Electronic Colloquium on Computational Complexity
ECCC Technical Reports Series 1994 Report Nr: TR94-026

On the Power of Different Types of Restricted Branching Programs[†]

Beate Bollig, Martin Sauerhoff, Detlef Sieling and Ingo Wegener

FB Informatik, LS II, Univ. Dortmund
44221 Dortmund, Fed. Rep. of Germany
bollig-, sauerhof-, sieling-, wegener@ls2.informatik.uni-dortmund.de

Received *December 23, 1994*

Abstract. Almost the same types of restricted branching programs (or binary decision diagrams BDDs) are considered in complexity theory and in applications like hardware verification. These models are read-once branching programs (free BDDs) and certain types of oblivious branching programs (ordered and indexed BDDs with k layers). The complexity of the satisfiability problem for these restricted branching programs is investigated and tight hierarchy results are proved for the classes of functions representable by k layers of ordered or indexed BDDs of polynomial size.

Keywords: Branching Programs, Binary Decision Diagrams, Lower Bounds, Hierarchy results

[†] Supported in part by DFG grant We 1066/7-1 and -2.

1. INTRODUCTION

Branching programs are a well established computation model for discrete functions.

Definition 1: A branching program G for a function $f: A^n \to B^m$, where $A = \{0, \dots, a-1\}$ and $B = \{0, \dots, b-1\}$ is a directed acyclic graph. The sink nodes are labeled by constants from B. The inner nodes are labeled by variables from $X = \{x_1, \dots, x_n\}$ and have a outgoing edges labeled by the different elements from A. Each node v represents a function $f_v: A^n \to B$. The computation path for f_v and the input $c \in A^n$ starts at v. At an inner node with label x_i the outgoing edge with label c_i is chosen. The label of the sink finally reached is defined as $f_v(c)$. The branching program G represents f, if each coordinate function f_j, $1 \le j \le m$, is represented at some node v_j. The size of G is equal to the number of its edges and is denoted by $|G|$.

In this Introduction we like to describe why researchers working in complexity theory and researchers interested in CAD tools for hardware verification and related problems are interested in similar types of restricted branching programs. The questions they ask are somehow different but nevertheless related. The researchers with different background have worked for a long time independently. One reason is the different notation, e. g. branching programs are called in applications binary decision diagrams. We apply later complexity theoretic methods to models having applications. We start with the models and results from complexity theory.

In the following we consider the Boolean case $a = b = 2$ and one-output functions, i. e. $m = 1$, if nothing else is mentioned.

The branching program size of Boolean functions f is known to be a measure for the space complexity of nonuniform Turing machines and known to lie between the circuit size of f and its $\{\wedge, \vee, \neg\}$-formula size (see e. g. Wegener (1987)). Hence, only small polynomial size lower bounds (Nečiporuk (1966)) can be proved for explicitly defined Boolean functions (excluding diagonalization or counting methods for lower bounds).

Many types of restricted branching programs have been investigated. We mention those two types most relevant for this paper.

Definition 2: A branching program is called read k times if each variable is tested on each path at most k times.

Read-once branching programs have been investigated intensively. Exponential lower bounds have been proved first simultaneously by Žák (1984) and Wegener (1988), even for functions representable by read twice branching programs of polynomial size. For larger k we have exponential lower bounds by Okolnisch'kova (1991) for $k \le \varepsilon(\log n)/\log\log n$ and Borodin, Razborov and Smolensky (1993) for $k \le \varepsilon \log n$ and even nondeterministic branching programs. These lower bound techniques are too coarse to establish tight hierarchies, i. e. it cannot be proved (for explicitly defined functions) that read k times branching programs of polynomial size can represent more Boolean functions than read $(k+1)$ times branching programs of polynomial size (if $k \geq 2$). The considered functions are known to be representable in polynomial size only if $k \geq n^{1/2}$ (the functions of Okolnisch'kova (1991)) or $k \geq n^2$ (the functions of Borodin, Razborov and Smolensky (1993)).

Definition 3: A branching program is called oblivious, if the node set can be partitioned into levels such that edges are leading from lower to higher levels and all inner nodes of one level are labeled by the same variable.

Exponential lower bounds for oblivious branching programs of restricted depth have been proved by Alon and Maass (1986), Babai, Nisan and Szegedy (1992), Krause (1991) and Krause and Waack (1991). The method of Babai, Nisan and Szegedy works up to depth $o(n \log^2 n)$. Again we do not obtain tight hierarchies.

Besides this complexity theoretic viewpoint people have used branching programs in applications. In hardware verification, test pattern generation, symbolic simulation, logical synthesis or analysis and design of circuits and automata (for a survey see Bryant (1992)) one needs representations of Boolean functions which allow efficient algorithms for many operations, in particular synthesis (combine two functions by a binary operation) and equality test (do two representations represent the same function?). The equality test for f and g is equivalent to the satisfiability problem for $f \oplus g$. Hence, the satisfiability problem plays an important role. Therefore, one has to use restricted types of branching programs. Bryant (1986) in his seminal paper has introduced ordered binary decision diagrams (OBDDs).

Definition 4: An OBDD is a branching program respecting a fixed ordering π of the variables, i. e. if an edge leads from an x_i-node to an x_j-node, the condition $\pi(j) > \pi(i)$ has to be fulfilled.

