A VERY SIMPLE FUNCTION THAT REQUIRES EXPONENTIAL SIZE READ-ONCE BRANCHING PROGRAMS

Beate Bollig* and Ingo Wegener*

FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
bollig/wegener@ls2.informatik.uni-dortmund.de

Abstract

A Boolean function in n variables is presented that is computable in depth 2 monotone AC^0 and has prime implicants of length 2 only but requires $2^{\Omega(\sqrt{n})}$ size read-once branching programs. The function considered is defined using (1,1)-disjoint Boolean sums and the solution of the famous problem of Zarankiewicz.

Key words: Computational Complexity; Read-once branching programs; Boolean sums; Problem of Zarankiewicz

1 INTRODUCTION

Branching programs are a model of computation measuring the space of (Turing machine) computations . Since the proof of non-polynomial lower bounds on the branching program complexity of explicitly defined Boolean functions is out of the scope of known lower bound methods, several restricted models have been studied.

Many papers consider read-once branching programs. This is the most general model where each graph-theoretical path is the computation path for some input. Read-once branching programs also may be applied under some restrictions as data structure for boolean functions for verification and other purposes [5, 18].

Exponential lower bounds on the read-once branching program complexity of explicit functions are known since a long time. The first bounds [21, 22] are of size $2^{\Omega(\sqrt{n})}$ where n is the number of variables.

The later development follows different directions. One aim was to prove lower bounds as large as possible. The first strongly exponential lower bounds of size $2^{\Omega(n)}$ are contained in [1] with a very small constant and in [11] with a constant of moderate size. The result for the Triangle-Parity function considered in [1] has been improved in [19]. Finally, Savický and Žák[17] have proved a lower bound of size $2^{n-O(n^{2/3}\log^{1/3} n)}$ which is almost optimal due to a general $O(2^n/n)$ upper bound.

*Supported in part by DFG grant We 1066/8.

Motivated by the applications of read-once branching programs the read-once branching program complexity of fundamental functions is of interest. Only recently Ponzio [15] has proved that integer multiplication needs read-once branching programs of exponential size.

Integer multiplication is for other models a quite simple function. It is contained in NC^1 and even in $TC^{0,3}$ (polynomial-size threshold circuits of depth 3) but not in AC^0 (polynomial-size $\{\vee, \wedge, \neg\}$-circuits of constant depth). Many people were interested whether simple functions (with respect to circuit based complexity classes) need read-once branching programs of exponential size. This could be proved for AC^0-functions in [6, 10]. In [3] functions from $AC^{0,2}$(polynomial-size DNF or CNF) are considered but lower bounds only are proved for the restricted model of ordered binary decision diagrams (OBDDs). Gál [4] presents a function from $\Pi^{0,2}$ (polynomial-size CNF) based on elementary properties of projective planes which requires read-once branching programs of size $2^{\Omega(\sqrt{n})}$. This function is remarkably simple and based on important combinatorial structures. It is monotone, the number of prime clauses is small and the length of its prime clauses is approximately $\sqrt{n}$, more precisely $q+1$ for functions on $n = q^2+q+1$ variables.

Here a function in $\Sigma^{0,2}$ (polynomial-size DNF, i.e., the dual function is in $\Pi^{0,2}$) is presented which requires branching programs of size $2^{\Omega(\sqrt{n})}$. This function also is monotone and also is based on the solution of a well-known combinatorial problem, namely the problem of Zarankiewicz (see Section 2). In one sense it is even simpler than the function of Gál. The prime implicants only have length 2, hence, the function belongs to the class of quadratic functions. The lower bound proof (see Section 3) differs from the typical lower bound proofs for read-once branching programs as described in the Lemma of [4].

It should be remarked that exponential lower bounds can be proved for more general than read-once branching programs. After early attempts [9, 10, 11, 16] Borodin, Razborov, and Smolensky [2] proved exponential lower bounds for nondeterministic read-k-times branching programs.

2 DEFINITIONS

A branching program on n variables is a directed acyclic graph with one source. Each sink is labeled by a Boolean constant and each other node by a Boolean variable from $\{x_1, \ldots, x_n\}$. These nodes have two outgoing edges one labeled by 0 and the other by 1. The branching program represents the Boolean function $f : \{0,1\}^n \to \{0,1\}$ defined in the following way. An input $a \in \{0,1\}^n$ activates all edges consistent with a, i.e., the edges labeled by a_i which leave nodes labeled by x_i. The value of the sink reached by the unique path with starts at the source and is activated by a is defined as $f(a)$. The branching program is called read-once if each path contains for each variable x_i at most one node labeled by x_i.

