Theoretical Computer Science 66 (1989) 15–25
North-Holland

ON MONOTONE SIMULATIONS OF NONMONOTONE NETWORKS

Paul E. DUNNE

Department of Computer Science, University of Liverpool, U.K. L69 3BX

Communicated by M.S. Paterson
Received April 1986
Revised March 1988

Abstract. We consider the following problem: given some n-argument monotone Boolean function, $f(X_n)$, with formal arguments $X_n = \{x_1, \ldots, x_n\}$, compute f using the $2n+1$ inputs $X_n \cup \{f_0, f_1, \ldots, f_n\}$. Here f_k is the *k-slice* of f, i.e. the n-argument monotone Boolean function $f_k(X_n) = (f \wedge T_k^n) \vee T_{k+1}^n$, where T_k^n is the n-argument monotone Boolean function which takes the value 1 iff at least k of its arguments are 1.

It is easy to see that if nonmonotone operations are permitted then $O(n)$ gates are sufficient by using the relation $f = \bigvee_{k=0}^n (f_k \wedge \overline{T_{k+1}^n})$. The properties of slice functions imply that efficient monotone solutions would allow superlinear lower bounds on the combinational complexity of f to be obtained from large enough lower bounds on the monotone complexity of f. Since negation is known to be superpolynomially powerful, some monotone functions must have superpolynomial complexity even if all the slice functions are given as extra inputs. However it is possible that efficient simulations, using slice functions, exist for restricted classes of monotone functions.

In this paper we examine a broad class of monotone Boolean functions, proving that for almost all of the functions in the class, no such simulation exists, and that in a very weak sense negation is exponentially powerful. In contrast to this an example of an efficient construction is given, again for a natural class of monotone Boolean functions.

1. Introduction

Although it has long been known that "almost all"[1] n-argument Boolean functions require exponentially many gates to be computed [10], the best lower bounds proved to date on the combinational complexity of explicitly defined functions are linear [4]. The difficulty of proving large lower bounds on the size of circuits which permit arbitrary 2-input Boolean functions as gate operations, has led to the consideration of more restricted types of Boolean networks. Probably the most widely studied of these is the class of monotone networks, in which only 2-input AND ($\wedge$) and OR ($\vee$) gates are permitted. Such networks compute exactly the class of monotone Boolean functions. Using this model there has been some success in obtaining good lower bounds on the size of networks computing sets of functions (e.g. [13]). In

[1] A property Π holds for "almost all" n-argument Boolean functions if the fraction of n-argument functions which do not have property Π tends to 0 with n.

0304-3975/89/$3.50

fact exponential lower bounds, on the complexity of one output functions, have been proved by Andreev [2], and Alon and Boppanna [1] for this model. The results of [1] are based on earlier, superpolynomial lower bounds of Razborov [7, 8], which show that negation is superpolynomially powerful for computing some monotone Boolean functions.

Unfortunately none of these results implies superlinear lower bounds on the combinational complexity of any function or set of functions.

Recent work of Berkowitz [3], Wegener [14, 15] and Dunne [5, 6] has considered the problem of relating the combinational and monotone network complexity of monotone Boolean functions via slice functions.

Berkowitz [3] established that the combinational and monotone network complexities of any k-slice function differed by at most a multiplicative constant and an additive term of $O(n \log^2 n)$. (This term is an improvement of Berkowitz' original construction which was independently obtained by Valiant [12] and Wegener [14].) Since f_k is easy to compute given f, this result establishes that any lower bound of $h(n) = \omega(n \log^2 n)$ on the *monotone* network complexity of a k-slice of f would imply that f had combinational complexity $\Omega(h(n))$. In addition the fact that f is easily computable, given its $n+1$ different slice functions (cf. Abstract above) implies that if f is "hard" then some slice of f must have large monotone complexity. Wegener [14] and Dunne [6] showed that the "canonical" slice of certain NP-complete predicates has polynomial complexity. The present author [5, 6] proved that for these same predicates (HAMILTONIAN CIRCUIT, CLIQUES and SAT) the $\frac{1}{2}n$-slice, (called "central" slice in [5, 6]) was also NP-complete and no easier than the underlying NP-complete function. In [14] Wegener introduced a set-theoretic interpretation of monotone networks computing slice functions and in [15] constructed new classes of monotone functions for which smaller lower bounds on monotone network complexity would be sufficient to deduce superlinear lower bounds on combinational network size.

