Electronic Colloquium on Computational Complexity - Reports Series 1995 - available via:
ECCC
TR95-042
FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

Read-once Projections and Formal Circuit Verification with Binary Decision Diagrams

Beate Bollig* and Ingo Wegener*

FB Informatik, LS II, Univ. Dortmund, 44221 Dortmund, Germany
e-mail: bollig/wegener@ls2.informatik.uni-dortmund.de

Abstract

Computational complexity is concerned with the complexity of solving problems and computing functions and not with the complexity of verifying circuit designs. The importance of formal circuit verification is evident. Therefore, a framework of a complexity theory for formal circuit verification with binary decision diagrams is developed. This theory is based on read-once projections. For many problems it is determined whether and how they are related with respect to read-once projections. It is proved that multiplication can be reduced to squaring but squaring is not a read-once projection of multiplication. This perhaps surprising result is discussed. For most of the common binary decision diagram models of polynomial size complete problems with respect to read-once projections are described. But for the class of functions with polynomial-size free binary decision diagrams (read-once branching programs) no complete problem with respect to read-once projection exists.

*Supported by DFG grants We 1066/7–2 and 7–3.

1. INTRODUCTION

Computational complexity theory has been developed to measure the complexity of problems, more exactly the complexity of computing functions and solving problems. Functions realized in hardware have circuits of small polynomial size and, therefore, are easy to compute. The distribution of the faulty Pentium division circuit has underlined the fact that new circuit designs should be verified formally. It has to be proved formally that the new circuit meets the specification.

In general this problem is NP-hard even for simple functions, since it is no problem to hide difficult problems in circuits for simple functions. Hence, a general approach for the circuit verification problem has to be a heuristic one. We can expect that in practical applications circuit designs for simple functions do not contain subcircuits which are quite independent from the output function. A lot of verification tools exist. Nowadays, circuit verification is performed most often with binary decision diagrams (for survey articles see Bryant (1992) and Wegener (1994)).

The general approach can be described as follows. A representation (or data structure) for Boolean functions is chosen. The variables should have simple and short representations. For two representations for g and h it should be efficiently possible to construct a representation for $f = g \otimes h$, where $\otimes$ is an arbitrary binary Boolean operator. The specification of the Boolean function f^* is translated with the synthesis algorithm described above step by step into a representation of f^* of the given form. Let f^{**} denote the function computed by the new circuit design. It should be formally proved that $f^* \equiv f^{**}$. Therefore, the circuit is translated gate by gate into a representation of f^{**} of the given form. Sometimes it can be checked directly for two representations whether they represent the same function. Otherwise a representation of $f^* \oplus f^{**}$ is constructed and checked for non-satisfiability. This approach is correct, since $f^* \equiv f^{**}$ iff $f^* \oplus f^{**} \equiv 0$. Circuits are not suitable as representation, since the satisfiability test is NP-complete.

We will consider representations, where the synthesis and the satisfiability problem can be solved in polynomial time. This does not lead to a polynomial time verification algorithm. A sequence of synthesis steps may lead to an exponential blow-up of the representation size even if a single synthesis step may cause only a small polynomial blow-up. The approach is successful, if the function f^* and all functions used in the specification and in the circuit design have short representations. It may depend on the chosen representation whether a verification can be accomplished within the given time and space resource bounds. Often the functions computed at the gates of a circuit for f^* have smaller representations than f^*. Because of these observations and since we cannot know which functions are computed in new circuits for f^*, the size of the representation of f^* is taken as a measure of the hardness to verify circuits for f^*. Based on this notion we develop a framework of a complexity theory for verification with binary decision diagrams. We always have to remember that it is not possible to prove the statement that each circuit for g^* is harder to verify than each circuit for f^*.

In Section 2 we introduce those variants of binary decision diagrams which are used as representations in circuit verification. The notion binary decision diagram is standard

in formal verification. In complexity theory binary decision diagrams are well-known as branching programs.

In Section 3 we discuss why the well-known reduction concepts including (monotone) projections are not appropriate for our purposes. Read-once projections turn out to have the desired properties of a reduction concept for all variants of binary decision diagrams (but not for functional decision diagrams).

One might think that read-once projections are too restricted to be useful. Hence, we show in Section 4 that a lot of known reductions between NP-complete problems and problems in P/poly (polynomial circuit size), in particular the fundamental arithmetic and sorting functions, can be modified to become read-once projections. Here we have to note that practical verification is often concerned with arithmetic functions.

In Section 5 we derive negative results. It is possible to prove that $f = (f_n)$ is not a read-once projection of $g = (g_n)$ by proving that f is more than polynomially harder than g with respect to quite restricted computation models. The most difficult and surprising result is that squaring is not a read-once projection of multiplication although it is a special case of multiplication. On the other hand multiplication is a read-once projection of squaring. We discuss why squaring is for verification purposes harder (for this notion see above) than multiplication.

In Section 6 a slightly more generalized reduction concept is discussed and a hierarchy result is proved.

In order to obtain a framework of a complexity theory we ask whether the classes of functions representable in polynomial size have complete problems with respect to read-once projections for the different models. Although read-once projections are quite restricted we prove in Section 7 that many of the considered complexity classes have complete problems. In Section 8 it is proved that the class of functions representable by polynomial-size free binary decision diagrams (also known as read-once branching programs) does not contain a complete problem with respect to read-once projections.

2. BINARY DECISION DIAGRAMS

We start with the well-known concept of general binary decision diagrams known in complexity theory as branching programs.

Definition 1: A binary decision diagram is a directed acyclic graph with two sinks labeled by the Boolean constants 0 and 1. The other nodes are labeled by Boolean variables and have two outgoing edges one labeled by 0 and the other by 1. Each node represents a Boolean function. The sinks represent the constant functions. If a node v is labeled by x_i, its 0-successor represents g and its 1-successor represents h, then the function represented by v is obtained by Shannon's decomposition rule as $f = \overline{x_i}g \vee x_i h$. The size of the BDD is the number of its nodes.

General BDDs are not used for formal circuit verification, since they have the same problem as circuits that the satisfiability problem is NP-complete and no good heuristic

algorithm exists for the satisfiability problem. Hence, restricted models are used for formal verification.

Definition 2: (i) A free BDD (FBDD or read-once branching program) is a BDD where each path contains for each variable x_i at most one node labeled by x_i.

(ii) An ordered BDD (OBDD) is an FBDD, where the orderings of variables on all paths are consistent with one ordering.

(iii) A k-OBDD consists of k layers of OBDDs respecting the same ordering.

(iv) A k-IBDD consists of k layers of OBDDs respecting perhaps different orderings.

