Some Separation Problems on Randomized OBDDs

Marek Karpinski* Rustam Mubarakzjanov †

Abstract

We investigate the relationships between complexity classes of Boolean functions that are computable by polynomial size branching programs. In the first part of this paper, we consider different general cases of branching programs: deterministic, non-deterministic, randomized and probabilistic, with and without restrictions on times or on order of reading inputs. We are able to show the following. If Q, Q_1, Q_2 are some of these complexity classes such that there are two functions f_1, f_2 in Q but not belonging to Q_1, Q_2 respectively then there is a function $f \in Q \setminus (Q_1 \cup Q_2)$. This fact gives a possibility to show non emptiness of different combinations of the complexity classes.

In the second part of this paper, we show that the class PP (polynomial time probabilistic) and all of the 11 classes obtained by some intersection or union of BPP (polynomial time randomized), NP and coNP are different for the ordered case of read-once branching programs.

We present also some complexity results on other classes of branching programs.

*Dept. of Computer Science, University of Bonn, and International Computer Science Institute, Berkeley, California. Research partially supported by DFG Grant KA 673/4-1, by the ESPRIT BR Grants 7097, and EC-US 030, and by the Volkswagen-Stiftung. Email: **marek@cs.uni-bonn.de**

†Dept. of Computer Science, University of Bonn. Visiting from Dept. of Theoretical Cybernetics University of Kazan. Research supported by the Volkswagen-Stiftung and partially by the Russia Fund for Basic Research 96-01-01692. Email: **rustam@ksu.ru**

1 Preliminaries

We recall basic definitions.

A *deterministic* branching program P for computing a Boolean function $h_n : \{0,1\}^n \to \{0,1\}$ is a directed acyclic multi-graph with a source node and two distinguished sink nodes: accepting and rejecting. The other nodes are called *internal* nodes. The out-degree of each non-sink node is exactly 2 and the two outgoing edges are labeled by $x_i = 0$ and $x_i = 1$ for variable x_i associated with the node. Call such a node an x_i-node. The label "$x_i = \delta$" indicates that only inputs satisfying $x_i = \delta$ may follow this edge in the computation. The branching program P computes function h_n in the obvious way: for each $\overline{\sigma} \in \{0,1\}^n$ we let $h_n(\overline{\sigma}) = 1$ iff there is a directed path starting in the source and leading to the accepting node such that all labels $x_i = \sigma_i$ along this path are consistent with $\overline{\sigma} = \sigma_1\sigma_2 \ldots \sigma_n$.

The branching program becomes *non-deterministic* if we allow "guessing nodes" that is nodes with two outgoing edges being unlabeled. A non-deterministic branching program P computes a function h_n in the obvious way; that is, $h_n(\overline{\sigma}) = 1$ iff there exists (at least one) computation on $\overline{\sigma}$ starting in the source node and leading to the accepting node.

A *probabilistic* branching program has in addition to its standard (deterministic) nodes specially designated nodes called random nodes. Each such a node corresponds to a random input y_i having values from $\{0,1\}$ with probabilities $\{1/2, 1/2\}$. The output of such a program is a random variable.

We say that a probabilistic branching program b-((a,b)-)*computes* a function h if it outputs 1 with a probability at least b for an input $\mathbf{x}$ such that $h(\mathbf{x}) = 1$ (and outputs 1 with a probability at most a for an input $\mathbf{x}$ such that $h(\mathbf{x}) = 0$). We call a probabilistic branching program randomized if it $(1/2-\epsilon, 1/2+\epsilon)$-computes the function h for $\epsilon > 0$.

For a branching program P we define $size(P)$ (the *complexity* of the branching program P) as the number of its internal nodes. The complexity of a probabilistic branching program is the sum of random nodes and x_i-nodes. The complexity of a non-deterministic branching program is the number of its internal nodes (without "guessing" nodes).

A *read-k-times* branching program has the restriction that along each path from the source to the accepted sink, each variable may be tested at most k times. *Read-once* branching program is a branching program in which for each path every variable is tested no more than once.