None of these results yields superlinear lower bounds on combinational complexity. However if good methods existed for computing f from its slice functions (using only monotone operations) then such lower bounds could be derived from lower bounds on the monotone complexity of f, i.e. without using slice functions directly. Although no efficient method can exist for all monotone functions, one can still consider such simulations for special classes. In this paper we examine the following class of monotone functions:

Definition 1. Let Δ_r denote an arbitrary partition of X_n into r nonempty sets, $X^{(1)}, \ldots, X^{(r)}$ of sizes $n_1, \ldots, n_r$. Let α denote an r-tuple $\langle a_1, \ldots, a_r \rangle$ where $1 \leq a_i < n_i$. PART(Δ_r, α) is the set of monotone Boolean functions such that for every prime implicant p of $f \in$ PART(Δ_r, α), p contains exactly a_i variables from $X^{(i)}$. For the case $r = 1$ and $a_1 = a$ we denote the class of functions PART(Δ_r, a) by $Q_{n,a}$. These are the functions $f \in M_n$ such that every prime implicant of f contains exactly a variables.

Where there is no risk of ambiguity we will dispense with the dependence on Δ_r and α, using instead PART(n).

The main result of this paper concerns the following complexity measure: let $C^{m^*}(f)$ denote the monotone network complexity of f when all slice functions of f are given free as extra inputs. We consider this measure for functions in PART(Δ_r, α), and show that almost all functions, f in this class have $C^{m^*}(f) = \Omega(G(\Delta_r, \alpha))$, where G depends on the partition Δ_r and on α. Consequently it is proved, that with certain choices of Δ_r and α, for almost all such functions f, $C^{m^*}(f)$ is asymptotically equal to $C^m(f)$. In contrast to this negative result, it is proved that for any constant k and all $f \in Q_{n,n-k}$ we have $C^{m^*}(f) = O(n^{k-1}/\log n)$, whereas $C(f) = \Omega(n^k/\log n)$ for almost all functions in this class.

Also considered are other methods of reconstructing f from its slice functions, namely the use of "monotone projections" in the sense of Skyum and Valiant [11].

The remainder of this paper is organised as follows. In the next section it is proved that, in general, a monotone Boolean function is not a projection of any slice function, while in Section 3 the simulation for the class $Q_{n,n-k}$, mentioned above, is given. Section 4 proves the main result.

Notation

M_n = the set of all n-argument monotone Boolean functions,

$f_k(X_n)$ = k-slice of some function $f(X_n)$ in M_n,

a = $\sum_{i=1}^{r} a_i$,

$C(f)$ = combinational complexity of f,

$C^m(f)$ = monotone network complexity of f.

For a monom m over X_n, var(m) denotes the set of variables in X_n upon which m essentially depends, i.e. $\{x \in X_n : m \leq x\}$. The *dual function* of $f(X_n)$, denoted $\tilde{f}$, is the function

$$\neg f(\neg x_1, \neg x_2, \ldots, \neg x_n).$$

If $f \in M_n$, it is easy to show, from De Morgan's Laws, that $\tilde{f} \in M_n$ and $C^m(\tilde{f}) = C^m(f)$.

All logarithms are to the base 2 unless stated otherwise.

2. A negative result on monotone projections

Definition 2. Let $f(X_n)$ and $g(Y_p)$ be n-input and p-input monotone Boolean functions ($p \geq n$). f is a *monotone projection* of g iff there is a mapping $\sigma : Y_p \to \{X_n, 0, 1\}$ such that $f(X_n) = g(\sigma(Y_p))$.

The result in [6] that the $\frac{1}{2}n$-slices of some NP-complete functions are also NP-complete is obtained by giving a (nonmonotone) projection from this slice function to f. Thus, for certain functions, projections offer an alternative method of computing f from a slice function. The following result establishes that, in general, monotone projections from k-slices to f do not exist.