For the definition of the function we like to consider we state the well-known problem of Zarankiewicz [23]. A set system $A_1, \ldots, A_n$ of subsets of $\{1, \ldots, n\}$ is called (h,k)-disjoint if the intersection of $h+1$ sets contains at most k elements. The problem was to maximize the size $|A_1| + \ldots + |A_n|$ of (h,k)-disjoint set systems. If we consider the matrix whose rows are the characteristic vectors of $A_1, \ldots, A_n$, this matrix is the adjacency matrix of a bipartite graph with two sets U and V of n vertices each. The set system is (h,k)-disjoint if $h+1$ vertices of U have at most k neighbors in common. We obtain Boolean functions $f : \{0,1\}^n \to \{0,1\}^n$, called Boolean sums, in the following way. The function f_i is the disjunction of all x_j such that $j \in A_i$. In order to obtain a one-output Boolean function the usual procedure is to consider n additional variables $y_1, \ldots, y_n$ and to define $f^* : \{0,1\}^{2n} \to \{0,1\}$ by $f^*(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigvee_{1 \le i \le n} y_i f_i(x_1, \ldots, x_n)$.

It is not necessary to refer to the extensive literature on the problem of Zarankiewicz. Here it is sufficient to present the solution for $(1,1)$-disjoint sets by Kovari, Sós, and Turán [8]. Let $n = p^2$ for some odd prime number p. The construction is based on the fact that straight lines in $\mathbb{Z}_p$ intersect in at most one point. This is somehow similar to the construction in [4]. Let $a, b \in \mathbb{Z}_p$. The set $A_{a,b}$ (or A_i for $i = a + bp$) contains all $j = c + dp$ where $c, d \in \mathbb{Z}_p$ and $c \equiv (a + bd) \bmod p$. It is known that $|A_i| = p = n^{1/2}$ for all i and $|A_i \cap A_j| \le 1$, if $i \ne j$. Let $B_0, \ldots, B_{n-1}$ be the sets whose characteristic vectors are the columns of the adjacency matrix defined by the set system. Then, by symmetry, $|B_i| = p = n^{1/2}$ for all i and $|B_i \cap B_j| \le 1$, if $i \ne j$.

Boolean sums with n outputs defined by (h,k)-disjoint set systems have a long history in the complexity theory of Boolean functions. Nechiporuk [13] has proved that the monotone circuit complexity of the Boolean sums obtained by the $(1,1)$-disjoint set system defined above equals $\Theta(n^{3/2})$. More general disjoint set systems and their corresponding Boolean sums leading to larger lower bounds on the monotone circuit complexity have been investigated [12, 14, 20]. At that time one tried to prove the first nonlinear lower bounds on the monotone circuit complexity of one-output Boolean functions considering the combinations f^* of these Boolean sums. This approach was not successful.

Here we investigate the Boolean function $f^* = y_1 f_1 \vee \ldots \vee y_n f_n$ on $2n$ variables $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ where f_i is the Boolean sum corresponding to the set A_i of the $(1,1)$-disjoint set system described above. By definition f^* is a monotone function with $n^{3/2}$ prime implicants of length 2 each consisting of an x- and a y-variable. Hence, it is contained in $\Sigma^{0,2} \subseteq AC^{0,2}$. Moreover it is based like Gál's function on a fundamental combinatorial structure.

3 THE LOWER BOUND

The aim is to prove an exponential lower bound on the read-once branching program complexity of the function f^* defined in Section 2. The typical method for such proofs (see [4]) is to prove that each read-once branching program for the considered function f has to start with a complete binary tree of depth d. Then the lower bound is $2^d - 1$. If a prime implicant or a prime clause has length l, then one may reach a sink after l steps. Hence, the proof method only works if all prime implicants and prime clauses are long. In our case the prime implicants have length 2 and we have to generalize the proof method. We choose inner nodes where we decide which successor is chosen. These nodes get outdegree 1 and we connect its predecessor(s) directly with its unique successor. We do this in such a way that the subgraph constructed from the read-once branching program has to start with a complete binary tree of large depth.

