the area of formal languages. Circuit complexity contributed to a better understanding of the regular languages, see [16], but to our knowledge, this has not yet been the case for OBDDs. At this stage of the exposition we cannot be extremely precise and so we resort to the more familiar notion of finite automaton. Indeed, an OBDD may be viewed as a finite automaton recognizing words of a fixed length n, where each letter of an input string is read at most once, not necessarily sequentially from left to right, but in a fixed arbitrary ordering given by a permutation of the set $\{1, 2, \ldots, n\}$. As for automata, there exists a notion of minimal OBDD which however, clearly depends on the ordering of reading the letters. Assuming a binary alphabet is given, the *string matching problem* is given by a fixed string, called the pattern. The charactacteristic function which assigns the value 1 to every input string having an occurrence of the pattern and 0 otherwise can be viewed as a sequence of Boolean functions indexed by the length n of the input. We show that under the uniform distribution hypothesis over orderings, with probability 1 when n tends to infinity, the size of the minimal OBDD relative to a random ordering of the first n integers, grows at least as fast as α^n for some $\alpha > 1$.

2 Preliminaries

We recall the main basic notions to make our work as self-contained as possible. We encourage the reader to consult Wegener's book for a more thorough presentation and possibly missing notions.

We are interested in Boolean functions $f : \{0,1\}^n \to \{0,1\}$ where $n \in \mathbb{N}$ is the *arity* of the function. Given a subset of indices $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ and Boolean values $a_{i_1}, \ldots, a_{i_k}$, we denote by $f_{|x_{i_1}=a_{i_1} \ldots, x_{i_k}=a_{i_k}} : \{0,1\}^{n-k} \to \{0,1\}$ the *restriction* of f to the variables $X - \{x_{i_1}, \ldots, x_{i_k}\}$ obtained by fixing each x_{i_j} to the value a_{i_j}, for $1 \leq j \leq k$.

We assume the Boolean variables are taken from an infinite set $X = \{x_1, x_2, \ldots\}$. For a fixed n, an *ordering* of the variables $x_1, x_2, \ldots, x_n$ is a permutation π on the set $\{1, 2, \ldots, n\}$, i.e., an element of the symmetric group $\mathfrak{S}_n$. A *π-ordered binary decision diagram*, or π-OBDD, over a set of n Boolean variables $x_1, \ldots, x_n$, is a pair consisting of a permutation $\pi \in \mathfrak{S}_n$ and a directed acyclic graph. This graph has two nodes of outdegree 0, called the *sinks* and a specific node with indegree 0 called the *source*. All other nodes are *internal nodes* and have in-degree different from 0 and out-degree equal to 2. The nodes are labelled by one of the n variables except the two sinks which are labeled by the Boolean constants 0 and 1. The edges of the graph are labeled by the two Boolean values 0 and 1 and are called the 0- and the 1-edges respectively. Furthermore it is assumed that along a path from the source to one of the sinks, the variables that are visited, are visited in an order which is compatible with the permuation π, i.e., that there exists an increasing sequence $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ for which the nodes visited along the path are $x_{\pi(i_1)}, x_{\pi(i_2)}, \ldots, x_{\pi(i_k)}$. In other words, in all paths, the variables are visited in the same order, where some variables may possibly be skipped. The Boolean function f associated with the OBDD is now explained. Given an assignment for the Boolean variables, start up from the source and follow the unique path by taking for each node labeled by, say x_i, the outgoing edge labeled by the value of the variable in the assignment. If the path ends up in the sink 0, then f takes on value 0, else value 1.

Example 1 Consider the function $f(x_1, x_2, x_3, x_4)$ with value 1 if and only if for some integer $1 \leq i < 4$ the equalities $x_i = 0$ and $x_{i+1} = 1$ hold. Considering the permutation π reduced to the cycle (1243), consists of querying the variables in the order x_2, x_4, x_1 and x_3 and leads to the following π-OBDD where the dotted edges are labeled by 0 and the straight edges are labeled by 1.