An OBDD can be described also as an oblivious read-once branching program with n levels labeled by the different variables. All important operations can be performed by efficient algorithms for OBDDs (Bryant (1986)) but even simple functions have only representations of exponential size. The following less restricted model has been introduced by Jain, Abadir, Bitner, Fussell and Abraham (1992).

Definition 5: A kIBDD is a branching program which can be partitioned to k layers such that the i-th layer is an OBDD respecting the ordering π_i and such that the edges leaving the i-th layer reach only nodes of the layers $j > i$ and the sinks.

Bitner, Jain, Abadir, Abraham and Fussell (1994) present a lot of successful experiments with IBDDs. Their satisfiability test is a clever heuristic algorithm. Such an approach is necessary, since the satisfiability test is NP-complete already for 2IBDDs. Therefore, we also investigate kOBDDs, a generalization of OBDDs, which are more restricted than kIBDDs.

Definition 6: A kOBDD is a kIBDD, where the permutations $\pi_1, \dots, \pi_k$ are equal to a fixed permutation π.

In Section 2 we investigate the complexity of the satisfiability problem for kOBDDs and kIBDDs. The most important result is that the satisfiability problem can be solved for kOBDDs and constant k in polynomial time.

In Section 3 we apply methods from communication complexity (Nisan and Wigderson (1993)) to prove that the classes PO(k) of Boolean functions representable by kOBDDs of polynomial size form a tight hierarchy, i. e. PO(k) is a proper subclass of PO($k+1$), if $k = o(n^{1/2}/\log^{3/2} n)$.

In Section 4 we prove that also the classes PI(k) of Boolean functions representable by kIBDDs of polynomial size form a tight hierarchy, i. e. PI(k) is a proper subclass of PI($k+1$), if $k \leq (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$.

In the final Section 5 we compare the classes PO(k) and PI(k). Obviously, PO(k) $\subseteq$ PI(k). It follows from the results of Section 3 and Section 4 that PO(k) $\not\subseteq$ PI($k-1$), if $k \leq (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$. We mention an example contained in PI(2) but not in any PO(k) for constant k.

2. THE COMPLEXITY OF THE SATISFIABILITY PROBLEM

The satisfiability problem can be solved in linear time for read-once branching programs (and, therefore, also for OBDDs), since it is sufficient to check whether the 1-sink is reachable from the source. But the satisfiability problem is NP-complete for 2IBDDs. For the sake of completeness we prove this folklore theorem.

Theorem 1: The satisfiability problem for 2IBDDs is NP-complete.

Proof: The problem is contained in NP, since a satisfying input can be guessed.

The satisfiability problem for general branching programs is NP-hard, since conjunctive normal forms can be simulated by branching programs of the same size. Let G be a branching program and let G' be the following 2IBDD based on G. If G contains $k(i)$ nodes labeled by x_i, these labels are replaced by $x_{i,1}, \dots, x_{i,k(i)}$ such that each new variable is used once. The 1-sink of G (w. l. o. g. there is only one) is replaced by a simple OBDD of size $O(k(1) + \dots + k(n)) = O(|G|)$ testing whether for each i the variables $x_{i,1}, \dots, x_{i,k(i)}$ have the same value. It follows that G' is a 2IBDD, $|G'| = O(|G|)$ and that the function represented by G' is satisfiable if and only if the function represented by G is satisfiable. $\square$

As a corollary we obtain that for 2IBDDs it is NP-complete to decide whether they represent different functions and to decide whether the function represented by a 2IBDD depends essentially on the variable x_i.

The reason for the difficulty of the satisfiability problem is the existence of so-called null chains, i. e. paths which are not computation paths, since some x_i-node is left via the 0-edge and another x_i-node is left via the 1-edge. Also kOBDDs contain these null chains. Therefore, it is surprising that the satisfiability problem is solvable for kOBDDs in polynomial time, if k is a constant.

Theorem 2: The satisfiability problem for kOBDDs and constant k is solvable in polynomial time.

Proof: Let G be a kOBDD and let us denote the layers by $G_1, \dots, G_k$. If a is a satisfying input the computation path for a leads through some layers $l(1) = 1 < l(2) < \dots < l(r) \leq k$ of G, where $G_{l(i)}$ is reached at some node v_i (v_1 is the source of G), and from some node in $G_{l(r)}$ the 1-sink is reached. We consider each of the at most $|G|^{k-1}$ possibilities to choose $r, l(2), \dots, l(r), v_2, \dots, v_r$ separately. For a specific choice of the parameters we consider the layers $G_{l(1)}, \dots, G_{l(r)}$ and the sinks. From $G_{l(i)}$ we construct an OBDD $G'_{l(i)}$ with source v_i. An edge e leaving $G_{l(i)}$ is replaced by an edge to a 1-sink, if either $i < r$ and e leads to v_{i+1} or $i = r$ and e leads to a 1-sink. All other edges leaving $G_{l(i)}$ are replaced by edges to a 0-sink.

We conclude that G has a satisfying input if and only if for some $r, l(2), \dots, l(r), v_2, \dots, v_r$ the OBDDs $G'_{l(1)}, \dots, G'_{l(r)}$ have a common satisfying input. Since the OBDDs $G'_{l(1)}, \dots,$ $G'_{l(r)}$ respect the same ordering, the synthesis algorithm for OBDDs (Bryant (1986)) can be applied to obtain in time $O(|G|^k)$ an OBDD G^* of size $O(|G|^k)$ representing the conjunction of the functions represented by $G'_{l(1)}, \dots, G'_{l(r)}$. Finally, the simple satisfiability algorithm for OBDDs is applied to G^*.