Theorem 3.1: The function f^* obtained from the optimal $(1,1)$-disjoint set system has a read-once branching program size of $2^{\Omega(\sqrt{n})}$.

Proof: We use the following notion. The variables x_i and y_j are called partners if $x_i y_j$ is a prime implicant of f^*. A prime clause contains a minimal set S of variables such that the subfunction of f^* where all variables of S are set to 0 is the constant 0. Since each variable has $n^{1/2}$ partners and f^* has $n^{3/2}$ prime implicants, each prime clause has a length of at least n. Let G be a read-once branching program for f^*.

We now describe a set P of paths which together will describe the tree whose existence we like to prove. Each path $p \in P$ starts at the source. Let us assume that we have reached the node v on p. W.l.o.g. the label of v is x_i (the case y_i is handled similarly). Then v is "free" with respect to p if p does not contain a node w labeled by a partner of x_i and left via the 1-edge. In this case we may leave v via the 0-edge or the 1-edge. Otherwise we have to leave v via the 0-edge and v is bound with respect to p. We reach the end of a path $p \in P$ if we have passed $d := \lfloor n^{1/2}/2 \rfloor$ free nodes.

We prove the following statements.

1.) A path $p \in P$ does not reach a sink before passing through d free nodes.

2.) Two different paths $p, p' \in P$ do not meet again after they have separated.

This implies that following the paths from P we reach 2^d different nodes. The paths from P induce a complete binary subtree of G of depth d.

The proof of Statement 1 is easy. We cannot reach the 1-sink. If p follows the 1-edge from a node, then it will by definition never follow the 1-edge from one of its partners. We also cannot reach the 0-sink. Each variable has $n^{1/2}$ partners. Hence, d free variables have at most $d\ n^{1/2}$ partners. The length of p is therefore bounded by $d(n^{1/2}+1) \leq n/2 + n^{1/2}/2 < n$. Hence, not all variables of a prime clause are tested.

For the proof of Statement 2 we prove at first the following Claim. For a path p let f_p^* be the subfunction of f^* which is obtained by replacing all variables tested on p by the constants prescribed by p. Then f_p^* depends for $p \in P$ essentially on all variables not tested on p. Here we use properties of monotone Boolean functions. Let x_i be a variable not tested on p and y_j be a partner of x_i. If f_p^* does not depend essentially on x_i, then we know that either y_j is tested to be 0 on p or y_j is a prime implicant of f_p^*, i.e., some partner of y_j is tested to be 1 on p. Such a partner of y_j is a free variable on p. If y_j is tested to be 0, either a free partner of y_j is tested to be 1 on p or y_j itself is free. We only have d free variables to destroy the $n^{1/2}$ prime implicants containing x_i. Each free variable helps only for one prime implicant containing x_i. This is obvious for free y-variables tested to be equal to 0. If x_k , $k \neq i$, is a free variable tested to be equal to 1, then by the definition of $(1,1)$-disjoint set systems, x_k has at most one common partner y_j with x_i. Then only x_iy_j becomes destroyed. This proves the Claim.

For the proof of Statement 2 it is now sufficient that two paths $p, p' \in P$ testing the same set of variables lead to different subfunctions f_p^* and $f_{p'}^*$ of f^*. There is a variable, w.l.o.g. x_i, which is tested to be 0 on p and 1 on p' (or vice versa). We prove Statement 2 by proving the existence of some x_i-partner y_j which is a prime implicant of $f_{p'}^*$ but not of f_p^*. Since $x_i = 0$ on p, the partners of x_i are not bound because of x_i. If a partner of x_i is tested on p, it may be tested as free variable or as bound variable which is bound by some variable x_k, $k \neq i$. We have already seen that different partners of x_i need different variables x_k. Hence, at most d partners of x_i have been tested on p and at least $\lceil n^{1/2}/2 \rceil$ partners of x_i are not tested on p and because of the Claim also not on p'. On p' the variable x_i is tested to be equal to 1. Hence, all x_i-partners not tested on p' are prime implicants of $f_{p'}^*$. Now we consider f_p^* where $x_i = 0$. An x_i-partner y_j is a prime implicant of f_p^* only if some y_j-partner is tested to be equal to 1. Again we need different free x-variables for different x_i-partners. Since the number of free variables is bounded by $\lfloor n^{1/2}/2 \rfloor$, only $\lfloor n^{1/2}/2 \rfloor$ x_i-partners are prime implicants of f_p^*. Since $n^{1/2}$ is an odd prime number, $\lfloor n^{1/2}/2 \rfloor < \lceil n^{1/2}/2 \rceil$ and we have proved the existence of some y_i which is a prime implicant of $f_{p'}^*$ but not of f_p^*.