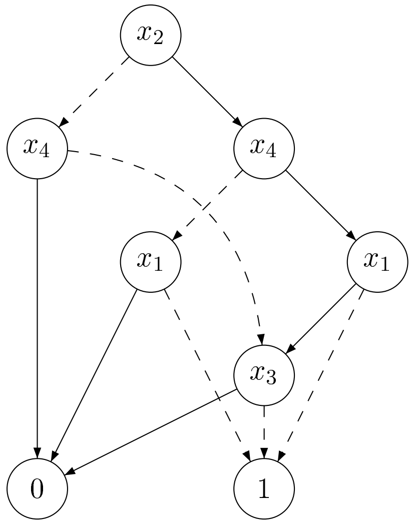

Figure 1: An OBDD with the permuation $\pi(1) = 2$, $\pi(2) = 4$, $\pi(3) = 1$, $\pi(4) = 3$

2.1 Reduced OBDD

Given a fixed permutation π over $\{1, \ldots, n\}$ and a Boolean function $f : \{0,1\}^n \to \{0,1\}$, it is possible to assign it a π-OBDD with a minimal number of nodes which is unique up to isomorphism, known as its *reduced* π-OBDD, whose *size* (i.e., its number of nodes), is denoted by π-OBDD(f).

Definition 1 *[6] A π-OBDD is* reduced *if the following conditions are satisfied.*

1) *if u and v are nodes labeled by the same variable and if the* 0*-outgoing (resp.* 1*-outgoing) edge leads to the same node, then $u = v$.*

2) *the two outgoing edges from a given node lead to two different nodes.*

The reduced OBDD equivalent to a given OBDD can be computed in linear time, [7] and [15], via an algorithm which is an elaboration of the test of isomorphism of two labeled trees, [1, Theorem 3.3.]. The reader may verify that the OBDD of example 1, is reduced.

There is another structural characterization of reduced OBDDs based on a concept akin to that of right contexts for finite automata, [15].

2.2 Sensitivity to the variable ordering

As a general rule, given a Boolean function, the size of its reduced OBDD depends on the chosen ordering of the variables. The sensitivity of a Boolean function is the ratio between the size of the smallest and the size of the greatest reduced OBDD when the orderings run over all possible permutations. For a random function this ratio is very close to 1, [17].

However, the focus here is different. We adopt the usual convention for which the term "Boolean function" refers actually to an infinite family (f_n) of Boolean functions depending on n variables and we study the asymptotic behaviour of the reduced OBDD associated with f_n, in the light of concepts defined in [18]. Its is easy to figure out functions for which the size of the reduced OBDD has a polynomial or to the contrary an exponential growth depending on the choice of the ordering. Wegener defines five different classes of asymptotic behaviours. One of them assumes two conditions. The first one says that there exist orderings for which the size of the reduced OBDD grows as slowly as some polynomial. The second one assumes that there exists a fraction of orderings π tending to 1 when n tends to infinity, for which the size of the π-reduced OBDD has exponential growth. A more formal definition is as follows.

Definition 2 *A function $f = (f_n)$ is* almost ugly *if the following two conditions are satisfied*

i) *There exists an integer k such that for all integers n there exists a permutation π_n for which the size π_n-OBDD(f_n) is less than n^k.*

ii) *There exist a real number $\alpha > 1$ and for each integer n there exists a fraction of permutations over n tending to 1, such that for each such permutation, say π, the size π-OBDD(f_n) is greater than α^n.*

The most significant bit of the sum of two binary integers and the comparison of two binary integers are examples of almost ugly functions, [18, Chapter 5].

3 OBDD and regular languages

3.1 Finite automata

A possible application of OBDDs is encoding finite, not necessarily deterministic, automata via the set of their transitions, e.g., [14]. Indeed, given a finite set of states Q and a finite alphabet Σ with $\mathrm{Card}(Q) \leq 2^k$ and $\mathrm{Card}(\Sigma) \leq 2^n$ respectively for some integers k and n, any set of transitions can be viewed as Boolean function of $\{0,1\}^{(2k+n)}$ into $\{0,1\}$ and therefore efficiently represented by an OBDD.