Altogether the satisfiability algorithm for the kOBDD G runs in time $O(|G|^{2k-1})$ which is polynomial, if k is a constant. □

3. A TIGHT HIERARCHY FOR kOBDDs

Let PO(k) be the class of Boolean functions representable by kOBDDs of polynomial size. We present explicitly defined functions which are contained in PO(k) but not in PO($k-1$). This implies that the classes PO(k) build a proper hierarchy.

It is obvious that kOBDDs are oblivious branching programs of depth kn. But the lower bound techniques for oblivious branching programs are not precise enough to separate PO($k-1$) from PO(k). We prove the lower bounds by communication complexity methods introduced by Nisan and Wigderson (1993) (and based also on Duris, Galil and Schnitger (1984), Halstenberg and Reischuk (1988), Lam and Ruzzo (1989) and McGeoch (1986)). Nisan and Wigderson (1993) have investigated the following communication game with two players A and B. The input is a directed bipartite graph on $V = \{v_0, \dots, v_{n-1}\}$ and $W = \{w_0, \dots, w_{n-1}\}$ where each node has outdegree 1. Hence, there is a unique path

$p = (p_0, p_1, \ldots, p_m)$ of length m starting at $p_0 = v_0$. Player A knows the edges leaving V and player B knows the edges leaving W. They cooperate to compute p_m. The number of communication rounds is bounded by m. If A may start the communication, a protocol of length $O(m \log n)$ is sufficient. If B has to start the communication, for some $\varepsilon > 0$ a protocol of length $\varepsilon n - m \log n$ is not sufficient to compute only a single bit of p_m.

We investigate pointer jumping functions similar to the problem described above. But our situation is different. In a variable ordering the pointers leaving V and W may be mixed and even the bits describing some pointer may be distributed arbitrarily in the ordering. We overcome these problems in two steps, first we consider functions, where each variable describes a whole pointer, i.e. the variables take values in $\{0, \ldots, n-1\}$ and the output range is also $\{0, \ldots, n-1\}$. In a second step we consider Boolean functions.

We investigate graphs on the node set $U \mathbin{\dot\cup} V \mathbin{\dot\cup} W$, where $U = \{u\}$, $V = \{v_0, \ldots, v_{n-1}\}$ and $W = \{w_0, \ldots, w_{n-1}\}$. The variables $z, x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}$ take values in $\{0, \ldots, n-1\}$. If $z = i$, the pointer from u points to v_i. If $x_j = i$, the pointer from v_j points to w_i. If $y_j = i$, the pointer from w_j points to v_i. (See also Fig. 1.)

(Insert Fig. 1 here.)

The unique path of length $2k+1$ starting at u is denoted by $p = (p_0 = u, p_1, \ldots, p_{2k+1})$. The pointer jumping function $\mathrm{PJ}^{\mathrm{int}}_{k,n}$ works on $z, x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}$ and outputs j, if $p_{2k+1} = v_j$. For the Boolean variant $\mathrm{PJ}^{\mathrm{Bool}}_{k,n}$ we assume that $n = 2^l$ and replace each variable by $l = \log n$ Boolean variables, e.g. x_i by $x_{i,l-1}, \ldots, x_{i,0}$. In order to obtain a Boolean output we add the Boolean variables $c_0, \ldots, c_{n-1}$ describing a coloring $c : V \to \{0, 1\}$ of the nodes in V. The output of $\mathrm{PJ}^{\mathrm{Bool}}_{k,n}$ is $c(p_{2k+1})$, the color of p_{2k+1}.

Theorem 3: The functions $\mathrm{PJ}^{\mathrm{int}}_{k,n}$ and $\mathrm{PJ}^{\mathrm{Bool}}_{k,n}$ can be computed by kOBDDs of size $O(kn^2)$.

Proof: We consider the integer case first. At the source (level 0) z is tested. On level i, $1 \le i \le 2k$, we have n nodes labeled by the variables $x_0, \ldots, x_{n-1}$, if i is odd, and by $y_0, \ldots, y_{n-1}$, if i is even. On level $2k+1$ we have n sinks labeled $0, \ldots, n-1$. Edges from level i lead to level $i+1$. If the edge label is j, the edge reaches the node with label x_j, y_j or j. Hence, we follow the path p. This branching program represents $\mathrm{PJ}^{\mathrm{int}}_{k,n}$ with size $O(kn^2)$. It is a kOBDD respecting the ordering $z, x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}$.

For the Boolean case the inner nodes are replaced by binary decision trees for the Boolean variables replacing the considered integer variable. The sink with label j is replaced by a test of the color variable c_j. The a-successor of this test is the sink with label $a \in \{0, 1\}$. $\square$

Theorem 4: Each $(k-1)$OBDD for $\mathrm{PJ}^{\mathrm{int}}_{k,n}$ has size $2^{\Omega(n/k)}/n$. Each $(k-1)$OBDD for $\mathrm{PJ}^{\mathrm{Bool}}_{k,n}$ has size $2^{\Omega(n^{1/2}/k)}/n$.

Proof: The following proof strategy is used. Let s be the size of a $(k-1)$OBDD G representing $\mathrm{PJ}_{k,n}^{\mathrm{int}}$ or $\mathrm{PJ}_{k,n}^{\mathrm{Bool}}$. Based on the ordering of the variables used by G a communication game for a suitable subfunction of the pointer jumping function is discussed. From G an upper bound on the minimal protocol length for this communication game is derived. Together with the lower bound on the protocol length due to Nisan and Wigderson (1993) we obtain the bound on s.