Lemma 1. *Let f be a member of M_n which depends on all its arguments, and let* $\mathbf{PI}(f)$ *denote the set of prime implicants of f. If there exist $q_1 = m_1 a$, $q_2 = m_2 b$ in* $\mathbf{PI}(f)$ *and c in $X_n - \{a, b\}$ such that $m_1 b \not\leqslant f$ and $m_2 ac \not\leqslant f$ then f is not a monotone projection of any k-slice of any $g(Y_p)$.*

Proof. Suppose that g, σ and k exist such that $f(X_n)$ is a monotone projection of the k-slice of $g(Y_p)$ using $\sigma: Y_p \to \{X_n, 0, 1\}$. Define for each $x \in X_n$, the *weight* of x under σ, denoted $w(x)$, as the number of arguments of Y_p which are projected onto x under σ. Clearly, since f depends on all its arguments, $w(x) \geqslant 1$ for each $x \in X_n$. For any assignment π to X_n the *contribution* of Y_p under σ, denoted $K(Y_p, \pi)$ is given by

$$|\{y \in Y_p : \sigma(y)^{|\pi} = 1\}|.$$

Clearly, for any assignment π which renders f equal to 1, it must hold that $K(Y_p, \pi) \geqslant k$ since it has been assumed that f is a projection of some k-slice function. Consider the assignment α which sets exactly the variables in q_1 to 1. From the preceding remark $K(Y_p, \alpha) \geqslant k$ since $f^{|\alpha} = 1$. By the same reasoning under the assignment β, which sets exactly the variables in q_2 to 1, $K(Y_p, \beta) \geqslant k$ also. Now let γ be that assignment which fixes only the variables in $m_2 \cup \{a, c\}$ to 1. Since f is now 0 it must be the case that $K(Y_p, \gamma) \leqslant k$ and as $w(c) \geqslant 1$, it follows that $w(b) > w(a)$. However, by applying the same argument to the variables of $m_1 \cup \{b\}$ we obtain $w(b) \leqslant w(a)$. This contradiction proves the lemma. □

3. An efficient simulation for the class $Q_{n,n-k}$

In this section we construct an efficient simulation for one class of monotone Boolean functions. The following results are required.

Fact 1 (Shannon [10]). *Let H be a subset of M_n. Then, for almost all $h \in H$,*

$$C(h) = \Omega\left(\frac{\log |H|}{\log \log |H|}\right).$$

Fact 2. *Let $k \geqslant 1$ be constant and $Q^m_{n,k}$ be the set of n-input m-output monotone Boolean functions such that for each $F = \langle f^1, \ldots, f^m \rangle$ in $Q^m_{n,k}$, $f_j \in Q_{n,k}$ for all $1 \leqslant j \leqslant m$. Let $C^m(Q^m_{n,k})$ denote*

$$\max\{C^m(F) : F \in Q^n_{n,k}\}.$$

Then,

(i) $C^m(Q^n_{n,1}) = O(n^2/\log n)$,

(ii) $C^m(Q^n_{n,k}) \leq nC^m(Q_{n,k})$,

(iii) $C^m(Q_{n,k}) \leq C^m(Q^n_{n,k-1}) + 2n - 1$,

(iv) *for* $k \geq 2$, $C^m(Q_{n,k}) = O(n^k/\log n)$.

Proof. (i) has been proved by Savage [9]; (ii) is obvious; a proof of (iii) is given in [16, p. 108] and (iv) is immediate from (i)–(iii). □

Lemma 2. *For almost all* $f \in Q_{n,n-k}$,

$$C(f) = \Omega(n^k/\log n).$$

Proof. The number of distinct monoms of size $n-k$ over X_n is $\Omega(n^k)$, since k is fixed. Thus, $|Q_{n,n-k}| = 2^{\Omega(n^k)}$ and the lemma follows from Fact 1. □

Theorem 1. *Let* $\gamma(n,k) = \max\{n, n^{k-1}/\log n\}$. *For all constant* $k \geq 2$, $\forall f \in Q_{n,n-k}$, $C^{m*}(f) = O(\gamma(n,k))$.

Proof. Observe that $f = f \wedge T^n_{n-k} = (f \wedge T^n_{n-k}) \vee T^n_n$. So it is sufficient to prove that $\forall q\ 2 \leq q \leq k$,

$$C^m(f \vee T^n_{n-k+q}) \leq C^m(f \vee T^n_{n-k+q-1}) + O(\gamma(n,k)).$$

Let S_{q-1} be an optimal network computing $(f \wedge T^n_{n-k}) \vee T^n_{n-k+q-1}$. We may express the function computed by S_{q-1} as

$$(f \wedge T^n_{n-k}) \vee p_1 \vee p_2 \vee \cdots \vee p_t,$$

where, for all p_i, $p_i \not\leq f \wedge T^n_{n-k} = f$.