Our approach is completely different since we are interested in the languages accepted by finite automata, in the spirit of the circuit complexity of regular languages considered, for example, in Straubing's textbook, [16]. The general framework is the following. The free monoid generated by the alphabet $\Sigma = \{0, 1\}$ is denoted by $\{0, 1\}^*$. The terminology of regular expressions is freely used. For example, the expression $00\Sigma^*1$ represents the set of strings starting with two 0's and ending with a 1. Also the set of strings of length n is represented by $\{0, 1\}^n$. Given an arbitrary subset $L \subseteq \{0, 1\}^*$ and an integer n, we define the function $f_n : \{0, 1\}^n \to \{0, 1\}$ by setting

$$f_n(a) = \begin{cases} 1 \text{ if } a \in L \cap \{0,1\}^n \\ 0 \text{ if } a \in \{0,1\}^n - L \end{cases} \tag{1}$$

It is the *characteristic function* of L for the strings of length n and can be viewed as a Boolean function, once the string $a = a_1 \cdots a_n$ is identified with the n-tuple of Boolean values $(a_1, \ldots, a_n)$. From now on, we will not distinguish between binary strings of length n and n-tuples of Boolean values of Boolean variables. If there exists a finite automaton recognizing L, this automaton processes the input sequentially. Furthermore, for a fixed integer n, the minimal finite automaton recognizing the strings of L of length n has linear size in n and therefore so has the reduced OBDD for the identity ordering, thus satisfying the first condition of Definition 2. Observe that, more generally and independently of being or not recognized by a finite automaton, if the growth function of a subset L, i.e., the number of strings of length n in L, is polynomial, then the reduced OBDD has polynomial growth relative to n, whatever the ordering of the variables. We will thus need to concentrate on the second condition of Definition 2.

3.2 String matching

The regular languages we are concerned with, are related to the *string-matching* problem which can be stated as follows where $\Sigma = \{0, 1\}$

STRING-MATCHING PROBLEM

INSTANCE: a string $w \in \Sigma^k$ (the *pattern*) for some integer k.

QUESTION: given a string $x \in \Sigma^*$ (the *text*), does it contain an occurrence of the pattern, i.e., does it belong to the set $\Sigma^* w \Sigma^*$?

The string $x = x_1 x_2 \cdots x_n$ with $n \geq k$ has an *occurrence* of the pattern $w = w_1 w_2 \cdots w_k$ if and only if there exists an integer $0 < i \leq n - k$ such that $x_{i+j-1} = w_j$ holds for all $1 \leq j \leq k$. For example, the text 0010110110 contains two occurrences of the pattern 101 but no occurrence of the pattern 000. Observe that given a pattern w, the set of strings containing an occurrence of w is a regular subset of Σ^*. There is an impressive production on the topic in the literature and several textbooks are directly or indirectly devoted to it, see [11, 9]. The two historic references [12, 5] share the technique of reading the text sequentially. Galil proposed in [10] his "Boyer-Moore" machine where the input is not necessarily read sequentially but where the reader may query the values of the letters in an arbitrary order inside a sliding window, see also [3] and [2]. With OBDDs, this constraint is completely relaxed as every position can be queried. Example 1 shows the reduced OBDD which recognizes all binary strings of length 4 having the pattern 01 where the input letters are not queried sequentially.

The proof of our main result, Theorem 2, has two ingredients. The first one is concerned with a statistics on permutations. The second is related to a characteristic property of reduced OBDDs.