OBDDs introduced by Bryant (1986) are the most popular representation for formal circuit verification. If the ordering is fixed, all operations can be performed in linear time with respect to input and output size. Since many functions have exponential OBDD size, the more general models are also used. It is known (Sieling and Wegener (1995)) that a synthesis of two FBDDs may cause an exponential blow-up of the FBDD size. We consider FBDDs, since typed BDDs (Gergov and Meinel (1994)) and graph driven BDDs (Sieling and Wegener (1995)) have the same expressive power as FBDDs (if only polynomial size is allowed) and allow efficient algorithms for many operations including synthesis and satisfiability test. The satisfiability test is already NP-complete for 2-IBDDs. Nevertheless, Jain, Abadir, Bitner, Fussell and Abraham (1992) have applied IBDDs successfully for the formal verification of complex circuits, where OBDDs, k-OBDDs and FBDDs would fail. They have combined an efficient synthesis algorithm with a heuristic algorithm for the satisfiability test. The more restricted k-OBDDs (Bollig, Sauerhoff, Sieling and Wegener (1994)) allow polynomial-time satisfiability checks, if k is fixed to a constant.

Kebschull, Schubert and Rosenstiel (1992) have introduced functional decision diagrams (FDDs). The syntax is the same as for BDDs. The Shannon decomposition rule is replaced by the Reed-Muller decomposition rule $f = g \oplus x_i h$. Ordered and free FDDs (OFDDs and FFDDs) are defined similarly to OBDDs and FBDDs and are applied successfully for the verification of certain circuits, although a single synthesis step may cause an exponential blow-up of the size (Becker, Drechsler and Werchner(1995)).

3. REDUCTION CONCEPTS

A lot of reduction concepts have been introduced for various types of problems. Projections and monotone projections intensively investigated already by Skyum and Valiant (1985) are perhaps the most restricted ones.

Definition 3: The function $f = (f_n)$, more precisely the sequence of functions, is a (polynomial) projection of $g = (g_n)$, $f \leq_{proj} g$, if

$$f_n(x_1, \ldots, x_n) = g_{p(n)}(y_1, \ldots, y_{p(n)})$$

for some polynomially bounded function p and $y_j \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, 0, 1\}$. For functions with many outputs, each output of f_n has to equal one output of $g_{p(n)}$. The number of j such that $y_j \in \{x_i, \bar{x}_i\}$ is called the multiplicity of x_i. The projection is monotone, $f \leq_{mp} g$, if $y_j \in \{x_1, \ldots, x_n, 0, 1\}$.

In the rest of this paper a projection always means a polynomial projection. A reduction concept "$\leq$" is useful for a representation type T only if $f \leq g$ implies that f has a polynomial-size T-representation if g has a polynomial-size T-representation. Our first result shows that projections are too general for our purposes.

Theorem 1: i) There exist functions f and g, where $f \leq_{proj} g$ for projections of multiplicity 2, g has OBDDs of linear size but f has only FBDDs and k-OBDDs (for constant k) of exponential size.

ii) There exist functions f^* and g^*, where $f^* \leq_{proj} g^*$, g^* has OBDDs of linear size but f^* has for constant k only k-IBDDs of exponential size.

Proof: i) Let X_n and Y_n be $n \times n$ matrices of Boolean variables and $g_n(X_n, Y_n) = 1$ iff X_n contains exactly one 1 in each row and Y_n contains exactly one 1 in each column. This function has $2n^2$ variables and an OBDD of size $O(n^2)$, if first the variables of X_n are tested rowwise and then the variables of Y_n are tested columnwise. Let $f_n(X_n) := g_n(X_n, X_n)$ be a projection of g_n of multiplicity 2. The function f_n tests by definition whether X_n is a permutation matrix. An exponential lower bound on the FBDD size of this function has been proved by Krause, Meinel and Waack (1988) and an exponential lower bound on the k-OBDD size (for constant k) by Krause (1991).

ii) Lower bounds for k-IBDDs have been proved by Bollig, Sauerhoff, Sieling and Wegener (1994) based on communication complexity theoretical methods due to Nisan and Widgerson (1993). We define f_n^* as a pointer jumping function on a graph with vertex set $U \cup V \cup W$, where $U = \{u\}$, $V = \{v_0, \ldots, v_{n-1}\}$ and $W = \{w_0, \ldots, w_{n-1}\}$. The variables z_j, x_{ij}, y_{ij}, $0 \leq j \leq \lceil \log n \rceil - 1$, $0 \leq i \leq n-1$, describe pointers. If $|z| = l$, the pointer from u points to v_l. If $|x_i| = l$ ($|y_i| = l$), the pointer from v_i (w_i) points to w_l (v_l). The further variables $c_0, \ldots, c_{n-1}$ describe a coloring of the vertices in V. The pointer jumping function has to compute the color of the unique vertex v reached by the path of length $2k+3$ starting at u. This function has exponential k-IBDD size.

The number of possible paths is bounded by the polynomial n^{2k+3}. Hence, the pointer jumping function can be described as disjunction of at most n^{2k+3} monoms of length $L := (2k+3)\lceil \log n \rceil + 1$ each.

As g_n^* we define the disjunction of n^{2k+3} monoms of length L each, where all Ln^{2k+3} variables are different. Therefore, g^* has linear size OBDDs with respect to the number of variables. It is sufficient to choose a variable ordering where the monoms are tested monom after monom. By definition $f^* \leq_{proj} g^*$.

□

By Theorem 1 we cannot base complexity theoretical results for OBDDs, k-OBDDs, k-IBDDs or FBDDs on projections even if the multiplicity of each variable is bounded by 2. Theorem 1 i) contains also the result that it can be harder to verify a special case of a function than the general case. Indeed a formal verification of typical circuits for g will be successful with OBDDs and FBDDs but a formal verification of any circuit for f with OBDDs, k-OBDDs or FBDDs leads to representations of exponential size. There is another issue which we will discuss also in Section 5. A clever circuit for a special case may take advantage of the special properties and may cause difficulties for verification.

Conclusion: A complexity theory for verification has to be different from computational complexity theory.

Definition 4: A projection is called read-once, $f \leq_{rop} g$, if the multiplicity of each variable is bounded by 1.

Theorem 2: i) $\leq_{rop}$ is reflexive and transitive.

ii) If $f \leq_{rop} g$ and g has polynomial OBDD (FBDD, k-OBDD, k-IBDD) size, then f has polynomial OBDD (FBDD, k-OBDD, k-IBDD) size.

iii) There exist functions f and g such that $f \leq_{mrop} g$ (m stands for monotone), g has polynomial OFDD size and f has exponential FFDD size.