An *ordered* read-k-times branching program (k-OBDD, OBDD for $k = 1$) is a branching program respecting some ordering π of variables. Such a program can be partitioned into k layers. For each layer, the variables have to be tested according to the ordering π .

Since branching programs are non-uniform model of computation, asymptotic statements about size refer to families of functions and of branching programs computing these functions and containing one program for each input size. This family of functions we call just a function implying that the function depends of the number of variables.

Following definitions of [S97a], we denote the class of Boolean functions which are computable by polynomial size deterministic (non-deterministic) bra-nching programs by $P{-}BP$ ($NP{-}BP$). The class $coNP{-}BP$ contains all Boolean functions the negations of which are computable by polynomial size non-deterministic branching programs.

We say that a function h_n belongs to a set $PP_{\{p_n\}}{-}BP$ for some sequence of numbers $\{p_n\}$ iff for any natural number n, there is a polynomial size probabilistic branching program B_n of n deterministic inputs p_n-computing the function h_n of n variables. Let $PP_{\{p_n\}}{-}BP = PP_p{-}BP$ if $p_n = p$ for any n. It is shown in [AKM98] that $PP_{1/2}{-}BP = PP_p{-}BP$ for any $p, 0 < p < 1$. Therefore we write just $PP - BP$ instead of $PP_p{-}BP$.

For the (a, b)-computation, $a < b$, we use other notation. Let $BPP_\epsilon{-}BP$ be the class of functions $(1/2{-}\epsilon, 1/2{+}\epsilon)$-computable by polynomial size probabilistic branching programs. We call such branching programs randomized. Furthermore, let

$$BPP - BP := \bigcup_{0<\epsilon\leq 1/2} BPP_\epsilon - BP.$$

We define analogous classes for $k - OBDD$s using "$-k - OBDD$" and for read-k-times branching programs using "$-BPk$" as a suffix to their notations.

Because $BPP = coBPP$ and $PP = coPP$, there are for every type of branching programs, 4 complexity classes of our interest: NP, $coNP$, BPP, PP . What is the relationship between these classes ?

In 1996 Ablayev and Karpinski found a function f_n which belonged to BPP–$OBDD$ (and the same time to $coNP-OBDD$) but did not belong to $NP-OBDD$ [AK96]. In 1997 Ablayev found a function from the class NP–$OBDD \setminus BPP$–$OBDD$ [A97]. These results are valid for the complexity classes of ordered branching programs. In 1997 Sauerhoff [S97a] shown that a function $PERM$ corresponding to a permutation matrix is in $(BPP–OBDD \cap coNP–OBDD) \setminus NBP-BP1$.

2 Operations closed for complexity classes

Lemma 1 *Let a Boolean function $h(\mathbf{x})$ be presented as follows*

$$h(\mathbf{x}) = h(\mathbf{x_1}, \mathbf{x_2}) = h_1(\mathbf{x_1}) \& h_2(\mathbf{x_2}).$$

If for $i = 1, 2$, $h_i \in Q$, where

$$Q = NP-BP \text{ or } Q \in \{NP-BPk, NP-k-OBDD\}$$

for some k, then

$$h(\mathbf{x_1}, \mathbf{x_2}) \in Q.$$

Proof. A branching program $B(h)$ computing h consists of two parts. The first part of $B(h)$ is a non-deterministic branching program $B(h_1)$ that computes the function h_1. Then the accepting sink node of $B(h_1)$ is identified with the source node of non-deterministic branching program $B(h_2)$ that computes h_2. If $B(h_1)$ and $B(h_1)$ correspond to the complexity class Q then $h(\mathbf{x_1}, \mathbf{x_2}) \in Q$. ∎

Corollary 1 *Let a Boolean function $h(\mathbf{x})$ be presented as follows*

$$h(\mathbf{x}) = h(\mathbf{x_1}, \mathbf{x_2}) = h_1(\mathbf{x_1}) \vee h_2(\mathbf{x_2}).$$