4 A statistics on permutations

Given an integer n meant to tend to infinity, decompose the interval $[n] = \{1, \dots, n\}$ into blocks of k consecutive elements, for a fixed integer k:

$$B_1 = \{1, \dots, k\}, B_2 = \{k+1, \dots, 2k\}, \dots, B_{\lfloor \frac{n}{k} \rfloor} = \{k(\lfloor \frac{n}{k} \rfloor - 1) + 1, \dots, n\}$$

From now on, the term block refers to any of these $\lfloor \frac{n}{k} \rfloor$ subsets. By abuse of language, the subset $\{1, \dots, \lfloor \frac{n}{2} \rfloor\}$ (respectively $\{\lfloor \frac{n}{2} + 1 \rfloor, \dots, n\}$) is called *first half interval* (respectively *second half interval*). Call *template* every subset of $\{1, \dots, k\}$. Given a nonempty collection $\mathcal{T}$ of templates and a permutation $\pi \in \mathfrak{S}_n$, we say that $\mathcal{T}$ occurs in block B_i if there exists a subset $T \in \mathcal{T}$ such that the set of variables in B_i which are queried in the first half interval are those whose indices belong to the subset $ik + T = \{ik + \ell \mid \ell \in T\}$, i.e.,

$$B_i \cap \pi(\{1, \dots, \lfloor \frac{n}{2} \rfloor\}) = k(i-1) + T$$

The subset $k(i-1) + T$ is the *occurrence* of T in block B_i.

Theorem 1 *With the previous notations, for every given $\epsilon > 0$, the probability that the fraction of blocks having an occurrence of a template in $\mathcal{T}$ for a random permutation in the uniform distribution is greater than $\frac{|\mathcal{T}|}{2^k} - \epsilon$, tends to 1 when n tends to infinity.*

Proof.

Associate with each block B different from the last one, a random variable b of value 1 if a template in $\mathcal{T}$ occurs in the block, 0 otherwise. Excluding the last block does not change the result and eliminates the technicality of having to deal with last blocks of length less than k. Let $0 < K \leq 2^k$ be the cardinality of $\mathcal{T}$. The probability of having $b = 1$ is obtained by summing up the probabilities over all templates in $\mathcal{T}$ to occur in B. Let $0 \leq r \leq k$ be the cardinality of a given template $T \in \mathcal{T}$. All possible ways of choosing a permutation so that $I \subseteq B$ be an occurrence of T in B can be obtained by a process in three steps: choose a one-to-one mapping of a subset of the first half interval onto I, then a one-to-one mapping of a subset of the second half interval onto $B-I$ and finally a one-to-one mapping of the remaining elements of the interval onto $[n]-B$. This leads to the following evaluation where $m = \lfloor \frac{n}{2} \rfloor$

$$\frac{\binom{m}{r}\binom{m}{k-r}r!(k-r)!(n-k)!}{n!} = \frac{m(m-1)...(m-r+1)m(m-1)...(m-k+r+1)}{n(n-1)...(n-k+1)}$$

The value of this quantity is $\frac{1}{2^k} + O(\frac{1}{n})$ and does not depend on r. Because $\mathcal{T}$ has K elements, the probability that some template of $\mathcal{T}$ occurs in B equals $\frac{K}{2^k} + O(\frac{1}{n})$. As there exist $\lfloor \frac{n}{k} \rfloor - 1$ blocks, the expectation of $b = b_1 + b_2 + \ldots + b_{\lfloor \frac{n}{k} \rfloor - 1}$ equals $(\lfloor \frac{n}{k} \rfloor)\frac{K}{2^k} + O(1)$, which is also the mean value of the number of blocks with a template in $\mathcal{T}$. The variance of this variable is equal to

$$E\left((b_1 + b_2 + \ldots + b_{\lfloor \frac{n}{k} \rfloor - 1})^2\right) - \left(E(b_1 + b_2 + \ldots + b_{\lfloor \frac{n}{k} \rfloor - 1})\right)^2 \qquad (2)$$