For any product p let $\chi(p)$ be the disjunction over all variables in X_n that do not occur in p. We claim that for all $m \in \mathbf{PI}(f)$, and for all p_i, $m \leq \chi(p_i)$. To see this, recall that $p_i \not\leq m$, thus there exists some $x \in X_n$ such that $m \leq x \leq \chi(p_i)$. S_q is the network which computes

$$((f \wedge T^n_{n-k}) \vee T^n_{n-k+q-1}) \wedge \bigwedge_{i=1}^{t} \chi(p_i)$$

which evaluates to $f \wedge T^n_{n-k} \vee T^n_{n-k+q}$.

Let $g = \bigwedge_{i=1}^{t} \chi(p_i)$. Each $\chi(p_i)$ defines a prime clause of g and hence every such clause of g contains exactly $n-(n-k+q-1) = k-q+1 \leq k-1$ variables. So $\tilde{g} \in Q_{n,k-1}$ and if $k \geq 3$, from Fact 2(iv) it follows that

$$C^m(S_q) \leq C^m(S_{q-1}) + O(n^{k-1}/\log n).$$

If $k = 2$ then g is just a product of at most n variables and so in this case, $C^m(S_q) \leq C^m(S_{q-1}) + n$. By repeatedly applying this construction to f_{n-k} which, we recall, is given free, we obtain after $k-1$ iterations a network computing f, which has size $O(\gamma(n,k))$. □

Corollary 1. *Let $k \geq 3$ be fixed and*

$$\mathrm{HARD}_k = \{f \in Q_{n,n-k}\colon C^{\mathrm{m}}(f) = \omega(n^{k-1}/\log n)\}.$$

$\mathrm{HARD}_k \neq \emptyset$ *and for any* $f \in \mathrm{HARD}_k$ *it holds that* $C(f) = \Omega(C^{\mathrm{m}}(f))$.

Proof.[2] That HARD_k is nonempty is immediate from Lemma 2. For the second part let f be any function in HARD_k. From Theorem 1(ii),

$$\begin{aligned} C^{\mathrm{m}}(f) &\leq C(f_{n-k}) + \mathrm{O}(n^{k-1}/\log n) \\ &\leq C(f) + \mathrm{O}(n) + \mathrm{O}(n^{k-1}/\log n) \\ &= C(f) + \mathrm{O}(n^{k-1}/\log n) \end{aligned}$$

and the result follows from the choice of f. $\square$

Corollary 2. $\forall f \in Q_{n,n-2}$, $C^{\mathrm{m}}(f) = \Theta(C(f))$.

Proof. From Lemma 2 the set of functions in $Q_{n,n-2}$ with superlinear monotone complexity is nonempty. If f is any such function then the result follows by the same argument used to prove Corollary 1. $\square$

4. Main result

Definition 3. Let $Y_n = \{y_0, \ldots, y_n\}$ be a set of $n+1$ Boolean variables disjoint from X_n. A *reconstruction function* for $f(X_n)$ is any $(2n+1)$-input monotone Boolean function $h(X_n, Y_n)$ such that $h(X_n, f_0, \ldots, f_n) = f(X_n)$. h is said to *cover* f.

Below, $d(n)$ and $g(n)$ are functions from N to N. Let $H = \langle H_1, H_2, \ldots, H_n, \ldots, \rangle$ where H_n is a set of $d(n)$ $(2n+1)$-input monotone Boolean functions over X_n, Y_n. H_n *covers* $J_n \subseteq M_n$ iff each $f \in J_n$ is covered by some member of H_n. H *covers* $J = \bigcup_{n=0}^{\infty} J_n \subseteq \bigcup_{n=0}^{\infty} M_n$ iff, for all n, H_n covers J_n. If every $h \in H_n$ has $C^{\mathrm{m}}(h) \leq g(n)$ then H is a *g-cover* for J. Note that a g-cover exists for J iff each $f \in J_n$ has $C^{\mathrm{m}*}(f) \leq g(n)$.

Before proving the main result we establish three preliminary lemmas.