First we consider a $(k-1)$OBDD for $\mathrm{PJ}_{k,n}^{\mathrm{int}}$. We change the variable ordering such that z becomes the first variable. The size of G is increased at most by a factor n, since we may use n disjoint copies of G, where the tests of z are deleted, as successors of the source. The new $(k-1)$OBDD is again called G. We define a communication game. Let L be the list representing the ordering of the x- and y-variables used by G. We break L in the middle into L_A and L_B. If L_A contains at least $n/2$ x-variables, player A of the communication game obtains z and the x-variables in L_A and player B the y-variables in L_B. Otherwise player A obtains z and the y-variables of L_A and player B the x-variables of L_B. Let $V' \subseteq V$ and $W' \subseteq W$ be the sets of nodes v_i resp. w_j such that x_i resp. y_j is given to some player. The set of inputs is restricted to those graphs, where the edges from V' (resp. $U \cup W'$) reach nodes in W' (resp. V'). The players A and B have to evaluate $\mathrm{PJ}_{k,n}^{\mathrm{int}}$ in $2k-2$ rounds of communication, where A writes in the first round.

Using the $(k-1)$OBDD G it is easy to obtain a communication protocol of length $(2k-2)\lceil \log s\rceil + \lceil \log n\rceil$. First the tests of variables not given to some player are eliminated by fixing these variables, e. g., to 0. Then the i-th layer of G can be partitioned into two sublayers such that player A knows the variables tested in the first sublayer and player B knows the variables tested in the second sublayer. The computation path starts at the source. If a player knows at which node the computation path reaches one of his sublayers, he computes and writes the number of the first node on the computation path not belonging to his sublayer. The number of the sink is written not later than in the $(2k-2)$-th round. The bound on the protocol length follows, since A may write in the first round, which copy of G is used. Afterwards each node can be coded with $\lceil \log s\rceil$ bits.

A lower bound of $\varepsilon n-(2k-2)\lceil \log n\rceil$ for some $\varepsilon > 0$ for the protocol length follows directly from the proof of Theorem 2 in Nisan and Wigderson (1993). Our input restriction is not crucial, since V' and W' contain at least $n/2$ nodes each. The lower bound works for $2k-1$ rounds of communication and even for $2k$ rounds, if player A gets y-variables.

Combining the bounds we obtain $\log s = \Omega(n/k) - \log n$.

For the Boolean case we cannot apply directly the result of Nisan and Wigderson (1993) that the lower bound holds even if only a single bit of p_{2k+1} has to be computed. This is for our communication game no longer true. The nodes in V' may have the same, e. g., last bit. This is the reason why we have introduced the color variables for the Boolean variant $\mathrm{PJ}_{k,n}^{\mathrm{Bool}}$. The second problem is that the Boolean variables representing some pointer may be distributed arbitrarily over the variable ordering used by G.

Let G be an ordered read $(k-1)$ times branching program representing $\mathrm{PJ}_{k,n}^{\mathrm{Bool}}$ with size s. Again it can be assumed w. l. o. g. that $z_{\log n-1}, \ldots, z_0$ are tested only at the top of G. Let

L be the list representing the ordering of the x- and y-variables used by G. For each i we mark in L the $(\log n)/2$-th Boolean variable $x_{i,\cdot}$ and the same for the $y_{i,\cdot}$-variables. Now we break L into L_A and L_B. The breakpoint is the n-th marked variable. If L_A contains at least $n/2$ marked x-variables, player A of our communication game obtains the z-variables and those $x_{i,\cdot}$-variables of list L_A, such that the marked $x_{i,\cdot}$-variable belongs to L_A, and player B obtains those $y_{j,\cdot}$-variables of list L_B, such that the marked $y_{j,\cdot}$-variable belongs to L_B. In the other case L_A contains at least $n/2$ marked y-variables and player A gets the z-variables and some y-variables chosen in a similar way as in the first case the x-variables. In the same way player B gets some x-variables. Let $V' \subseteq V$ and $W' \subseteq W$ be the sets of nodes v_i resp. w_j such that some $x_{i,\cdot}$- resp. $y_{j,\cdot}$-variables are given to some player.

The variables not given to some player are now fixed. Variables belonging to nodes in $V - V'$ or $W - W'$ are set to 0. Let $v_i \in V'$ (the case $w_j \in W'$ is handled similarly).

There are $r \leq n^{1/2}$ different ways to fix the $x_{i,\cdot}$-variables not given to some player. This gives a partition of W into r subsets of equal size $n/r \geq n^{1/2}$. Since W' contains at least $n/2$ nodes, the $x_{i,\cdot}$-variables not given to some player can be fixed in such a way that at least $n^{1/2}/2$ nodes in W' are reachable by an edge from v_i.

We investigate a random coloring of the vertices in V'. It holds for each $w_j \in W'$ that the probability that less than a third of the nodes in V' reachable by an edge from w_j has color 0 or less than a third of these nodes has color 1 is exponentially small (Chernov's bounds). Hence, the color variables can be fixed in such a way that for each $w_j \in W'$ at least a third of the nodes in V' reachable by an edge from w_j has color 0 and at least a third has color 1.

After having fixed the variables in the way described above the players A and B have to evaluate $\mathrm{PJ}^{\mathrm{Bool}}_{k,n}$ in $2k-2$ rounds, where A writes in the first round.