Observe that $E(b_i) = E(b_i^2)$ holds for all $i = 1, \ldots, \lfloor \frac{n}{k} \rfloor - 1$. By developping the previous formula, simplifying it and dividing by 2, we obtain

$$\sum_{i \neq j} (E(b_i b_j) - E(b_i)E(b_j)) \qquad (3)$$

In order to compute $E(b_i b_j) - E(b_i)E(b_j)$ observe first that the following holds

$$E(b_i)E(b_j) = (\frac{K}{2^k})^2 + O(\frac{1}{n})$$

Now, the term $E(b_i b_j)$ can be computed similarly as the expectation $E(b_i)$. Indeed, the probability of $b_i b_j = 1$ is the probability that both blocks have occurrences of $\mathcal{T}$, say $I \subseteq B_i$ and $J \subseteq B_j$. Let us examine under which conditions this happens. If $0 \leq r \leq k$ and $0 \leq s \leq k$ are the cardinalities of the occurrences $I \subseteq B_i$ and $J \subseteq B_j$, a routine computation leads to the formula

$$\frac{m(m-1)...(m-r+1)m(m-1)...(m-s+1)m(m-1)...(m-2k+r+s+1)}{n(n-1)...(n-2k+1)}$$
$$= \frac{1}{2^{2k}} + O(\frac{1}{n})$$

Summing up all possible choices for the subsets of $\mathcal{T}$, we obtain

$$\frac{K^2}{2^{2k}} + O(\frac{1}{n})$$

As there are $O(n^2)$ terms in formula (3), the variance is $O(n)$ and the standard deviation is $O(n^{\frac{1}{2}})$. Set $n' = \lfloor \frac{n}{k} \rfloor - 1$ and consider the random variable $b' = \frac{b}{n'}$. We get $E(b') = \frac{E(b)}{n'}$ and $\sigma^2(b') = \frac{\sigma^2(b)}{n'^2}$. By Chebyshev's theorem applied to the random variable b' we obtain

$$\Pr(|b' - E(b')| \geq \epsilon) \leq \frac{\sigma^2(b')}{\epsilon^2} = \frac{\sigma^2(b)}{n'^2\epsilon^2} = \frac{O(n)}{n'^2\epsilon^2} = \frac{O(1)}{n\epsilon^2}$$

or equivalently $\Pr(|b' - E(b')| < \epsilon) = 1 - \frac{O(1)}{n\epsilon^2}$ which completes the proof. ■

We next apply the previous result to the case of *incomplete* blocks in the following sense. For a given permutation $\pi \in \mathfrak{S}_n$, a given block B_i, where $i \neq \lfloor \frac{n}{k} \rfloor$ if $B_{\lfloor \frac{n}{k} \rfloor}$ has less than k elements, is *incomplete* if some of its elements, but not all, appear as the images of the elements of the first half interval, i.e.,

$$0 < \mathrm{Card}(B_i \cap \pi(\{1, \dots, \lfloor \frac{n}{2} \rfloor\})) < k$$

Example 2 With $[12] = \{1, 2, \dots, 12\}$ and $k = 3$, we have four blocks

$$B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}, B_3 = \{7, 8, 9\}, B_4 = \{10, 11, 12\}$$

Consider the permutation $(1\ 4\ 6\ 5\ 10)(2\ 11\ 3)(7\ 9\ 12\ 8)$. Then the only incomplete blocks are B_1 and B_4, since $B_1 \cap \pi([6]) = \{2\}$, $B_2 \cap \pi([6]) = \{4, 5, 6\}$, $B_3 \cap \pi([6]) = \emptyset$ and $B_4 \cap \pi([6]) = \{10, 11\}$.

The next corollary is immediate.

Corollary 1 *With the previous notations, for every real $\epsilon > 0$, the probability that the proportion of incomplete blocks of a random permutation is greater than or equal to $1 - \frac{1}{2^{k-1}} - \epsilon$, tends to 1 when n tends to infinity.*

5 Asymptotic behaviour

The purpose of this section is to prove our main theorem.

Theorem 2 *Let $w = w_1 w_2 \dots w_k$ be a string of length $k > 1$ over the alphabet Σ, which is different from 01 and 10. For each integer n, let f_n be the function which assigns the value 1 to the string $x = x_1 \cdots x_n$ if and only if it has an occurrence of $w_1 w_2 \dots w_k$, else the value 0. Then the function $f = (f_n)$ is almost ugly.*

Proof.