If for $i = 1, 2$, $h_i \in Q$, where

$$Q \in \{coNP-BP, coNP-BPk, coNP-k-OBDD\}$$

for some k, then

$$h(\mathbf{x_1}, \mathbf{x_2}) \in Q.$$

Lemma 2 *Let a Boolean function $h(\mathbf{x})$ be presented as follows*

$$h(\mathbf{x}) = h\mathbf{x_1}, \mathbf{x_2}) = h_1(\mathbf{x_1}) + h_2(\mathbf{x_2}).$$

If for $i = 1, 2$, $h_i \in Q$, where

$$Q \in \{PP\!-\!BP, BPP\!-\!BP\}$$

or

$$Q \in \{PP\!-\!BPk, PP\!-\!k\!-\!OBDD, BPP\!-\!BPk, BPP\!-\!k\!-\!OBDD\}$$

for some k, then

$$h(\mathbf{x_1}, \mathbf{x_2}) \in Q.$$

Proof. Because of a result of [AKM98], one can assume that the function h_1, h_2 are 1/2- ($(1/2-\epsilon, 1/2+\epsilon)$-) computable by probabilistic (randomized) branching programs B_1, B_2. Firstly, we consider the case of randomized branching programs.

A branching program $B(h)$ computing h consists of two parts. The first part of $B(h)$ is a randomized branching program $B(h_1)$ that computes the function h_1. Then the rejecting sink node of $B(h_1)$ is identified with source node of branching program $B(h_2)$ that computes h_2. The accepting sink node of $B(h_1)$ is identified with source node of branching program $B'(h_2)$ that is a copy of $B(h_2)$ with one exception: the places of the sink nodes are changed.

Let $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2})$. If the probability of computing 1 on $\mathbf{x_i}$ by $B(h_i)$ is p_i for $i = 1, 2$ then $B(h)$ computes 1 with the probability

$$p = p_1 + p_2 - 2p_1p_2 = p_1(1 - 2p_2) + p_2.$$

Let $h(\mathbf{x}) = 1$. Then $h_1(\mathbf{x_1}) = 1, h_2(\mathbf{x_2}) = 0$ or $h_1(\mathbf{x_1}) = 0, h_2(\mathbf{x_2}) = 1$. For the first case

$$p_1 \geq 1/2 + \epsilon, p_2 \leq 1/2 - \epsilon.$$

Therefore $1 - 2p_2 \geq 0$ and

$$p = p_1(1 - 2p_2) + p_2 \geq (1/2 + \epsilon)(1 - 2p_2) + p_2$$

$$= 1/2 + \epsilon - 2p_2\epsilon \geq 1/2 + \epsilon - 2(1/2 - \epsilon)\epsilon = 1/2 + 2\epsilon^2.$$

If $h_1(\mathbf{x_1}) = 0, h_2(\mathbf{x_2}) = 1$ then $p \geq 1/2 + 2\epsilon^2$ too.

Let $h(\mathbf{x}) = 0$. Then $h_1(\mathbf{x_1}) = h_2(\mathbf{x_2}) = 0$ or $h_1(\mathbf{x_1}) = h_2(\mathbf{x_2}) = 1$. For the first case

$$p_1 \leq 1/2 - \epsilon, p_2 \leq 1/2 - \epsilon.$$

Therefore

$$p = p_1(1 - 2p_2) + p_2 \leq (1/2 - \epsilon)(1 - 2p_2) + p_2$$

$$= 1/2 - \epsilon + 2p_2\epsilon \leq 1/2 - \epsilon + 2(1/2 - \epsilon)\epsilon = 1/2 - 2\epsilon^2.$$

If $h_1(\mathbf{x_1}) = h_2(\mathbf{x_2}) = 1$ then

$$p_1 \geq 1/2 + \epsilon, p_2 \geq 1/2 + \epsilon.$$

Therefore $1 - 2p_2 \leq 0$ and

$$p = p_1(1 - 2p_2) + p_2 \leq (1/2 + \epsilon)(1 - 2p_2) + p_2$$

$$= 1/2 + \epsilon - 2p_2\epsilon \leq 1/2 + \epsilon - 2(1/2 + \epsilon)\epsilon = 1/2 - 2\epsilon^2.$$

Therefore $B(h)$ is a randomized branching program that $(1/2 - 2\epsilon^2, 1/2 + 2\epsilon^2)$-computes the function h.