We investigate this new communication game. The upper bound $(2k-2)\lceil \log s \rceil + \lceil \log n \rceil$ follows in the same way as in the non Boolean case, since we have fixed enough variables such that G can be divided into $2k-2$ sublayers and all variables of each layer are known by one of the two players.

The lower bound on s follows, if we can prove a lower bound of $\varepsilon n^{1/2} - (2k-2)\log n$ on the protocol length. For the proof of the lower bound on the protocol length we consider only a subset of the still possible graphs. Let $V'_j \subseteq V'$ be the set of possible direct successors of $w_j \in W'$. We choose $V''_j \subseteq V'_j$ such that $|V''_j| \geq n^{1/2}/3$ and exactly one half of the nodes in V''_j has color 0. The input is chosen at random from the set of all possible graphs, where the edge from u leads to V', the edge from $v_i \in V'$ leads to some node in W'_i, the set of possible successors in W', the edge from w_j leads to some node in V''_j. Now the proof method of Nisan and Wigderson (1993) can be applied directly to obtain the proposed lower bound on the protocol length. $\square$

We remark that the lower bounds of Theorem 4 hold even for kOBDDs, if in the k-th layer only the c-, x- and z-variables may be tested.

As a corollary we obtain the proposed hierarchy result. Here we have to take into account that $\mathrm{PJ}_{k,n}^{\mathrm{Bool}}$ is defined on $\Theta(n \log n)$ Boolean variables.

Corollary 1: $\mathrm{PO}(k-1) \subsetneq \mathrm{PO}(k)$, if $k = o(n^{1/2}/\log^{3/2} n)$.

If we consider functions on n variables taking values in $\{0, \ldots, n-1\}$, a corresponding hierarchy result is proved even for $k = o(n/\log n)$.

Remark: The results of this section are contained already in the extended abstract Sieling and Wegener (1994).

4. A TIGHT HIERARCHY FOR kIBDDs

Let $\mathrm{PI}(k)$ be the class of Boolean functions representable by kIBDDs of polynomial size. The upper bounds for the pointer jumping functions proved in Theorem 3 hold also for kIBDDs. Hence, we look for lower bounds.

Theorem 5: Each $(k-1)$IBDD for $\mathrm{PJ}_{k,n}^{\mathrm{int}}$ has size $2^{\Omega(n/(k2^k))}/n$. This bound grows exponentially, if $k \leq (1-\varepsilon)\log n$ for some $\varepsilon > 0$. The size of $(k-1)$IBDDs for $\mathrm{PJ}_{k,n}^{\mathrm{Bool}}$ is not polynomially bounded, if $k \leq (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$.

Proof: IBDDs may use different variable orderings in different layers. Hence, the set of variables given to the players has to be chosen more carefully. We assume again that z is tested only at the source and consider first a $(k-1)$IBDD G of size s representing $\mathrm{PJ}_{k,n}^{\mathrm{int}}$. The set of good variables GV_0 is initialized as set of all x- and y-variables. Let L_i be the list representing the ordering of the good variables in GV_{i-1} with respect to the variable ordering of the i–th layer of G. We break L_i in the middle. One part contains at least one half of the x-variables in GV_{i-1} and the other part contains the same number of y-variables in GV_{i-1}. The set of good variables GV_i contains exactly these variables. The final set GV_{k-1} contains at least $n/2^{k-1}$ x-variables and at least $n/2^{k-1}$ y-variables.

Let V' resp. W' be the set of nodes v_i resp. w_j such that x_i resp. y_j is good. The set of inputs is restricted to those graphs, where the edges from V' (resp. $U \cup W'$) reach nodes in W' (resp. V'). Player A obtains in each case z and the good x-variables, if they are tested in the first layer of G before the good y-variables, and the good y-variables otherwise. Player B obtains the good variables not given to player A. Both players have to evaluate $\mathrm{PJ}_{k,n}^{\mathrm{int}}$ in $2k-2$ rounds and player A starts writing.

It may happen that in some layers of G the good x-variables are tested before the good y-variables while in other layers of G the good y-variables are tested before the good x-variables. This makes the communication game for the players even simpler. Simulating G a protocol of length $(2k-2)\lceil \log s \rceil + \lceil \log n \rceil$ is obtained. The results of Nisan and

Wigderson (1993) lead to a lower bound of $\varepsilon n/2^{k-1} - (2k-2)\lceil\log(n/2^{k-1})\rceil$. From both bounds on the protocol length the bound $\log s = \Omega(n/(k2^k)) - \log n$ on the $(k-1)$IBDD size s can be derived.

For the Boolean case we combine the ideas of the proof of Theorem 4 for the Boolean case and the ideas for the integer case above. The set of good variables GV_0 is initialized as set of all x- and y-variables. Let L_i be the list representing the ordering of the variables in GV_{i-1} with respect to the variable ordering π_i used in the i-th layer of G. The list L_i contains for at least $n/2^{i-1}$ V-nodes at least $(\log n)/2^{i-1}$ Boolean variables each and the same holds for W. For each node such that a Boolean variable representing this node is contained in L_i the middle variable with respect to L_i is marked. Then the middle variable with respect to all marked variables in L_i is determined and we break L_i after this variable. One part called V-part contains at least $n/2^i$ marked V-variables and the other part called W-part at least $n/2^i$ marked W-variables.