For reasons of symmetry, we may assume without loss of generality that the pattern w begins with the letter 0. The reason why the property does not hold for the particular cases 01 is clear. Indeed, the strings of length n with no occurrences of w are of the form $1^s 0^r$ with $s + r = n$. Consider the function satisfying $h(x) = 1$ if and only if $f(x) = 0$. The number of strings x of length n for which $h(x) = 1$ holds, grows polynomially with n. Then, whatever the ordering π, the size of the reduced π-OBDD grows also polynomially. The same holds for the function f since its reduced π-OBDD is obtained by simply exchanging the 0- and the 1-sinks.

The proof of the theorem proceeds in two steps. First, we define a subset $\mathcal{O}_n$ of orderings of the variables such that the ratio between the cardinality of $\mathcal{O}_n$ and that of $\mathfrak{S}_n$ tends to 1. Then we show that for all the orderings $\pi \in \mathcal{O}_n$, the size of the reduced π-OBDD is greater than α^n for some fixed real $\alpha > 1$. We start with a technical assertion.

Claim 1 *If Theorem 2 holds for the string* 1^k *with* $k > 2$*, then it holds for the string* 01^k*.*

Proof. Denote by $f_n, g_n : \Sigma^n \longrightarrow \Sigma$ the string-matching functions associated with the patterns $w = 01^k$ and $w' = 1^k$ respectively for the texts of length n. Observe that $g_n = f_n \vee h_n$ holds where h_n is the characteristic function of $1^k\Sigma^*$. Denote by G_{f_n}, G_{g_n}, G_{h_n} respectively the OBDDs of f_n, g_n and h_n. By a result of Bryant, [8], we have $|G_{g_n}| \leq |G_{f_n}|.|G_{h_n}|$ for the same ordering. Since $|G_{h_n}|$ is linear whatever the ordering, the fact that the function g is almost ugly implies that so is the function f.

∎

We now return to the proof of the theorem. In order to simplify it, we relax the second condition of a reduced π-OBDD in Definition 1 and thus use the notion of *quasi-reduced* π-OBDD. This implies in particular that in each path leading from the source to one of the sinks, no variable is skipped. Therefore, we may speak of the i-th level of the OBDD, for $1 \leq i \leq n$, which consists of all nodes which are at distance i from the source or equivalently those which are labeled by the variable x_{π_i}. It can be shown easily that the size of the quasi-reduced π-OBDD is at most n times the size of the reduced π-OBDD. It thus suffices to prove that for a fraction of orderings tending to 1, their size grows exponentially. As the alphabet is binary, it is advantageous to consider the involution $a \rightarrow \iota(a)$ exchanging 0 and 1. To the string $a_1a_2...a_p \in \Sigma^*$, this involution assigns the string $\iota(a_1)\iota(a_2)...\iota(a_p)$.

The notions of permutation, block and incomplete block of section 4 extend naturally to the set of positions in the text x with length n. The common length of the blocks, except possibly that of the last block, is equal to the length k of the pattern. Let $\mathcal{O}_n \subseteq \mathfrak{S}_n$ be the set of permutations such that in the image of the first half interval, at least half of the blocks are incomplete. By Corollary 1, the ratio of these permutations among all permutations on n elements tends to 1 with n tending to infinity. For each permutation $\pi \in \mathcal{O}_n$, choose, once and for all, a maximal number of incomplete blocks $B_1, .., B_r$ separated by at least two blocks, thus at distance at least $2k$ from one another. Clearly, $r \geq \frac{n}{6k}$ holds. We call these blocks *selected*. The remaining are the *nonselected* blocks. Decompose each selected block B, into $B^{(1)} \cup B^{(2)}$ where

$$B^{(1)} = B \cap \pi(\{1, \ldots, \lfloor \frac{n}{2} \rfloor\}) \text{ and } B^{(2)} = B \cap \pi(\{\lfloor \frac{n}{2} + 1 \rfloor, \ldots, n\})$$