Lemma 3. *Δ_r and α are as in Definition* 1. *Let $\{p_1, \ldots, p_t\}$ be a set of t products satisfying the constraints on the prime implicants of functions in* PART(n). *There are exactly*

$$2^{\prod_{i=1}^{r} \binom{n_i}{a_i} - t}$$

functions in PART(n) *which have* all *the p_i as prime implicants.*

Proof. Obvious. $\square$

[2] If HARD_k were empty then the second part of the corollary would be trivially true. Consider a statement such as "$\forall f \in M_n$ $C^{\mathrm{m}}(f) \geq 2^n \Rightarrow C(f) \geq 2^n$". This is true since no $f \in M_n$ has $C^{\mathrm{m}}(f) \geq 2^n$.

Note: To avoid unwieldy expressions, $P(n)$ will denote $\prod_{i=1}^{r} \binom{n_i}{a_i}$.

Lemma 4. *Let $m = z_1 z_2 \ldots z_b$ where $b = a_j + 1$ and $\{z_1, \ldots, z_b\} \subseteq X^{(j)}$. Suppose that $\{p_1, \ldots, p_s\}$ is a set of s products each containing $a - a_j$ variables and such that for all $i \neq j$ each of these products contains exactly a_i variables from $X^{(i)}$. Note that s is at most $\prod_{1 \leqslant i \neq j \leqslant r} \binom{n_i}{a_i}$. If $r = 1$ then $s = 1$.*

There are exactly $(2^{P(n)-bs})(2^b - 1)^s$ functions $f \in \mathrm{PART}(n)$ such that $\bigvee_{i=1}^{s} mp_i \leqslant f$.

Proof. Let $q_i = m \wedge p_i$ for each $1 \leqslant i \leqslant s$. By considering the product obtained after setting any z_k to 1, it is easily seen that each q_i gives rise to b possible prime implicants. The number of functions which contain none of the bs distinct prime implicants arising from all the q_i, is $2^{P(n)-bs}$. For each of these functions there are $(2^b - 1)^s$ ways of extending the set of prime implicants so that each q_i is an implicant of the new function. (For each q_i some non-empty subset of the b possible products must be added to the set of prime implicants.) Multiplying these two factors gives the expression in the lemma. □

Lemma 5. *For any g-cover H, $|H_n| \leqslant 2^{O(g(n) \log g(n))}$.*

Proof. If, for all constants c, there exists an arbitrarily large n such that $|H_n| > 2^{c\, g(n) \log g(n)}$ then Fact 1 implies that one of the functions in H_n has monotone complexity $\omega(g(n))$ and so H could not be a g-cover. □

Fact 3. *Let $m : \mathrm{N} \to \mathrm{N}$. If $s > 2^b (\log_e 2) m(n)$ then*

$$(2^{P(n)-bs})(2^b - 1)^s < 2^{P(n)-m(n)}.$$

Proof. The inequality in the lemma holds if $2^{m(n)} < \exp(s/2^b)$. This is true if and only if $s > 2^b (\log_e 2) m(n)$ as required. □

Our main result is the following theorem.

Theorem 2. *Let $r \geqslant 3$ or $r = 2$ and $a_i \leqslant n_i - 2$ (for $i = 1$ and $i = 2$). Without loss of generality suppose that $b = a_1 \leqslant a_2 \leqslant \cdots \leqslant a_r$. If H is a g-cover for* PART *then*

$$g(n) = \Omega\left(\frac{P(n)}{2^b \binom{n_1}{b} \log P(n)}\right).$$

Proof. Suppose that H is a *g-cover* for PART, with H_n in H consisting of a set of $d(n)$ reconstruction functions which cover all the functions in $\mathrm{PART}(n)$. From Lemma 5 it follows that $d(n) \leqslant 2^{l(n)}$ where $l(n) \leqslant c\, g(n) \log g(n)$ for some constant $c > 0$. Some function $h \in H_n$ must cover at least $|\mathrm{PART}(n)|/d(n)$ different functions from $\mathrm{PART}(n)$.

Let D denote the subset of PART(n) covered by this h. So by the previous argument $|D|$ is at least $2^{P(n)-l(n)}$. Now consider any set $\{p_1, \ldots, p_t\}$ of products over X_n, each product being as in Lemma 3. From Lemma 3 it follows that if $t > c\,g(n) \log g(n)$, then some function in D cannot have all of these products as prime implicants.

Similarly, consider any set $\{q_1, \ldots, q_s\}$ of products over X_n, each product being as in Lemma 4 with $j = 1$. Again from Lemma 4, by using Fact 3, it follows that if $s > 2^b(c \log_e 2) g(n) \log g(n)$ then some function in D does not have all of these s products as implicants.