For the case of probabilistic programs, we can prove the Lemma in the same way. ∎

Lemma 1 and Lemma 2 give the following Theorem.

Theorem 1 *Suppose that a family of functions $\{h_1, \ldots, h_t\}$ satisfies*

$$h_i \in Q \setminus Q_i, i \in \{1, \ldots, t\}$$

for the complexity classes $Q, Q_1, \ldots, Q_t$ that are NP, $coNP$, BPP, PP for some type of branching programs. Then the class

$$Q \setminus \cup_{i=1}^{t} Q_i$$

is not empty.

Indeed, using above Lemmas and manipulating the functions h_i we obtain a function h belonging to Q.For each $i \in \{1, \ldots, t\}$ there are assignments for variables of h being not arguments of h_i such that the function h is equal to h_i. Therefore $h \notin Q_i$.

Many complexity classes can be obtained by consideration different kinds of branching programs. It is interesting to know the relationship between these classes. If in some complexity class, such a function exists that does not belong to the other class, it means that some kind of a branching program could be more powerful than other one for some functions. Unions and intersections of complexity classes are interesting too: if a function is "difficult" for some kinds of branching programs, it does not belong to the union of corresponding classes and if some function is in the intersection of some complexity classes it means that this function is "simple" for corresponding branching programs. Different complexity classes, their unions and intersections determine a partially ordered set with respect to $\subseteq$. For some classes Q_1 and Q_2, it is easy to show that, for example, $Q_1 \subseteq Q_2$ but it is not easy to show that the inclusion is proper. We call the inclusion $Q_1 \subseteq Q_2$ evident if $Q_1 = Q_2 \cap Q_3$ for some complexity class Q_3. The following very simple Remark will be useful for us.

Remark 1 *Let Q_1, Q_2, Q_3, Q_4 be some sets such that $Q_1 \subseteq Q_2$, $Q_3 \subseteq Q_4$. If there is an element $f \in Q_1 \setminus Q_4$, then $f \in Q_2 \setminus Q_3$. Therefore it is convenient to find such function $f \in Q_1 \setminus Q_4$ that the class Q_1 would be possible "small" and the class Q_4 would be possible "big".*

3 Unions and intersections of NP, $coNP$, BPP for $OBDD$

We study the complexity classes for $OBDD$ only, in this section. Therefore we use here an abbreviated notation, for example, just BPP instead of $BPP{-}OBDD$. There are 12 complexity classes which relationship is interesting for us: PP, NP, $coNP$, BPP and 4 possible unions and 4 possible intersections of three latter classes. We show that all these classes are different, evident inclusions are proper and there is no non-evident inclusion. The following functions will be used : g_n defined in [A97] (used also in [SZ96a]), f_n defined in [AK96] and q_n defined in [AKM98].

Remark 2 *$g_n \in (NP \cap coNP) \setminus BPP$, $f_n \in (BPP \cap coNP) \setminus NP$, $\neg f_n \in (BPP \cap NP) \setminus coNP$, $q_n \in BPP \setminus (NP \cup coNP)$.*

Because of the Remark 1, the functions in Remark 2 are in certain sense the best possible for the considered complexity classes. Remarks 1 and 2 give the following Lemma.

Lemma 3 *1. The function g_n belongs to the following classes*

$$(NP \cup BPP) \setminus BPP, (coNP \cup BPP) \setminus BPP,$$

$$NP \setminus (NP \cap BPP), coNP \setminus (coNP \cap BPP),$$

$$(NP \cap coNP) \setminus (NP \cap coNP \cap BPP).$$

2. The function f_n belongs to the following classes

$$(NP \cup BPP) \setminus NP, (NP \cup coNP) \setminus NP,$$

$$BPP \setminus (NP \cap BPP), coNP \setminus (NP \cap coNP),$$

$$(BPP \cap coNP) \setminus (NP \cap coNP \cap BPP).$$

3. The function $\neg f_n$ belongs to the following classes

$$(coNP \cup BPP) \setminus coNP, (NP \cup coNP) \setminus coNP,$$

$$BPP \setminus (coNP \cap BPP), NP \setminus (NP \cap coNP),$$

$$(BPP \cap NP) \setminus (coNP \cap NP \cap BPP).$$

4. The function q_n belongs to the following class

$$(NP \cup coNP \cup BPP) \setminus (NP \cup coNP).$$

Lemma 4 *There are functions r_n, r'_n and r''_n such that*

$$r_n \in PP \setminus (NP \cup coNP \cup BPP),$$

$$r'_n \in NP \setminus (coNP \cup BPP), r''_n \in coNP \setminus (NP \cup BPP).$$