A variable $x_{j,\cdot}$ is included in GV_i if it is in the V-part of L_i and if the marked Boolean variable for v_j is also in the V-part of L_i. W-variables are treated in a similar way. The final set GV_{k-1} contains for at least $n/2^{k-1}$ V-nodes and for at least $n/2^{k-1}$ W-nodes at least $(\log n)/2^{k-1}$ Boolean variables each.

Player A obtains the z-variables and the V-variables in GV_{k-1}, if they are before the W-variables in GV_{k-1} with respect to L_1 and the W-variables in GV_{k-1} otherwise. Player B obtains the other variables in GV_{k-1}.

The variables not given to some player are fixed. Let $V' \subseteq V$ and $W' \subseteq W$ be the sets of nodes v_i resp. w_j such that some $x_{i,\cdot}$- resp. $y_{j,\cdot}$-variables are given to some player. Variables belonging to nodes in $V - V'$ or $W - W'$ are set to 0. Let $v_i \in V'$ (the case $w_j \in W'$ is handled similarly). At most $(1 - 2^{-(k-1)})\log n$ variables of type $x_{i,\cdot}$ have to be fixed and at least $n/2^{k-1}$ of the nodes of W are in W'. Hence, by the pigeonhole principle we can fix the $x_{i,\cdot}$-variables not in GV_{k-1} in such a way that at least

$$N(k) := (n/2^{k-1})/n^{1-2^{-(k-1)}} = n^{2^{-(k-1)}} \cdot 2^{-(k-1)}$$

nodes in W' are still reachable from v_i.

If $k \leq (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$, $N(k) = 2^{\Omega(\log^{\varepsilon} n)}$, and $N(k)$ grows faster than any polylogarithmic function. The following holds for a random coloring of the nodes in V'. By Chernov's bounds the probability that less than a third of the nodes in V' reachable from $w_j \in W'$ has color 0 (or color 1) is bounded above by $2^{-\alpha N(k)}$ for some $\alpha > 0$.

Since $n2^{-\alpha N(k)} < 1$ for large n, we can fix the color variables in such a way that each $w_j \in W'$ has the property that at least a third of the possible successors in V' has color 0 and at least a third has color 1.

Now we continue as in the proof of Theorem 4. The upper bound on the protocol length is $O((\log\log n)\log s + \log n)$ and the lower bound $\Omega(2^{\Omega(\log^{\varepsilon} n)} - (\log\log n)\log n)$. Hence, $\log s = 2^{\Omega(\log^{\varepsilon} n)}$ and s is not polynomially bounded. $\square$

Corollary 2: $\mathrm{PI}(k-1) \subsetneq \mathrm{PI}(k)$, if $k \le (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$.

The corresponding hierarchy for functions on n variables taking values in $\{0,\ldots,n-1\}$ is proved even if $k \le (1-\varepsilon)\log n$ for some $\varepsilon > 0$.

We have ideas for an improved hierarchy result but we are not able to solve the following communication game.

There are k node sets $V_1,\ldots,V_k$ containing n nodes each. To simplify the notation let $V_{k+1} = V_1$. Some node $v_1 \in V_1$ is fixed. The functions $f_i : V_i \to V_{i+1}$ describe pointers from V_i to V_{i+1}. We are interested in the $(k+1)$–th node on the unique path starting at v_1. The path is $p = (v_1,\ldots,v_k,v_{k+1})$, where $v_i \in V_i$.

There are k players and player i knows all f_j except f_i. Each player may write one message and the players have to write according to their numbers, first player 1, then player 2 and so on.

Conjecture: For some $\varepsilon > 0$ protocols of length $\varepsilon n^{\varepsilon}$ are not long enough to compute v_{k+1} or only a single bit of v_{k+1}.

Communication games, where player i knows everything except the i–th part of the information, have already been considered by Babai, Nisan and Szegedy (1992). They proved large lower bounds for difficult functions. Their methods cannot be used here, since our game becomes trivial, if the players may write in arbitrary order. Player 2 may write v_2 and then player 1 may write v_{k+1}. This protocol has length $2\lceil\log n\rceil$.

Remark: The communication game has not been published before. After a talk at the Oberwolfach conference on complexity theory (1994) Noam Nisan pointed out that he and others have also tried to solve the same communication problem and that a proof of the conjecture has even more implications. Independently Babai, Kimmel and Lokam (1994) have considered this communication game and they have obtained results, if all players have to write their messages simultaneously.

We sketch our ideas how we obtain an improved hierarchy for kIBDDs of polynomial size, if the conjecture holds.

We describe a new pointer jumping function $\mathrm{PermPJ}_{k,n}$ on $k\lceil\log k\rceil + (nk+1)\lceil\log n\rceil + n$ Boolean variables:

- $k\lceil\log k\rceil$ variables describe a permutation σ on $\{1,\ldots,k\}$.
- $\lceil\log n\rceil$ variables z_l, $0 \le l < \lceil\log n\rceil$, describe a pointer, a number in $\{0,\ldots,n-1\}$, for a node u.
- $nk\lceil\log n\rceil$ variables $x_{i,j,l}$, $1 \le i \le k$, $0 \le j \le n-1$, $0 \le l < \lceil\log n\rceil$, describe pointers for the nodes $v_{i,j}$.

– n variables c_j, $0 \le j \le n-1$, describe a coloring of n nodes.