In example 2, the selected blocks are B_1 and B_4 and we get

$$B_1^{(1)} = \{2\}, B_1^{(2)} = \{1,3\}, B_4^{(1)} = \{10,11\}, B_4^{(2)} = \{12\}$$

In order to establish the result, we define a set E of 2^r strings of length n of which only the positions in the range

$$\{1,\ldots,n\} \setminus (\bigcup_{1\leq i\leq r} B_i^{(2)})$$

are specified. In the running example, the set of positions which are specified is $\{2,4,5,6,7,8,9,10,11\}$. The idea of the construction is that the strings in E can only contain occurrences of the pattern w, if at all, in the selected blocks. Thus, the nonselected blocks will be filled with a unique (in one case with two) string depending on w (see the three cases below) in such a way that w cannot possibly have a nontrivial overlap with a nonselected block. Concerning the selected blocks, for the positions corresponding to the variables visited in the first half, all of them are assigned the corresponding letter of the pattern or all of them are assigned the opposite letter. These partially defined strings label paths in the quasi-reduced OBDD starting from the source. Observe that the specified letters of these 2^r strings can only differ on the positions belonging to the selected blocks. The main property is that, at level $\lfloor \frac{n}{2} \rfloor$, i.e., after having queried half of the positions, the paths reach different nodes. This will be established by proving that the partial functions naturally associated with the nodes labeled by the variable $x_{\lfloor \frac{n}{2} \rfloor}$ are different.

Now we show how the specified positions of an arbitrary string u of E are defined. Concerning the positions in $\bigcup_{1\leq i\leq r} B_i^{(1)})$ we proceed as follows. Associate with u a Boolean vector $(\epsilon_1,\ldots,\epsilon_r) \in \Sigma^r$, where $\epsilon_i = 1$ means that the variables of block B_i visited in the first half are assigned the corresponding letter of the pattern and $\epsilon_i = 0$ that they are assigned the opposite letter. E. g., if $k = 5$, if the pattern is 01101 and if the relative positions of the variables inside a selected B is $\{1,3,4\}$, then $\epsilon = 1$ means that the partially defined block is $0-10-$, where the symbol $-$ stands for an unspecified letter, whereas if $\epsilon = 0$ the partially defined block is $1-01-$. Now, consider a position $\pi(i) = \ell$ for some $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$ which belongs, say, to the selected block B. The notation $\overline{\ell}$ stands for the unique integer of the interval $\{1,\ldots,k\}$ equal to i modulo k. Then set

$$u_\ell = \begin{cases} w_{\overline{\ell}} \text{ if } \epsilon_j = 1 \\ \iota(w_{\overline{\ell}}) \text{ if } \epsilon_j = 0 \end{cases}$$

Concerning the positions which do not belong to a selected block, i.e., which are in $\{1,\ldots,n\} \setminus (\bigcup_{1\leq i\leq r} B_i)$, the letters depend on their positions and on the pattern uniquely, via one or two specific strings (a motif), but as said above, they are independent of the particular string $u \in E$. Three cases arise:

Case 1: $w \in 0\Sigma^*0$. There is only one particular motif $t = 1^k$.

Case 2: w is of the form $0^{k-s}1^s$ with $k-s, s \geq 2$ (observe that the two cases $s = 1$ and $s = k-1$ are covered by case 1, in virtue to Claim 1). Then the motif is $t = (01)^{k/2}$ if k is even and $t = 0(01)^{\frac{k-1}{2}}$ if it is odd.

Case 3: $w \in 0\Sigma^*1 - 0^2 0^* 1^2 1^*$. In that case there are two particular motifs $t = 0^k$ and $z = 1^k$.

We set $u_\ell = t_{\overline{\ell}}$ in case 1 or 2. In case 3 we set $u_\ell = t_{\overline{\ell}}$ if B is immediately to the right of a selected block and $u_\ell = z_{\overline{\ell}}$ otherwise.