Theorem 1 implies this Lemma by combining the functions $g_n, f_n, \neg f_n$. For example, r'_{2n} is obtained from $g_n, \neg f_n$ that are in NP.

Corollary 2 $r'_n \in (NP \cup coNP \cup BPP) \setminus (coNP \cup BPP), r''_n \in (NP \cup coNP \cup BPP) \setminus (NP \cup BPP)$.

Theorem 2 *For the complexity classes PP, NP, $coNP$, BPP and 4 possible unions and 4 possible intersections of three latter classes, the following is true. All these classes are different and there is no non-evident inclusion in the set of these complexity classes.*

The Theorem follows from Lemma 3, Lemma 4 and the following evident fact: if $Q_1 \cup Q_2 \neq Q_1$ for some sets Q_1, Q_2 than $Q_2 \not\subseteq Q_1$.

4 Complexity Classes for read-once Branching Programs

We use some functions and notations defined in other papers. For details of their definitions we refer to corresponding papers.

One can consider relationship between complexity classes determined not only for $OBDD$ but for $k-OBDD$ and read-k-times branching programs. The inclusion $Q' \subseteq Q''$ will be also evident for these classes if this inclusion is a transitive closure of evident inclusions (defined in the previous section) and the inclusions of the following form: $Q_1 \subseteq Q_2$ where $Q_1 = Q$–k–$OBDD$ and $Q_2 = Q$–k+1–$OBDD$ or $Q_2 = Q-BPk$ for an integer number $k \geq 1$ and a complexity class Q. For this consideration, the functions from the Remark 2 are not the best ones. It would be better if one could show that these functions do not belong to complexity classes corresponding to $k-OBDD, k>1$ and $BPk, k\geq 1$. From this point of view, there are no results for the function from Remark 2 except that it has been proved that $f_n \notin NP{-}k{-}OBDD$ [AK98].

It is worth to note that as it was mentioned in [P95b] the Boolean function ACH ("Achilles-Heel") is in $P{-}BP1 \setminus \cup_{k\in N} P{-}k{-}OBDD$. We shall show in this section that ACH is difficult for non-deterministic and randomized OBDD.

Thathachar [T98] gave functions in $P{-}BP(k+1)\setminus NP{-}BP$. Other results of the paper [T98] and a result [S97b] about randomized branching programs presented functions not belonging to BPP_ϵ with small ϵ. There is no known function that does not belong to $BPP-BP1$.

If one considers the complexity class AC^0 and looks for smallest complexity class containing some function it is important to know if the function belongs to AC^0. It is proved in [AKM98] that $f_n, q_n \notin AC^0$. We believe that $g_n \notin AC^0$ but we were not able to prove it.

We summarize known results.

Remark 3 *1. [S97a] The function $PERM$ ("permutation matrix") belongs to $AC^0 \cap BPP{-}OBDD \cap coNP{-}OBDD \setminus NP{-}BP1$;*

2. [JRSW97](Theorem 3.3) there is an explicit Boolean function in $AC^0 \cap NP{-}BP1 \cap coNP{-}BP1 \setminus P{-}BP1$;

3. [S98b] there is an explicit Boolean function $ADDR(\lambda)$ in $BPP \cap NP{-}BP1 \cap coNP{-}BP1 \setminus P{-}BP1$;

4. [P95b] there is an explicit function ISA in $AC^0 \cap P-BP1 \setminus P-OBDD$;

5. [P95b] there is an explicit function HWB in $P-BP1 \setminus AC^0 \cap P-OBDD$.