The pointer jumping function works on an input in the following way. One starts at u, the pointer from u leads to a node p_1 in $V_{\sigma(1)}$, the pointer from p_i leads to a node p_{i+1} in $V_{\sigma(i+1)}$, where $\sigma(k+1) := \sigma(1)$. The output is the color of p_{k+1}, where the color variables describe a coloring of the nodes in $V_{\sigma(k+1)} = V_{\sigma(1)}$.

Lemma 1: The pointer jumping function $\mathrm{PermPJ}_{k,n}$ is contained in $\mathrm{PO}(k)$, if $k = O((\log n)/(\log\log n))$.

Proof: The following variable ordering is chosen. First the permutation variables followed by the z-variables, then the x-variables and, finally, the color variables. In the first layer a complete decision tree for the permutation variables is used. It is of polynomial size, since $k = O((\log n)/(\log\log n))$. The following parts of the kOBDD are disjoint for different permutations. Still in the first layer it is possible to compute p_2. Then p_{i+1} is computed in the i-th layer. In the last layer it is possible to compute also the color of p_{k+1}. □

If $k = o((\log n)/(\log\log n))$ and the conjecture holds, the pointer jumping function $\mathrm{PermPJ}_{k,n}$ is not contained in $\mathrm{PI}(k-1)$. To prove this claim we consider a $(k-1)$IBDD G of size s representing $\mathrm{PermPJ}_{k,n}$. We assume that the permutation variables and the z-variables are only tested at the top of G. We can always achieve this property without increasing the size of G by more than a polynomial factor. Later we fix the permutation variables and the z-variables.

The block B_i contains all variables $x_{i,\cdot,\cdot}$. Let GV_0 (set of good variables) contain all x-variables and let CV_0 (set of chosen variables) be empty. We ensure that CV_{i-1} contains for each of the $i-1$ blocks $B_{\sigma(1)}, \ldots, B_{\sigma(i-1)}$ ($\sigma(1), \ldots, \sigma(i-1)$ will be defined step by step) for at least n/k nodes $(\log n)/k$ variables each and that GV_{i-1} contains for the other $k-i+1$ blocks for at least $(k-i+1)n/k$ nodes at least $(k-i+1)(\log n)/k$ variables each. Let L_i be the list representing the variables of GV_{i-1} according to the variable ordering π_i used in the i-th layer of G. We run through L_i in the reversed order until we have found for at least n/k nodes of some block B at least $(\log n)/k$ variables each. Then $\sigma(i)$ is defined such that $B = B_{\sigma(i)}$. The set CV_i contains the variables of CV_{i-1} and for n/k nodes of $B_{\sigma(i)}$ $(\log n)/k$ of those variables we have found. The set GV_i contains all variables fulfilling the following conditions:

– the variables are in GV_{i-1}.

– the variables do not belong to $B_{\sigma(i)}$.

– the variables do not belong to nodes for which we have found at least $(\log n)/k$ variables.

– the variables have not been found during our run through L_i.

It is easy to prove that the proposed conditions on GV_i and CV_i are fulfilled. Finally, $\sigma(n)$ is defined such that σ is a permutation. The set CV contains the variables of CV_{k-1} and for at least n/k nodes of $B_{\sigma(k)}$ at least $(\log n)/k$ variables each, where these variables are chosen from GV_{k-1}.

The permutation variables are fixed to describe σ. The node sets $V_1, \ldots, V_k$ are reduced to sets $V_1', \ldots, V_k'$ containing only nodes belonging to chosen variables. The variables for nodes in $V_i - V_i'$ are set to 0. Let $v_{\sigma(i),j} \in V_{\sigma(i)}'$. The variables $x_{\sigma(i),j,\cdot}$ not chosen are fixed with the help of the pigeonhole principle in such a way that at least $n^{1/k}/k$ nodes in $V_{\sigma(i+1)}'$ are reachable. Using Chernov's bounds and the fact that $n^{1/k}/k$ grows faster than any polylogarithmic function, we can fix the color variables in such a way that for each node in $V_{\sigma(k)}'$ the same number of $\Omega(n^{1/k}/k)$ nodes of each color class in $V_{\sigma(1)}'$ is reachable. The z-variables are fixed in such a way that a node in $V_{\sigma(1)}'$ is reached from u.

Player i obtains all chosen variables except those from $B_{\sigma(i)}$. For this situation we consider the communication game. The protocol length cannot be smaller than for the general situation with $n^{1/k}/k$ nodes in each node set. If the conjecture holds and $k = o((\log n)/(\log \log n))$, the protocol length is not bounded by any polylogarithmic function in n.

The given $(k-1)$IBDD G for $\mathrm{PermPJ}_{k,n}$ leads to a protocol for the communication game. After fixing variables in the prescribed way, we can divide the layers of G in the following way. In layer i the chosen variables from block $B_{\sigma(i)}$ are tested after all other chosen variables from blocks $B_{\sigma(j)}$, where $j > i$. We partition the i-th layer into these two sublayers. Player 1 knows everything in the first sublayer of layer 1, player $i \in \{2, \ldots, k-1\}$ knows everything in the second sublayer of layer $i-1$ and in the first sublayer of layer i, and player k knows everything in the second sublayer of layer $k-1$. Hence, we obtain a protocol of length $k\lceil \log s \rceil$.

Combining the two bounds we see that $\log s$ is not bounded by a polylogarithmic function and s is not polynomially bounded.