Proof: i) is obvious.

ii) is easy. If $f_n(x_1, \ldots, y_n) = g_{p(n)}(y_1, \ldots, y_{p(n)})$, we obtain an OBDD for f_n by taking an OBDD for $g_{p(n)}$ and replacing the i-th variable by $y_i \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, 0, 1\}$. The levels labeled by constants can easily be eliminated. A level labeled by $\bar{x}_j$ can be substituted by a level labeled by x_j, if for each node the 0-edge and the 1-edge are exchanged. Then we obtain an OBDD for f_n whose size is not larger than the OBDD for $g_{p(n)}$. The same arguments work for the other models.

iii) If an OFDD or an FFDD represents at node v with label x_i the function f, the 0-successor represents g and the 1-successor h, then $f = g \oplus x_i h$ and g and h do not depend essentially on x_i. Therefore, $f_{|x_i=1} = g \oplus h$. We define the function g_n on $m = \binom{n}{2} + l$, where $l = \lceil \log \binom{n}{3} \rceil$, variables by its OFDD. The OFDD starts with a complete binary tree T of depth l, where the levels are labeled by $y_1, \ldots, y_l$. The other $\binom{n}{2}$ variables x_{rs}, $1 \leq r < s \leq n$, describe a graph $G(x)$. Let $\Delta_{i,j,k}$ for $1 \leq i < j < k \leq n$ decide whether $G(x)$ consists of the triangle $\{i, j, k\}$ and $n-3$ isolated vertices. Then $\Delta_{i,j,k}$ can be described by a single minterm. Hence, it is easy to see that $\Delta_{i,j,k}$ can be described by an OFDD of size $O(n^2)$ for any variable ordering. Each $\Delta_{i,j,k}$ is represented at some leaf of T, the remaining leaves represent 0. This OFDD for g_n has by definition size $O(n^5)$. Let f_n be the restriction of g_n, where $y_1 = \ldots = y_l = 1$. Then f_n is by definition a monotone read-once projection of g_n. By our first argument f_n is the $\oplus$-sum of all $\Delta_{i,j,k}$. Since at most one $\Delta_{i,j,k}$ can be equal to 1, f_n is also the disjunction of all $\Delta_{i,j,k}$ and computes 1 iff the graph

$G(x)$ consists of an arbitrary triangle and $n-3$ isolated vertices. Becker, Drechsler and Werchner (1995) have proved that this function has exponential FFDD size.

□

Theorem 2 has the following implications. Read-once projections are a suitable reduction concept for OBDDs, FBDDs, k-OBDDs and k-IBDDs and a complexity theory for these models may be based on read-once projections. There seems to exist no natural reduction concept adequate for functional decision diagrams.

4. READ-ONCE PROJECTIONS — POSITIVE RESULTS

Read-once projections are quite restricted. So one might be afraid that read-once projections are not powerful enough to compare many important problems. Therefore, we present several read-once projections. We start with NP-complete problems, a P/poly-complete problem and two famous problems with polynomial-time algorithms, although these functions are not realized in hardware. Later we investigate the arithmetic functions usually realized in hardware. For the problems we use the abbreviations of Garey and Johnson (1979).

Theorem 3: (i) (NP-complete problems)
CLIQUE $\leq_{mrop}$ SAT $\leq_{mrop}$ COLOR $\leq_{mrop}$ CLIQUE COVER,
CLIQUE $\leq_{rop}$ IP $\leq_{rop}$ VC $\leq_{rop}$ CLIQUE $\leq_{mrop}$ SUBGRAPH ISO,
DHC $\leq_{mrop}$ HC $\leq_{mrop}$ TSP,
DHP $\leq_{mrop}$ HP $\leq_{mrop}$ BOUNDED DEGREE SPANNING TREE,
SAT $\leq_{mrop}$ 3DM $\leq_{mrop}$ X3C,
3-PARTITION $\leq_{rop}$ SUBFOREST ISOMORPHISM,
3-PARTITION $\leq_{rop}$ SEQUENCING WITH INTERVALS,
BIPARTITE HC $\leq_{mrop}$ PLANAR SUBGRAPH,
CLIQUE $\leq_{mrop}$ MINIMUM NODE-DELETION BIPARTITE SUBGRAPH.

(ii) BIPARTITE PERFECT MATCHING $\leq_{mrop}$ NETWORK FLOW.

(iii) (a P/poly complete problem)
For each function f of polynomial circuit size, i. e. $f \in$ P/poly, $f \leq_{mrop}$ CVP (circuit value problem).

Sketch of proof: (i) The coding of graph problems as Boolean functions is often obvious. Numbers as the desired clique size are given in unary. For SAT we use the coding of Skyum and Valiant (1985).

$$\mathrm{SAT}_n(X,Y) = \bigvee_{a:\{1,\ldots,n\}\to\{0,1\}} \bigwedge_{1\leq i\leq n} \bigvee_{1\leq j\leq n} [(x_{ij} \wedge a(j)) \vee (y_{ij} \wedge \overline{a(j)})].$$

The function has $2n^2$ Boolean variables and can describe all inputs for SAT with n clauses and n variables $z_1, \ldots, z_n$. The variable x_{ij} describes whether z_j is contained

in the i-th clause, y_{ij} does the same for $\bar{z}_j$. The function SAT computes 1 iff there is a variable assignment a such that for all clauses there exists a literal set to 1.

CLIQUE $\leq_{mrop}$ SAT can be proved by adapting the sophisticated proof of Skyum and Valiant (1985) for CLIQUE $\leq_{rop}$ SAT. The read-once projections SAT $\leq_{mrop}$ COLOR and SAT $\leq_{mrop}$ 3DM are based on the polynomial-time reductions from 3SAT to COLOR and 3DM (see Garey and Johnson (1979)). There are some technical difficulties which have to be overcome to make the reductions read-once projections. The result BIPARTITE HC $\leq_{mrop}$ PLANAR SUBGRAPH is based on a polynomial-time reduction due to Yannakakis (1978). All other results are direct translations of known reductions (Garey and Johnson (1979)).

(ii) The well-known reduction from BIPARTITE PERFECT MATCHING to NETWORK FLOW (Even (1979)) is a monotone read-once projection.

(iii) This follows directly from Ladner (1975).

□

Next we consider the fundamental arithmetic and sorting functions which have been investigated by Chandra, Stockmeyer and Vishkin (1984) for constant-depth reducibility. Most of those reductions are not read-once projections and often not even projections. We use the following abbreviations.
AND — conjunction of n variables.
COMP — the decision whether a binary number is smaller than another.
ADD, SUB, MUL, DIV — the usual arithmetic functions.
MULTADD — the addition of n n-bit numbers.
SQU — the squaring of an n-bit number.
INV — the computation of the n most significant bits of the inverse of an n-bit number.
IP — the inner product $x_1y_1 \oplus \ldots \oplus x_ny_n$.
BSUM/USUM — the binary/unary sum of n bits.
MAJ — majority, the decision whether the input has more ones than zeros.
SOR — the sorting of n n-bit numbers.
THR — threshold, where the threshold value is given in unary.
SUB*/SQU* — the highest bit of SUB/SQU.

Theorem 4: AND $\leq_{mrop}$ COMP $\leq_{rop}$ ADD $\leq_{mrop}$ MULTADD $\leq_{mrop}$ MUL $\leq_{mrop}$ SQU $\leq_{mrop}$ INV $\leq_{mrop}$ DIV.
IP $\leq_{mrop}$ MUL.
COMP $\leq_{mrop}$ SUB.
THR $\leq_{mrop}$ MAJ $\leq_{mrop}$ BSUM $\leq_{mrop}$ MULTADD.
MAJ $\leq_{mrop}$ USUM $\leq_{mrop}$ SOR.
SUB* $\leq_{mrop}$ COMP, SQU* $\leq_{mrop}$ COMP.

Proof: AND $\leq_{mrop}$ COMP: $\mathrm{AND}_n(x_0, \ldots, x_{n-1}) = \mathrm{COMP}_n(1, \ldots, 1, 0, x_{n-1}, \ldots, x_0)$.