We show that all these results except the first one can be improved in the sense of the Remark 1 by proving that these function are in

$$NP{-}OBDD \cap coNP{-}OBDD \setminus BPP{-}OBDD.$$

We use some definitions based on those from [S98a].

Let h_n be a Boolean function with n variables X, $d(k)$ be an integer function, $k', k \in \{1, \ldots, n-1\}$, $k' \leq k$. We call h_n as $(k, k', d(k'))$-*stable* if the following holds. For an arbitrary set $X_1 \subset X$, $|X_1| = k$, there are a set $X_2 \subseteq X_1, |X_2| = k'$, a set of assignments S, $|S| \geq d(k)$, to the variables X_2 such that for each variable $x \in X_2$ there is an assignment $\overline{\beta}$ to the variables $X \setminus X_1$ that $h_n(\overline{\alpha} + \overline{\beta})$ is equal to the assignment of x for all $\overline{\alpha} \in S$ or $h_n(\overline{\alpha} + \overline{\beta})$ is not equal to the assignment of x for all $\overline{\alpha} \in S$.

Note that if a function is k-mixed (following the definitions of [JRSW97]) or k-stable (following the definitions of [S98a]) then it is $(k, k, 2^k)$-stable.

We list some examples of $(k, k', d(k'))$-*stable* functions of n variables.

Examples.

1. All 3 functions considered in [JRSW97] a $(k, k, 2^k)$-stable, for the following k (see also [S98a]):
   (a) if $n = q^2 + q + 1$ then $k = (q+1)/2$ if q is prime, $k = \lceil \sqrt{q} \rceil$ otherwise for "characteristic function of blocking set";
   (b) $k = \lceil (n/(2\lceil log_2 n \rceil)^{(1/2)} \rceil - 1$ for the function based on AND of ORs of variables in some blocks ([JRSW97], theorem 3.2);
   (c) $k = \lceil \lceil (n/(\lceil log_2 n \rceil)^{(1/2)} \rceil^2/4 - 1 \rceil$ for the function $ADDR(\lambda)$ based on MAJORITY of MAJORITYs of variables in some blocks ([JRSW97], theorem 3.3)
2. The function $ADDR(\lambda)$ on n variables from [S98a] is $(k, k, 2^k)$-stable, where $\lambda : \{0,1\}^m \to \{0,1\}$, $m = \lceil (n/(\lceil log_2 n \rceil) \rceil$, is a function with the property that any assignment of constant values to at most $k \leq m-1$ variables does not make λ a constant function;
3. HWB from [B91], [P95b] is $(0.6n, 0.1n, \binom{o.2n}{0.1n})$-stable;
4. ISA from [BHR95], Theorem 3, [P95b] is $(k, k, 2^k)$-stable, where $k = \lceil n/\lceil log_2 n \rceil \rceil - 3$;
5. g_n from [A97] is $(n - 3\sqrt{n}, n - 3\sqrt{n}, 2^{n-3\sqrt{n}}/n)$-stable;
6. ACH from [P95b] is $(m/2, m/4, 2^{m/4})$-stable $n = 2m + log_2 m$.

Theorem 3 *Let h_n be a $(k, k', 2^{d'(k')})$-stable function of n variables. Let k' is an unbounded increasing function of n and $d'(k') \geq (1-\delta)k'$ for k' large enough for any $\delta > 0$. Then for every n large enough, it holds that any randomized branching program $(1/2-\epsilon, 1/2+\epsilon)$- computing h_n, $\epsilon < 1/2$ has the size not less than*

$$\frac{1}{4}2^{d'(k')-k'H(p)} \geq \frac{1}{4}2^{k'\delta'},$$

for some $\delta' > 0$ where $p = 1/2+\epsilon$ and $H(p) = -plog_2p - (1-p)log_2(1-p)$ is the Shannon entropy.

Proof. We use ideas of [A97] in this proof. For any X_1, $|X_1| = k$, the function h_n can be described by communication matrix CM for a partition $(X_1, X \setminus X_1)$. Because h_n is a $(k, k', 2^{d'(k')})$-stable function this matrix has a property that it contains $|S|$ different rows corresponding to the set of assignments S, $|S| \geq 2^{d'(k')}$ to the variables X_2 such that for each variable $x \in X_2$ the following holds. There is a column of CM corresponding to this variable x that has different components in two rows iff assignments of x are different.