Example 3 We illustrate the three cases, with $k = 4$ and template $\{1, 2, 3\}$. We show 6 possible blocks of the string, the second and fifth being selected blocks with ϵ equal to 1 and 0 respectively. We also add nonexisting bars between blocks in order to make the examples more readable.

An example for case 1 with $w = 0110$.

$$\ldots|1111|011-|1111|1111|100-|1111|\ldots$$

An example for case 2 with $w = 0011$.

$$\ldots|0101|001-|0101|0101|110-|0101|\ldots$$

An example for case 3 and $w = 0101$.

$$\ldots|1111|010-|0000|1111|101-|0000|\ldots$$

Observe that in all cases, whatever the values of the unspecified letters, the only occurrences of w may only appear inside the sepcified blocks.

Claim 2 *The nodes of the quasi-reduced* OBDD *reached by the strings in E at level $\lfloor \frac{n}{2} \rfloor$, are all different.*

Consider two nodes at level $\lfloor \frac{n}{2} \rfloor$, reached by different partial strings u and v of the set E. Then u and v are associated with two different Boolean vectors, say $(\epsilon_1, \ldots, \epsilon_r)$ and $(\mu_1, \ldots, \mu_r)$. Assume $\epsilon_i = 1$ and $\mu_i = 0$ for some $1 \leq i \leq r$. Define

$$u_\ell = v_\ell = \begin{cases} w_{\overline{\ell}} \text{ if } \ell \in B_i^{(2)} \\ \iota(w_{\overline{\ell}}) \text{ if } \ell \in B_j^{(2)}, j \neq i \end{cases}$$

Then there is a unique occurrence of w which is in block B_i while there is no occurrence of w in v. This is clear for cases 1 and 2, since the strings t and w do not overlap. Concerning Case 3, no nontrivial suffix of w is a prefix of t and no nontrivial prefix of w is a suffix of z. Since at level $\lfloor \frac{n}{2} \rfloor$ the quasi-reduced OBDD contains at least $2^{\frac{n}{6k}}$ different nodes, the size of the diagram is of the order of α^n with $\alpha = 2^{\frac{1}{6k}}$. This completes the proof. ■

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[2] R. Baeza-Yates, V. Bruyère, O. Delgrange, and R. Scheihing. About the size of Boyer-Moore automata. In *Proc. of WSP'96*, pages 31–46. Carlton University Press, 1996.

[3] Ricardo Baeza-Yates, Christian Choffrut, and Gaston Gonnet. On Boyer-Moore Automata. *Algorithmica*, 12:268–292, 1994.

[4] B. Bollig and I. Wegener. Improving the variable ordeing of obdds is np-complete. *IEEE Transactions on Computers*, 45:993–1002, 1996.

[5] R. Boyer and S. Moore. A fast string matching algorithm. *Communications of the ACM*, 20:762–772, 1977.

[6] R. Bryant. Symbolic manipilation of boolean functions using a graphical representation. Number 22, pages 688–694, 1985.

[7] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[8] R. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24:293–318, 1992.

[9] M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, 1994.

[10] Z. Galil. Open problems in stringology. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *F*, pages 1–12. Springer Verlag, Berlin, 1985.

[11] D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.

[12] D.E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in string. *SIAM J. on Computing*, 6:323–350, 1977.

[13] C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systel Technical Journal*, 38:985–999, 1959.

[14] C. Meinel and T. Theobald. On the influence of the state encoding on obdd-representation of finite state-machines. *ITA*, 33:21–32, 1999.

[15] D. Sieling and I. Wegener. Reduction of OBDDs in linear time. *Inform. Proc. Letters*, 48:139–144, 1993.

[16] H. Straubing. *Finite Automata, Formal Logic and Circuit Complexity.* Birkhäuser, 1884.

[17] I. Wegener. The size of reduced OBDDs and optimal read-once branching programs for almost all Boolean functions. *I.E.E.E. Trans. on Computers*, 43:1962–1969, 1994.

[18] I. Wegener. *Branching Programs and Binary Decision Diagrams – Theory and Applications.* SIAM Monographs on Discrete Methods and Applications, 2001.