COMP $\leq_{rop}$ ADD*: $\mathrm{COMP}_n(x_{n-1},\ldots,x_0,y_{n-1},\ldots,y_0) = \mathrm{ADD}^*_n(\bar{x}_{n-1},\ldots,\bar{x}_0,y_{n-1},\ldots,y_0)$, since $|x| < |y|$ iff $x_i = 0$, $y_i = 1$ and $x_j = y_j$ for $j > i$ and some i. In this case we obtain a carry in the addition of y and $\bar{x}$, since $\bar{x}_i = y_i = 1$ and $\bar{x}_j \oplus y_j = 1$ for $j > i$. A carry can be produced only in this way.

ADD $\leq_{mrop}$ MULTADD: Obvious.

MULTADD $\leq_{mrop}$ MUL: If we want to add the n-bit numbers $x^1,\ldots,x^n$, we can multiply the number y which is the concatenation of $x^1,\ldots,x^n$ separated in each case by n zeros and the number z which contains n ones separated in each case by $2n-1$ zeros. Using the school method for multiplication we see that we add $x^1,\ldots,x^n$ and that this sum does not overlap with other sums, since we have separated the numbers by enough zeros.

MUL $\leq_{mrop}$ SQU: The product of the n-bit numbers x and y can be found in the middle of the square of $(x_{n-1},\ldots,x_0,0,\ldots,0,y_{n-1},\ldots,y_0)$, if we take $n+2$ zeros. This again follows by considering the school method for multiplication.

SQU $\leq_{rop}$ INV: This read-once projection is based on ideas from Wegener (1993) who proved optimal lower bounds on the depth of polynomial-size threshold circuits. Let x be the n-bit number we like to square. Let $t = 4n$, $T = 10n$ and $y = x2^{-t} + 2^{-T}$. Since

$$(1-y)^{-1} = 1 + y + y^2 + y^3 + \ldots,$$

we hope that we find x^2 in the binary representation of $(1-y)^{-1}$. This is proved by the following estimation.

$$\begin{aligned}(1-y)^{-1} &= 1 + (x2^{-t} + 2^{-T}) + (x2^{-t} + 2^{-T})^2 + (x2^{-t} + 2^{-T})^3 + \ldots \\ &= (1 + x2^{-t} + x^2 2^{-2t}) + rest.\end{aligned}$$

Obviously, there is no overlap of x and x^2. Hence, it is sufficient to prove that $rest < 2^{-2t} = 2^{-8n}$. We have, since $x \leq 2^n$,

$$\begin{aligned}rest &= 2^{-T} + 2x2^{-t-T} + 2^{-2T} + (x2^{-t} + 2^{-T})^3 + \ldots \\ &\leq 2^{-10n} + 2^{-12n} + 2^{-20n} + 2 \cdot 2^{-9n} < 2^{-8n}, \text{ if } n \geq 2.\end{aligned}$$

The number $(1-y)$ is a read-once projection of x. Let $0.y_{-1}\ldots y_{-T}$ be the binary representation of y and $0.y'_{-1}\ldots y'_{-T}$ the binary representation of $1-y$. Then $y'_{-T} = 1$ and $y'_{-i} = \bar{y}_{-i}$ for $i < T$. Since each x-bit is contained once in y, we are done. The term 2^{-T} was necessary to obtain a read-once projection.

INV $\leq_{mrop}$ DIV: Obvious.

IP $\leq_{mrop}$ MUL: This read-once projection is similar to MULTADD $\leq_{mrop}$ MUL. The first factor contains $x_n,\ldots,x_1$ separated in each case by $\lceil \log n \rceil$ zeros and the second

factor contains $y_1, \ldots, y_n$ separated by $\lceil \log n \rceil$ zeros. Since we have used for x the reversed order, we find in the product $x_1y_1 + \ldots + x_ny_n$ whose last bit is the inner product.

COMP $\leq_{mrop}$ SUB: $x < y$ iff $x - y < 0$. Hence, the sign of $x - y$ is COMP(x, y). This proves also SUB$^* \leq_{mrop}$ COMP.

The next read-once projections are left to the reader, since they are easy. We mention only that SQU$^* \leq_{mrop}$ COMP, since the first bit of x^2 equals 1 iff $x^2 \geq 2^{2n-1}$, i. e. $x \geq (2^{2n-1})^{1/2}$ or $x > \lfloor 2^{n-1/2} \rfloor$. □

Ponzio (1995) has proved that the FBDD size of multiplication is exponential. The same is known for k-OBDDs and constant k. As a corollary of Theorem 4 and Theorem 2 we obtain the same bounds for SQU, INV and DIV.

Theorem 3 and Theorem 4 prove that read-once projections are powerful enough to compare many NP-complete problems as well as many fundamental functions with polynomial circuit size with each other.

5. READ-ONCE PROJECTIONS — NEGATIVE RESULTS

Theorem 2 ii) proves that if $f \leq_{rop} g$ and g can be represented efficiently with respect to one of several models, then f can be represented efficiently with respect to the same model. For negative results we can use the contraposition. This approach leads to concrete results, since we are able to prove large lower bounds for explicitly defined functions with respect to the considered models. The next theorem is a collection of such conditions which have concrete implications. We recall that a function is called symmetric, if the output depends only on the number of ones in the input, and that it is called unate, if it is monotone increasing or monotone decreasing with respect to each variable x_i.

Theorem 5: The function f is not a read-once projection of the function g, if one of the following conditions is fulfilled.

1) g is symmetric and f is not.

2) g is unate and f is not.

3) g_n has an OBDD size $s(n)$ and the OBDD size of f_n is not bounded by a polynomial in $s(n)$ (the same for k-OBDDs, k-IBDDs and FBDDs).

4) g_n has an OBDD width w or $w(n)$ (size of the largest level) and the OBDD width of g_n is not bounded by the constant w or not bounded by a polynomial in $w(n)$ (the same for k-OBDDs, k-IBDDs and FBDDs).

Proof: The result is obvious, since a read-once projection (but not necessarily a projection) of a symmetric function is symmetric. The same holds for unate functions. Condition 3 follows from Theorem 2 ii) and Condition 4 follows in the same way. □

Conditions 3 and 4 can be generalized to a lot of other computation models like $\{\wedge, \vee, \neg\}$-circuits of depth d or threshold circuits of depth d. We list the applications concerning the functions considered in Theorem 4. Additionally let MOD m test whether the number of ones in the input is a multiple of m. Let DSA be the direct storage access function, ISA the indirect storage access function and HWB the hidden weighted bit function (for definitions see Wegener (1994)). From the following characterizations we obtain a lot of negative results.

Among the considered functions exactly the following ones are symmetric: AND, THR with fixed threshold value, MAJ, BSUM, USUM, and MOD m.

Exactly the following functions are unate: AND, COMP, THR, MAJ, USUM, ADD*, SUB*, SQU*.

Exactly the following functions have polynomial OBDD size: AND, COMP, ADD, MULTADD, IP, SUB, THR, MAJ, BSUM, USUM, SOR, SQU*, DSA, MOD m.