That means that CM has $|S| \times k'$ Boolean sub-matrix with different rows. Following notations of [A97] we obtain that

$$ts(CM) = k'$$

and for the complexity of randomized branching program $(1-p, p)$-computing the function h_n, the following holds

$$size(P) \geq 2^{PC^U_{p,\pi}(f)-1}$$

where

$$PC^U_{p,\pi}(f) \geq \lceil log|S| \rceil (1 - k'H(p)/log|S|) - 1 \geq d'(k', n) - k'H(p) - 1.$$ ∎

Corollary 3 *All of the functions presented in Examples are not in $BPP-OBDD$.*

Lemma 5 *All of the functions presented in Examples except ACH are in $NP-OBDD \cap coNP-OBDD$. The function ACH is in $coNP-OBDD \setminus BPP-OBDD \cup NP-OBDD$.*

Proof. The way to compute the functions and their negations by non-deterministic read-once program presented in the corresponding papers uses $OBDD$-s with natural order of variables.

It is only necessary to consider the function ACH. Let P be a non-deterministic OBDD computing ACH. One can fix the order of reading variables. Following the proof of Ponzio ([P95b]) for ACH, there is an assignment to Z and w.l.o.g. some set X_1 of the cardinality $m/4$ of variables of X which are read before corresponding variables $Y_1 \subseteq Y$. Let W be the set of nodes when P read all variables of X_1. For any two assignments to the variables of X_1 there is some assignment to nodes of $(X \cup Y \cup Z \setminus (X_1 \cup Y_1))$ and such assignment to Y_1 exists that the function ACH has different values. Therefore the cardinality of W is at least the number of different assignments to X_1: $2^{m/4}$.

To compute the negations of ACH one can read firstly z and read the variables in the following order $x_1, y_1, x_2, y_2, \ldots, x_m, y_m$. If $z = 0$ OBDD tests all the pairs of variables to compute AND of ORs of their negations. If $z \neq 0$ non-deterministic program guesses a 'good' pair where the conjunction of the negations of variables is equal 0. ∎

Theorem 4 *There is an explicit Boolean function in*

$$PP{-}OBDD \cap P{-}BP1 \setminus (BPP{-}OBDD \cup NP{-}OBDD \cup coNP{-}OBDD).$$

Corollary 4 *All the complexity classes being an intersection or an union of $NP, coNP, BPP$ for OBDD are proper subsets of the corresponding classes for read-once branching programs.*

5 Las Vegas Complexity Class versus P for OBDD and BP1

The interesting complexity class is determined by "Las Vegas" (error-free) algorithms. For these randomized algorithms, it is possible an answer "I do not know" with probability less than $\epsilon < 1/2$ for each input. Otherwise it is not allowed to make mistakes: the algorithms give allways correct outputs.

It is shown in [S98b] that the Las–$Vegas$–$BP1$ and P–$BP1$ complexity classes are different (see Remark 3). For OBDDs, we have a surprisingly different result. It is shown in [DHRS97] that the one-way communication complexity of $Las-Vegas$ computation is at most 2 times smaller than the one-way deterministic communication complexity. A proof of this result can be directly transformed to the proof that $P = Las-Vegas$ for "weak-ordered" (see [AK98] for a definition) branching programs and therefore

$$P-OBDD = Las-Vegas-OBDD$$

(the authors proved this fact first for the Las-Vegas *public coin* OBDDs with all random variables read before the deterministic variables [KM98]).

The following theorem presents some relationships between the complexity classes determined as a combinations of those for OBDD and BP1.

Theorem 5 *The function $Addr(\lambda)$ ([S98b]) is in*

$$(NP\text{–}OBDD \cap coNP\text{–}OBDD) \cap Las\text{–}Vegas\text{–}BP1 \setminus BPP\text{–}OBDD \cup P\text{–}BP1$$

and the functions ISA, HWB ([P95b]) are in

$$(NP-OBDD \cap coNP-OBDD) \cap P-BP1 \setminus BPP-OBDD.$$

References

[AK96] F. Ablayev and M. Karpinski, *On the power of randomized branching programs*, Proc. ICALP'96, LNCS 1099, Springer, 1996, pp. 348-356.