Altogether we have proved the Theorem. $\square$

Our proof method may be useful also for other functions.

4 REMARKS

Gál [4] discusses whether Boolean functions may need exponential read-once programs even if they and their negations can be computed by polynomial-size nondeterministic read-once programs. This problem has been solved affirmatively by Jukna, Razborov, Savický, and Wegener [7].

There remains one open problem. It is open wether Boolean functions in the rather small class $\Sigma^{0,2} \cap \Pi^{0,2}$ (polynomial-size DNF *and* polynomial-size CNF) which certainly have polynomial-size nondeterministic and co-nondeterministic read-once branching program size may need deterministic read-once branching programs of exponential size.

References

[1] M. Ajtai, L. Babai, P. Hajnal, J. Komlós, P. Pudlák, V. Rödl, E. Szemerédi, and G. Turán, Two lower bounds for branching programms, in: Proc. of 18th Symp. on Theory of Computing (1986) 30 - 38.

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

[3] S. Devadas, Comparing two-level and ordered binary decision diagram representations of logic functions, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 5 (1993) 722 - 723.

[4] A. Gál, A simple function that requires exponential size read-once branching programs, Information Processing Letters 62 (1997) 13 - 16.

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

[6] S. Jukna, Entropy of contact circuits and lower bound on their complexity, Theoretical Computer Science 57 (1988) 113 - 129.

[7] S. Jukna, A. Razborov, P. Savický, and I. Wegener, On P versus NP $\cap$ coNP for decision trees and read-once branching programs, in: Proc. of Int. Symp. on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science 1295 (1997) 319 - 326.

[8] T. Kovari, V. Sós, and P. Turán, On a problem of K. Zarankiewicz, Colloquium Mathematicum 3 (1954) 50 - 57.

[9] M. Krause, Exponential lower bounds on the complexity of real time and local branching programs, J. Inform. Process. Cybernet. 24 (1988) 99 - 110.

[10] M. Krause, C. Meinel, and S. Waack, Separating the eraser Turing machine classes L_e, NL_e, $co - NL_e$ and P_e, Theoretical Computer Science 86 (1991) 267 - 275.

[11] K. Kriegel and S. Waack, Lower bounds on the complexity of real-time branching programs, in: Proc. of Fundamentals of Computation Theory. Lecture Notes in Computer Science 278 (1987) 263 - 267.

[12] K. Mehlhorn, Some remarks on Boolean sums, Acta Informatica 12 (1979) 371 - 375.

[13] E. I. Nechiporuk, On a Boolean matrix, Systems Theory Research 21 (1971) 236 - 239.

[14] N. Pippenger, On another Boolean matrix, Techn. Rep. IBM Yorktown Heights (1977).

[15] S. Ponzio, A lower bound for integer multiplication with read-once branching programs, in: Proc. of 27th Symp. on Theory of Computing (1995) 130 -139.

[16] A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, in: Proc. of Fundamentals of Computation Theory. Lecture Notes in Computer Science 529 (1991) 47 - 60.

[17] P. Savický and S. Žák, A large lower bound for 1-branching programs, ECCC Report TR 036 - 96 (1996).

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

[19] J. Simon and M. Szegedy, A new lower bound theorem for read-only-once branching programs and its applications, in: Advances in Computational Complexity Theory (ed. J. Cai). DIMACS Series in Discrete Mathematics and Theoretical Computer Science 13 (1993) 183 - 193.

[20] I. Wegener, A new lower bound on the monotone network complexity of Boolean sums, Acta Informatica 13 (1980) 109 - 114.

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

[22] S. Žák, An exponential lower bound for one-time-only branching programs, in: Proc. of Int. Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 176 (1984) 562 - 566.

[23] K. Zarankiewicz, Problem 101, Colloquium Mathematicum 2 (1951) 301.