The function ISA has polynomial 2-OBDD size and polynomial FBDD size, the same holds for HWB (see Sieling and Wegener (1995)), while MUL, SQU, INV and DIV have exponential k-OBDD size and FBDD size.

Condition 4 has implications for functions with polynomial OBDD size. For the following functions we know the optimal width.

Width 1:	AND, SQU*.
Width 2:	COMP, ADD, SUB, IP.
Width m:	MOD m.
Width $n/2$:	MAJ.

What can be done, if such simple results do not help? For very simple functions we can describe all read-once projections, e. g., among the read-once projections of MOD 4 we do not find MOD 2, although MOD 2 $\leq_{mp}$ MOD 4. For more complicated functions we need special arguments. This is also necessary for our main negative result, namely SQU $\not\leq_{rop}$ MUL.

Theorem 6: SQU $\not\leq_{rop}$ MUL.

Proof: We know of no computation model, where the size of MUL is much larger than that of SQU. The proof strategy is the following one. We assume that

$$(*) \qquad \mathrm{SQU}_n(x_{n-1}, \ldots, x_0) = \mathrm{MUL}_{p(n)}(y_{p(n)-1}, \ldots, y_0, z_{p(n)-1}, \ldots, z_0),$$

where $y_i, z_i \in \{x_0, \bar{x}_0, \ldots, x_{n-1}, \bar{x}_{n-1}, 0, 1\}$, the multiplicity of each x_i is bounded by 1 and p is a polynomial. We replace certain x-variables by constants such that the following holds. Some bit of SQU_n is replaced by an inner product of input length $\Theta(n/\log n)$ and one factor of $\mathrm{MUL}_{p(n)}$ has become a constant. Equality $(*)$ holds also after this replacement by constants. The multiplication with a constant factor is the projection (not read-once projection) of MULTADD. Siu, Bruck, Kailath and Hofmeister (1993) have shown that MULTADD can be realized with threshold circuits of polynomial size and depth 2. This holds also for all projections. Hence, it follows from our assumption

that IP can be realized with threshold circuits of polynomial size and depth 2. This is a contradiction to the well-known lower bound due to Hajnal, Maass, Pudlák, Szegedy and Turán (1987).

We have to make the ideas precise. We replace all x_i by 0, where i is not a multiple of $m := \lceil \log n \rceil + 1$. If we talk in the following about a variable x_i, we assume always that i is a multiple of m. Using the school method for multiplication and squaring we conclude that SQU(x) contains for each $k = lm$, $0 \leq l \leq 2\lfloor (n-1)/m \rfloor$, a block B_k, where the sum of all $x_i x_j$ with $i + j = k$ is computed. The second last bit of B_k equals the $\oplus$-sum of all $x_i x_j$ with $i + j = k$ and $i < j$, since these terms are counted twice. This is the inner product of suitable vectors. Let $N := \lfloor (n-1)/m \rfloor + 1$ be the number of still existing x-variables. The $\binom{N}{2}$ products $x_i x_j$ are spread over $\Theta(N)$ inner products.

Since we have assumed that SQU is a read-once projection of MUL, each x_i (or $\bar{x}_i$) belongs in $(*)$ to the first and not to the second factor or vice versa. A pair (x_i, x_j) is called good, if x_i and x_j belong to different factors, and bad otherwise. If r x-variables belong to the first vector, we have

$$\binom{r}{2} + \binom{N-r}{2} \geq N^2/4 - \Theta(N)$$

bad pairs. The bad pairs are also spread over $\Theta(N)$ inner products. By the pigeon-hole principle there exists one inner product containing $\Omega(N)$ bad pairs. Even one of the two factors contains $\Omega(N)$ bad pairs. We replace all variables by the constant 0 which do not belong to those $\Omega(N)$ bad pairs which are all contained in the same factor and belong to the same inner product.

Now we have achieved our aims. SQU_n still realizes an inner product of input length $\Omega(N) = \Omega(n/\log n)$ and one factor of $\mathrm{MUL}_{p(n)}$ has become a constant. With the arguments presented at the beginning of the proof we have proved our theorem. □

It is our intuitive feeling that squaring is simpler than multiplication, since it is a special case. As we have already discussed after Theorem 1, special functions may be harder to verify than the general case. Hence, we should not be too surprised that SQU(x) = MUL(x, x), MUL $\leq_{mrop}$ SQU but SQU $\not\leq_{rop}$ MUL. For the known BDD variants we do not know of any difference in the representation size of SQU and MUL. It is possible that in some further variant MUL has polynomial size while SQU has exponential size. If a variant fulfils Theorem 2 ii) it is not possible that SQU has polynomial size while MUL has exponential size. With read-once projections we can distinguish SQU and MUL which are equivalent with respect to monotone projections even if the multiplicity is bounded by 2.

6. GENERALIZED READ-ONCE PROJECTIONS AND HIERARCHY RESULTS

Read-once projections are suitable reductions for our purposes but already projections with multiplicity 2 are too general. What about intermediate reduction concepts? First

we investigate, how many variables may be used arbitrarily often such that the statement of Theorem 2 ii) still holds. We say that $f \leq_{rop+k(n)} g$, if f is a projection of g such that for f_n, defined on exactly n variables, at most $k(n)$ variables may have a mulitiplicity larger than 1.

Theorem 7: i) The statements of Theorem 2 ii) hold also, if rop is replaced by $rop+k(n)$ and $k(n) = \mathrm{O}(\log n)$.

ii) If $k(n) = \omega(\log n)$, there are functions such that $f \leq_{rop+k(n)} g$, g has linear OBDD size but f does not have polynomial FBDD size.

Proof: i) Let f_n be a projection of $g_{p(n)}$, where at most $k(n)$ variables have a multiplicity larger than 1. These variables are called repeated variables. Let $G_{p(n)}$ be the OBDD (FBDD, k-OBDD, k-IBDD) of polynomial size for $g_{p(n)}$. We construct an OBDD (FBDD, k-OBDD, k-IBDD) for f_n. It starts with a complete binary tree of $k(n)$ levels labeled by the repeated variables. Each of the $2^{k(n)}$ leaves corresponds to a subfunction f_n, where the repeated variables are replaced by constants. These subfunctions of f_n are read-once projections of $g_{p(n)}$ and can be represented by OBDDs (FBDDs, k-OBDDs, k-IBDDs), whose size is not larger than the size of $G_{p(n)}$. Altogether we obtain for f_n an OBDD (FBDD, k-OBDD, k-IBDD) whose size is bounded by $2^{k(n)} \left|G_{p(n)}\right| + 2^{k(n)}$. Since $G_{p(n)}$ has polynomial size and $k(n) = \mathrm{O}(\log n)$, the size is still polynomial.

ii) As function g_n on $3n$ variables y_{ij}, $1 \leq i \leq n$, $1 \leq j \leq 3$, we consider the function which decides whether in the matrix Y the number of rows consisting of ones only is odd. It is obvious that g_n has linear OBDD size, if we test the variables rowwise. Let f_n^* be a function on $n = \binom{m}{2}$ variables x_{ij}, $1 \leq i < j \leq m$, describing an undirected graph $G(x)$. The function f_n^* decides whether the number of triangles in $G(x)$ is odd. It is easy to see that f_n^* can be described as $\oplus$-sum of $\binom{n}{3}$ terms of length 3 representing the possible triangles. Ajtai, Babai, Hajnal, Komlós, Pudlák, Rödl, Szemerédi and Turán (1986) have proved that the FBDD size of f_n^* is $2^{\Omega(n)}$. Let f_n be defined on n variables and let f_n realize $f_{k(n)}^*$ for some $k(n) = \omega(\log n)$ on its first $k(n)$ variables. Hence, f_n does not depend essentially on $n - k(n)$ of its variables. By our description above it follows that f_n is a projection of g_N, where $N = \binom{n}{3}$. Since f_n depends essentially only on $k(n)$ variables, this is a read-once projection with $k(n)$ exceptions. The FBDD size of f_n is $2^{\Omega(k(n))}$ which grows faster than each polynomial in n.