5. A COMPARISON BETWEEN kOBDDs AND kIBDDs

By definition it is obvious that $\mathrm{PO}(k) \subseteq \mathrm{PI}(k)$ for all k. Since our hierarchy results have been proved for the same pointer jumping functions, we conclude that $\mathrm{PO}(k) \not\subseteq \mathrm{PI}(k-1)$, if $k \leq (1-\varepsilon)\log\log n$ for some $\varepsilon > 0$. It is not always possible to save one layer, if it is allowed to change the ordering from layer to layer. But the change of the variable ordering can be quite powerful.

Proposition 1: PI(2) is not contained in the union of all $\mathrm{PO}(k)$, where $k \in \mathbb{N}$.

Proof: Let PERM be defined on $n \times n$ Boolean matrices X. PERM(X) equals 1 if and only if X is a permutation matrix, or equivalently, if each row and each column of X

contains exactly one entry 1. It is easy to see that PERM $\in$ PI(2). In the first layer a rowwise ordering of the variables is used and in the second layer a columnwise ordering. Krause (1991) has shown that PERM is not contained in any PO(k). □

References

Alon, N. and Maass, W. (1986). Meanders, Ramsey theory and lower bounds for branching programs. In 22nd Symp. on Found. of Computer Science, 410–417.

Babai, L., Kimmel, P. and Lokam, S. V. (1994). Simultaneous messages vs. communication. To appear in STACS'95.

Babai, L., Nisan, N. and Szegedy, M. (1992). Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. Journal of Computer and System Sciences 45, 204–232.

Bitner, J., Jain, J., Abadir, M., Abraham, J. A. and Fussell, D. S. (1994). Efficient algorithmic circuit verification using indexed BDDs. Fault Tolerant Computing Symposium, 266–275.

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

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers 35, 677–691.

Bryant, R. E. (1992). Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys 24, 293–318.

Duris, P., Galil, Z. and Schnitger, G. (1984). Lower bounds of communication complexity. In 16th Symp. on Theory of Computing, 81–91.

Halstenberg, B. and Reischuk, R. (1988). On different modes of communication. In 20th Symp. on Theory of Computing, 162–172.

Jain, J., Abadir, M., Bitner, J., Fussell, D. S. and Abraham, J. A. (1992). IBDDs: An efficient functional representation for digital circuits. European Design Automation Conference, 440–446.

Krause, M. (1991). Lower bounds for depth-restricted branching programs. Information and Computation 91, 1–14.

Krause, M. and Waack, S. (1991). On oblivious branching programs of linear length. Information and Computation 94, 232–249.

Lam, T. and Ruzzo, L. (1989). Results on communication complexity classes. In 4th Structures in Complexity Theory Conference, 148–157.

McGeoch, L. A. (1986). A strong separation between k and $k-1$ round communication complexity for a constructive language. Techn. Rep. CMU-CS-86-157, Carnegie Mellon University, Pittsburg, PA.

Nečiporuk, È. I. (1966). A Boolean function. Soviet Mathematics Doklady 7(4), 999–1000.

Nisan, N. and Wigderson, A. (1993). Rounds in communication complexity revisited. SIAM Journal on Computing 22, 211–219.

Okolnisch'kova, E. A. (1991). Lower bounds on the complexity of realization of characteristic functions of binary codes by branching programs. Diskretnii Analiz 51, 61–83 (in Russian).

Sieling, D. and Wegener, I. (1994). New lower bounds and hierarchy results for restricted branching programs. To appear in Proc. of 20th Workshop on Graph Theoretic Concepts in Computer Science.

Wegener, I. (1987). The complexity of Boolean functions. Wiley-Teubner.

Wegener, I. (1988). On the complexity of branching programs and decision trees for clique functions. Journal of the ACM 35(2), 461–471.

Žák, S. (1984). An exponential lower bound for one-time-only branching programs. In 11th Symp. on Mathematical Foundations of Computer Science, 562–566.

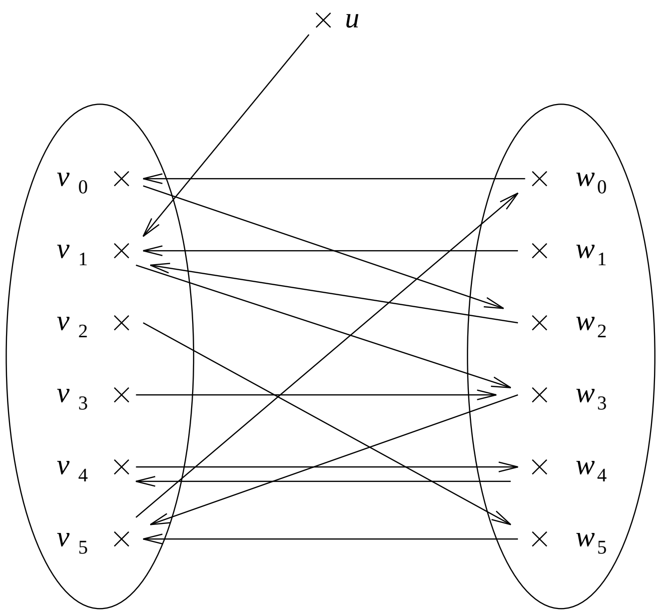

Figure 1: $\mathrm{PJ}^{\mathrm{int}}_{3,6}(z, x_0, \ldots, x_5, y_0, \ldots, y_5) = 1$ for the input in the figure, since $p = (u, v_1, w_3, v_5, w_0, v_0, w_2, v_1)$.