[A97] F. Ablayev, *Randomization and nondeterminism are incomparable for ordered read-once branching programs*, Proc. ICALP'97, LNCS 1256, Springer, 1997, pp. 195-202; also available as ECCC TR97-021 (1997) at **http://www.ecc.uni-trier.de/eccc**

[AK98] F. Ablayev and M. Karpinski, *On the power of randomized ordered branching programs*, ECCC TR98-004, 1998, available at `http://www.ecc.uni-trier.de/eccc`

[AKM98] F.Ablayev, M.Karpinski and R.Mubarakzjanov, *On BPP versus NP* $\cup coNP$ *for Ordered Read-Once Branching Programs*, Proc. Randomized Algorithms, Brno, 1998.

[BDG88] J.L.Balcazar, J.Diaz and J.Gabarro, *Structural Complexity I*, Springer, Berlin, 191 P.,1988.

[BRS93] A. Borodin, A. Razborov and R. Smolensky, *On lower bounds for read-k-times branching programs*, Computational Complexity, 3, (1993), 1-18.

[BHR95] Y.Breitbard, H.B.Hunt III and D.Rosenkratz, *On the size of binary decision diagrams representing Boolean functions*, Theoretical Computer Science, 145 (1995), pp.45-69.

[B91] R.E. Bryant, *On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication*, IEEE Trans. Computers, C-40(2): 205-213, Feb.1991.

[DHRS97] P.Duris, J.Hromokovic, J.D.P.Rolim and G.Schnitger,*On the Power of Las Vegas for One-Way Communication Complexity, Finite Automata, and Polynomial-time Computations*, ECCC TR97-029, 1997, available at `http://www.ecc.uni-trier.de/eccc/`

[JRSW97] S.Jukna, A.Razborov, P.Savicky and I.Wegener,*On P versus* $NP \cap co-NP$ *for decision trees and read-once branching programs*, ECCC TR97-023, 1997, available at `http://www.ecc.uni-trier.de/eccc/`

[KM98] M.Karpinski and R.Mubarakzjanov, *Some Separation Problems on Randomized OBDDs*, Manuscript, 1998.

[P95a] S.Ponzio,*A lower bound for integer multiplication with read-once branching programs*, Proc. 27-th STOC, (1995), 130-139.

[P95b] S.Ponzio,*Restricted Branching Programs and Hardware Veryfication*, PhD thesis, Massachusetts Institute of Technology, 1995.

[SZ96a] P. Savicky and S. Zak,*A large lower bound for 1-branching programs*, ECCC, Revision 01 of TR96-036, (1996), available at `http://www.eccc.uni-trier.de/eccc/` .

[SZ96b] P. Savicky and S. Zak,*A hierarchy for* $(1, +k)$*-branching programs with respect to* k, ECCC, TR96-050, (1996), available at http://www.eccc.uni-trier.de/eccc/ .

[S97a] M.Sauerhoff,*A Lower Bound for Randomized Read-k-Times Branching Programs*, ECCC, TR97-019, 1997, available at http://www.eccc.uni-trier.de/eccc/

[S97b] M.Sauerhoff,*On Nondeterminism versus Randomness for Read-Once Branching Programs*, ECCC, TR97-030, 1997, available at http://www.eccc.uni-trier.de/eccc/

[S98a] M.Sauerhoff, *Randomness and Nondeterminism are Incomparable for Read-Once Branching Programs*, ECCC, TR98-018, 1998, available at http://www.eccc.uni-trier.de/eccc/

[S98b] M.Sauerhoff,*Comment 1 on the paper: Randomness and Nondeter-minism are Incomparable for Read-Once Branching Programs*, ECCC, TR98-018, 1998, available at http://www.eccc.uni-trier.de/eccc/

[T98] J.Thathachar, *On Separating The Read-k-Times BranchingProgram Hierarchy*, ECCC, TR98-02, 1998, available at http://www.eccc.uni-trier.de/eccc/