□

Can our negative results of Section 5 be generalized to read-once projections with repeated variables? This is indeed often the case. In particular, SQU $\not\leq_{rop+\alpha n}$ MUL for $\alpha < 1$. The proof of Theorem 6 works also, if we replace in advance αn variables by constants.

Does the repetition of some variables help in some case? For this problem it is good to investigate the inner product $x_1y_1 \oplus \ldots \oplus x_ny_n$, since it is easy to see which functions we

may obtain by projections. E. g., if we replace $x_1, \ldots, x_k$ by x_1, this is possible with one variable which is repeated k times. It is easy to see that it is not possible to obtain this function, if arbitrarily many variables can be repeated at most $k-1$ times.

Theorem 8: If $1 \leq k(n) \leq n$, there exist functions such that $f \leq_{rop+k(n)} g$ but $f \not\leq_{rop+k(n)-1} g$.

Proof: Let g_n be the inner product of n variables, namely $x_1y_1 \oplus \ldots \oplus x_{n/2}y_{n/2}$ and

$$f_n(z_1, \ldots, z_n) = z_1z_2 \oplus z_2z_3 \oplus \ldots \oplus z_{k(n)-1}z_{k(n)} \oplus z_{k(n)}z_1.$$

Obviously, f_n is a monotone projection of $g_{2k(n)}$, where $k(n)$ variables have a multiplicity of 2. Let us investigate all possible projections of g_m. We can obtain terms 0, 1, z_i, $\bar{z}_i = z_i \oplus 1$, z_iz_j, $\bar{z}_iz_j = z_iz_j \oplus z_j$ and $\bar{z}_i\bar{z}_j = z_iz_j \oplus z_i \oplus z_j \oplus 1$. Each variable used once is contained only in at most one term of length 2. Since f_n contains $k(n)$ variables contained in two terms each and the representation as $\oplus$-sum of positive monoms is unique, we conclude that $f \not\leq_{rop+k(n)-1} g$. □

This theorem establishes a tight hierarchy, the reduction concept $\leq_{rop+k(n)}$ is essentially more powerful than $\leq_{rop+k(n)-1}$.

7. COMPLETE PROBLEMS

Are read-once projections powerful enough to allow complete problems for the class of functions representable by polynomial-size OBDDs (FBDDs, k-OBDDs, k-IBDDs)? Theorem 3 iii) gives us some hope, since the well-known result of Ladner (1975) that CVP is complete for P/poly and P with respect to log-space reductions holds even under the restricted model of monotone read-once projections. CVP is a structurally defined problem which 'includes' all problems of P/poly by construction. We obtain similar results for P-OBDD (the class of all sequences of functions with polynomial OBDD size), P-k-OBDD, if k is fixed, P-k-IBDD, if k is a constant, and P-OFDD. For P-OFDD we obtain a complete problem, although this class is not closed under read-once projections, see Theorem 2 iii).

Theorem 9: The complexity classes P-OBDD, P-k-OBDD, if k is fixed, P-k-IBDD, if k is a constant, and P-OFDD have complete problems with respect to monotone read-once projections.

Proof: We start with the class P-OBDD. Our aim is to construct the complete problem $f = (f_n)$ in such a way that each function g_m on $m \leq n$ variables representable by an OBDD of width $w \leq n$ is a monotone read-once projection of f_n. We define f_n on $N = n(2n\lceil \log n \rceil + 1)$ variables by an OBDD of size $O(n^3)$. This is sufficient, since for $g \in$ P-OBDD the OBDD width of g_n is bounded by a polynomial $p(n)$. Hence, g_n is a monotone read-once projection of $f_{max\{n,p(n)\}}$.

The variables of f_n are divided into n 'true' variables $x_1, \ldots, x_n$ and $2n^2\lceil \log n \rceil$ control variables $c_{i,j,k}$, $1 \leq i \leq n$, $1 \leq j \leq 2n$, $1 \leq k \leq \lceil \log n \rceil$. The function f_n is defined by its OBDD consisting of n layers of depth $2n\lceil \log n \rceil + 1$ each. The first level of the i-th layer contains n nodes and is labeled by x_i. There are $2n$ edges leaving these n nodes. For each edge we have reserved $\lceil \log n \rceil$ levels, the control variables $c_{i,j,\cdot}$ are exclusively used on the $\lceil \log n \rceil$ levels for the j-th edge leaving the x_i-level. These levels build a binary tree with n leaves. These leaves are identified with the n nodes of the x_{i+1}-level. The last layer can be even smaller. We need only trees with two leaves, which have to be identified with the two sinks. The size bound $O(n^3)$ follows from the construction.

We show that each OBDD G^* on $m \leq n$ variables whose width is bounded by n is 'contained' in the OBDD G_n for f_n. We only need the last m layers of G_n. The true variables $x_{n-m+1}, \ldots, x_n$ are identified with the m variables of G^* using the same ordering as in G^*. We still have to organize the connection between the levels labeled by the true variables. If in G^* the j-th edge leaving the i-th level reaches the l-th node on the next level, we can organize this in G_n by replacing the control variables for the j-th edge leaving the x_{n-m+i}-level by appropriate constants.

Almost the same construction can be used for k-OBDDs. In the different layers we use the same variables but different control variables.

For k-IBDDs we use the same construction with n^k tentatively true variables and a suitable number of control variables. According to the actual variable ordering used in the different layers we will choose the n actually true variables. The layers for the tentatively but not actually true variables are replaced by constants in such a way that the j-th node on the first level of the layer is identified with the j-th node on the first level of the next layer. We still have to prove that we can realize with n^k variables $x_{i(1),\ldots,i(k)}$, $1 \leq i(j) \leq n$ for $1 \leq j \leq k$, all variable orderings in the different layers. In the l-th layer we choose an ordering such that $x_{i(1),\ldots,i(k)}$ precedes $x_{j(1),\ldots,j(k)}$, if $i(l) < j(l)$. If a k-IBDD uses the variable orderings $\pi_1, \ldots, \pi_k$ in the different layers we choose the variables $x_{\pi_1(1),\pi_2(1),\ldots,\pi_n(1)}, \ldots,$ $x_{\pi_1(n),\pi_2(n),\ldots,\pi_n(n)}$ as actually true variables. These variables are ordered in the l-th layer according to π_l. Hence, our k-IBDD contains a 'copy' of each k-IBDD on n variables whose width is bounded by n.

For OFDDs we use the same ideas as for OBDDs. The only difficulty results from the control levels. In an OBDD the replacement of a control variable by the constant a is equivalent to the identification with the a-successor. This holds in OFDDs only for $a = 0$. If $a = 1$, the $\oplus$-sum of the 0-successor and the 1-successor is realized according to the Reed-Muller decomposition rule. For each control level i for OBDDs (see Fig. 1) we now use two control levels labeled by c^1 and c^2. This enables us to choose the represented functions among the functions g and h represented at two nodes. If $c^1 = c^2 = 0$, the function g is realized. If $c^1 = c^2 = 1$, the function $g \oplus g \oplus 0 \oplus h = h$ is realized. □

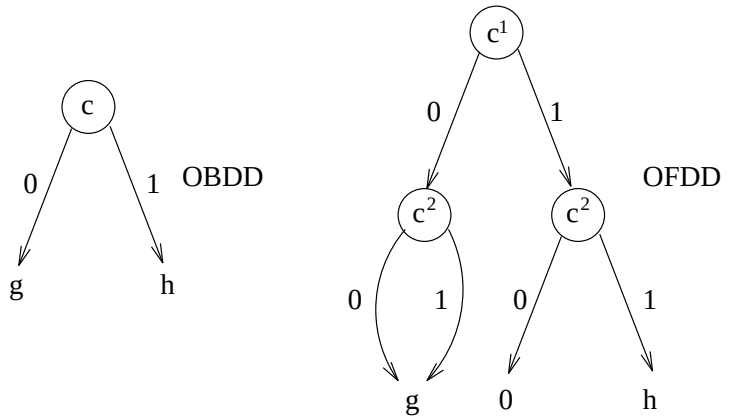

Fig. 1

8. COMPLETENESS — A NEGATIVE RESULT

In OBDDs, OFDDs, and k-OBDDs we may use only one variable ordering which in k-OBDDs may be repeated k times. In k-IDBDs we may use different variable orderings in the different layers. In FBDDs we may choose different variable orderings on all paths. This freedom implies that we cannot obtain complete problems for P-FBDD on a way similar to the constructions used in the previous section. Since P/poly, a superclass of P-FBDD, has complete problems with respect to read-once projections, it is not evident whether P-FBDD has complete problems with respect to read-once projections. We can prove that no complete problem exists.

Theorem 10: The complexity class P-FBDD has no complete problem with respect to read-once projections.

Proof: Let us assume that $f = (f_n)$ represented by FBDDs $G = (G_n)$ of polynomial size is complete for P-FBDD with respect to read-once projections. We can assume that f_n is defined on n variables and w. l. o. g. that each variable is tested on each path of G_n. This increases the size of G_n at most by a factor of $O(n)$. Then it is possible to talk about the different levels of G_n and, therefore, of the width as the maximal size of a level. We also can assume that the width of G_n is bounded by n. Since the width is smaller than the size, the width is bounded by a polynomial $p(n)$. We may add $p(n) - n$ dummy variables and may renumber our sequence. Since the numbers are changed only in a polynomial way, we still have a complete problem with respect to polynomial read-once projections.

Our aim is to describe a class of functions in P-FBDD such that we can find functions on n variables in this class which are not even read-once projections of f_m for $m \leq 2^{n^{1/2}}$. For this purpose we discuss properties of the disjoint quadratic form DQF_n computing $x_1x_2 \vee x_3x_4 \vee \ldots \vee x_{n-1}x_n$. This function has obviously linear OBDD size for the variable ordering $x_1, x_2, \ldots, x_n$. For small size it seems to be necessary that the variables of many pairs $x_{2i-1}x_{2i}$ are tested shortly one after the ofter. We call x_{2i} the partner of x_{2i-1} and

vice versa. If some variable x_j is tested with different results on two computation paths in an FBDD, the partner of x_j is not yet tested and no pair is tested to equal 1, then these two computation paths cannot lead to the same node. The simple reason is that we have to reach different sinks, if the partner of x_j equals 1 and all other untested variables equal 0. We consider a random variable ordering π for DQF_n. A variable is called a singleton, if it is tested according to π among the first $n/2$ variables while its partner is tested among the last $n/2$ variables.

Claim 1: According to a random variable ordering the probability that the number of singletons is at least $n/6$ is $1 - 2^{-\Omega(n)}$.

Proof of Claim 1: There are $\binom{n}{n/2}$ possibilities to choose the first $n/2$ variables in the variable ordering. In order to have exactly k singletons, we have $\binom{n/2}{k}$ possibilities to choose the pairs for which we want to choose singletons. Then there are 2^k possibilities to choose the singletons and $\binom{n/2-k}{n/4-k/2}$ possibilities to choose the pairs among the first $n/2$ variables. The probability to obtain exactly k singletons is 0, if $n/2-k$ is odd, and

$$\binom{n/2}{k} 2^k \frac{\binom{n/2-k}{n/4-k/2}}{\binom{n}{n/2}},$$

if $n/2-k$ is even. We know that the trinomial coefficient

$$\binom{n/2}{k, n/4-k/2, n/4-k/2} = \binom{n/2}{k}\binom{n/2-k}{n/4-k/2}$$

is maximal, if $k = n/6$. Then it equals $\Theta(3^{n/2}n^{-1})$ and $\binom{n}{n/2} = \Theta(2^n n^{-1/2})$. If we sum the probabilities over all $k \leq n/6$, we obtain as upper bound

$$O(2^{n/6}3^{n/2}n^{-1}n^{1/2}2^{-n}) = O(n^{-1/2}2^{\alpha n})$$

for $\alpha = \frac{1}{2}\log 3 - \frac{5}{6} < 0$. □

With DQF_n^π we denote the function $\mathrm{DQF}_n^\pi(x_{\pi(1)}, \ldots, x_{\pi(n)})$. We may interpret Claim 1 in the following way. Each of the variable orderings leads only for a small fraction of all DQF_n^π to an OBDD of small size. The function $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ for permutations $\pi_1, \ldots, \pi_n$ on $\{1, \ldots, n\}$ is defined on $n + \lceil \log n \rceil$ variables $x_1, \ldots, x_n, c_0, \ldots, c_{\lceil \log n \rceil - 1}$. If the control vector $(c_{\lceil \log n \rceil - 1}, \ldots, c_0)$ is the binary representation of i, the function $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ realizes $\mathrm{DQF}_n^{\pi_{i+1}}$, if $i \leq n-1$, and 0 otherwise. We consider the class of all functions $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$. For fixed n these are $(n!)^n = 2^{\Theta(n^2 \log n)}$ functions. Each of these functions can be realized by an FBDD of size $O(n^2)$. It is sufficient to start with a complete binary tree for the control variables. For the value i of the control vector, the variable ordering π_{i+1} is used to realize $\mathrm{DQF}_n^{\pi_{i+1}}$ in linear size.

We want to find functions $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ which are not read-once projections of any f_m, where $m \leq 2^{n^{1/2}}$. This will contradict the assumption that $f = (f_n)$ is complete for

P-FBDD. W. l. o. g. we replace G_n by G_n^* on $n + \lceil \log n \rceil$ variables. The FBDD G_n^* starts with a complete binary tree of the $\lceil \log n \rceil$ new variables and the leaves are sources of disjoint copies of G_n. If $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ is a read-once projection of f_m, then it is also a read-once projection of f_m^* realized by G_m^*. Furthermore, we can assume that the control variables are mapped to the complete binary tree. By renumbering we again denote the FBDDs of the complete problem by G_n and assume that G_n works on at most n variables and has a width bounded by n.

Now we estimate the number of functions $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ which are read-once projections of f_m. We have

$$\binom{m}{n} n! 2^n \leq (2mn)^n$$

possibilities to choose the variables which are replaced by $x_1, \ldots, x_n$ (remember that the control variables are already mapped to the first variables), the ordering of these variables in the read-once projection and the set of variables which are negated in the read-once projection. We fix one of these possibilities. Then we only have the freedom how we replace the remaining variables of f_m by constants. Since G_m is levelled, we can consider the cut through G_m, where exactly $n/2$ x-variables have been tested.

Claim 2: The number of permutations π on $\{1, \ldots, n\}$ such that DQF_n^π is a read-once projection of G_m after one possibility is fixed, is bounded above by $n! 2^{-\Omega(n)}$, if $m \leq 2^{n^{1/2}}$.

Proof of Claim 2: We consider a random permutation π and some node v on the cut of G_m. The size of the cut is by our assumptions bounded by m^2. We have proved in Claim 1 that the set of variables tested before v contains at least $n/6$ singletons with probability $1 - 2^{-\Omega(n)}$. With probability $1 - m^2 2^{-\Omega(n)} = 1 - 2^{-\Omega(n)}$ this holds also for all nodes on the cut simultaneously. It is now sufficient to prove that DQF_n^π cannot be represented, if there are at least $n/6$ singletons for each node on the cut. We fix an assignment of the variables not already replaced by $x_1, \ldots, x_n$. Then we have $2^{n/2}$ paths between the node, where DQF_n^π is assumed to be represented, and the cut. Hence, there is one node on the cut reached by at least $2^{n/2}/m^2$ paths. Each path represents one assignment of the $n/2$ tested x-variables. We consider only assignments where pairs have the value 0. Then we know that two different assignments to the singletons cannot lead to the same node v. Since we have at least $n/6$ singletons, at most $2^{n/2}/2^{n/6} = 2^{n/3}$ paths may lead to v. Since $2^{n/3} < 2^{n/2}/m^2$, we have a contradiction to the assumption that DQF_n^π is presented. □

We conclude from Claim 2 that the number of functions $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$ which are read-once projections of G_m, where $m \leq 2^{n^{1/2}}$, is bounded above by $(n!)^n 2^{-\Omega(n^2)}$. Multiplied by the number of possibilities $(2mn)^n$ and summed over all $m \leq 2^{n^{1/2}}$ this is much less than $(n!)^n$, the number of all considered functions. Hence, a lot of the functions $\mathrm{DQF}_n^{\pi_1,\ldots,\pi_n}$, which all are in P-FBDD, are not read-once projections of any f_m, $m \leq 2^{n^{1/2}}$. Hence, we have a contradiction to the assumption that Theorem 10 is wrong. □

It can be proved by similar arguments that P-FFDD does not contain a complete problem with respect to read-once projections. The proof is based on results of Becker, Drechsler

and Werchner (1995) that graphs where each variable is tested on each path exactly once represent for some easily describable operator τ the function $\tau(f)$ as FFDD, if the graph represents f as FBDD. Hence, we can use the proof of Theorem 10 also for FFDDs.

Conclusion

Is has been shown that a complexity theory for verification has to be different from computational complexity theory. Circuits for special cases may be harder to verify than circuits for the general case. Complexity theoretical results for verification with binary decision diagrams can be based on read-once projections. For many problems and functions it can be decided whether one is a read-once projection of the other or not. Squaring is a special case of multiplication but not a read-once projection of multiplication while multiplication is a read-once projection of squaring. Many natural complexity classes have complete problems even with respect to monotone read-once projections. But for some natural complexity classes it can be proved that they do not have complete problems with respect to read-once projections.

References

Ajtai, M., Babai, L., Hajnal, A., Komlós, J., Pudlák, P., Rödl, V., Szemerédi, E. and Turán, G. (1986). Two lower bounds for branching programs. 18. STOC, 30–38.

Becker, B., Drechsler, R. and Werchner, R. (1995). On the relation between BDDs and FDDs. LATIN '95, LNCS 911, 71–83.

Bollig, B., Sauerhoff, M., Sieling, D. and Wegener, I. (1994). On the power of different types of restricted branching programs. Electronic Colloquium in Computational Complexity, TR 94-026.

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.

Chandra, A., Stockmeyer, L. and Vishkin, U. (1984). Constant depth reducibility. SIAM J. on Computing 13, 423–439.

Garey, M. R. and Johnson, D. B. (1979). Computers and intractability — a guide to the theory of NP-completeness. Freeman, New York.

Gergov, J. and Meinel, C. (1994). Efficient Boolean manipulation with OBDDs can be extended to FBDDs. IEEE Trans. on Computers 43, 1197–1209.

Hajnal, A., Maass, W., Pudlák, P., Szegedy, M. and Turán, G. (1987). Threshold circuits of bounded depth. 28. FOCS, 99–110.

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

Kebschull, U., Schubert, E. and Rosenstiel, W. (1992). Multilevel logic synthesis based on functional decision diagrams. Proc. European Design Automation Conf., 43–47.

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

Krause, M., Meinel, C. and Waack, S. (1988). Separating the eraser Turing machine classes L_e, NL_e, $coNL_e$ and P_e. MFCS, LNCS 324, 405–413.

Ladner, R.E. (1975). The circuit value problem is log space complete for P. SIGACT News 7, 18–20.

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

Ponzio, S. (1995). A lower bound for integer multiplication with read-once branching programs. 27. STOC, 130–139 (see also Ph. D. Thesis, MIT/LCS-TR-633. Sept. 1995).

Sieling, D. and Wegener, I. (1995). Graph driven BDDs - a new data structure for Boolean functions. Theoretical Computer Science 141, 283–310.

Siu, K. S., Bruck, J., Kailath, T. and Hofmeister, T. (1993). Depth-efficient neural networks for division and related problems. IEEE Trans. on Information Theory 39, 946–956.

Skyum, S. and Valiant, L. G. (1985). A complexity theory based on Boolean algebra. Journal of the ACM 32, 484–502.

Wegener, I. (1993). Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. Information Processing Letters 46, 85–87.

Wegener, I. (1994). Efficient data structures for Boolean functions. Discrete Mathematics 136, 347–372.

Yannakakis, M. (1978). Node- and edge-deletion NP-complete problems. 10. STOC, 253–264.