We can now examine the structure of h in greater detail. h computes some $(2n+1)$-input monotone Boolean function of X_n, Y_n. Each prime implicant of this function consists of some product, m say, over X_n which is $\bigwedge$'ed with some product over Y_n. When the ith Y_n input is replaced by the i-slice of some function f in D then this product of y's reduces to a single slice function, $f_k(X_n)$ say. Thus the "prime implicant" reduces to a function, w, of the form

$$m \wedge f_k = m \wedge ((f \wedge T_k^n) \vee T_{k+1}^n).$$

(From here on we drop the explicit dependence on X_n.)

Note that the product m is present regardless of f since we are considering a single monotone Boolean function $h(X_n, Y_n)$.

Now since only functions in PART(n) are of interest, we may assume that $k \leq a$ and that each m contains at most a_i variables from $X^{(i)}$. With these assumptions we claim that any product m, as above, contains at least a_i variables from each class $X^{(i)}$. To see this, suppose that there is some m containing fewer than a_i variables from $X^{(i)}$. Note that m therefore depends on at most $a-1$ variables. Consider the following three cases:

(I) $|\mathrm{var}(m)| > k$: then $m \wedge T_{k+1}^n = m$ and so m would be an implicant of every function in D. This is a contradiction since no function in PART(n) has m as an implicant.

(II) $|\mathrm{var}(m)| < k$: construct a product p of exactly $k-1$ variables to satisfy the following:

(i) $\mathrm{var}(m) \subseteq \mathrm{var}(p)$.

(ii) For each $j \neq i$, p contains at most a_j variables of $X^{(j)}$.

(iii) p contains at most $a_i - 1$ variables from $X^{(i)}$.

Thus $m \wedge p \equiv p$.

The conditions on r, a_j and n_j in the theorem guarantee that the set $X_n - X^{(i)} - \mathrm{var}(p)$ contains at least two variables, y and z say. $p \wedge y \wedge z$ is a product of $k+1$ variables and so is a prime implicant of T_{k+1}^n. Additionally

$$m \wedge p \wedge y \wedge z \equiv p \wedge y \wedge z.$$

These two facts imply that $p \wedge y \wedge z$ is an implicant of every function in D. This is a contradiction: $p \wedge y \wedge z$ is not an implicant of any function in PART(n) since it contains only $a_i - 1$ variables from $X^{(i)}$.

(III) $|\mathrm{var}(\boldsymbol{m})| = k$: This is similar to Case (II), for by the same argument we can identify at least one variable in $\boldsymbol{X}_n - \boldsymbol{X}^{(i)} - \mathrm{var}(\boldsymbol{m})$, y say, and so appeal to the reasoning concluding Case (II) with regard to the monom $\boldsymbol{m} \wedge y$.

Therefore we can further assume that the product $\boldsymbol{m}$ contains exactly a_i variables from each class of the partition of $\boldsymbol{X}_n$.

Note: Without the restriction on Δ_r and α in the theorem, this assumption is not valid.

If $k = a$ then this function w simplifies to

$$\boldsymbol{m} \wedge (f \vee T^n_{a+1}). \tag{A}$$

We call the product $\boldsymbol{m}$ occurring in an expression having the form of (A) a *Type (A) term*. $\boldsymbol{m}_A$ will denote an arbitrary Type (A) term. Note that these depend solely on h and so are independent of f. Additionally, since we have assumed that any $\boldsymbol{m}_A$ contains exactly a_i variables from $\boldsymbol{X}^{(i)}$, it follows that different Type (A) terms contribute different prime implicants to any $f \in D$.

If $k < a$ then w simplifies to

$$\boldsymbol{m} \wedge T^n_{k+1} \equiv \boldsymbol{m}. \tag{B}$$

Such products $\boldsymbol{m}$ will be referred to as *Type (B) terms*, $\boldsymbol{m}_B$ denoting an arbitrary such term.

From the consequence of Lemma 3, stated earlier, the number of Type (B) terms in h is at most $c\,g(n) \log g(n)$. This is because any Type (B) term is a prime implicant of each function in D.

Consider the function $\mathrm{MAX} \in \mathrm{PART}(n)$ which is defined by

$$\mathrm{MAX} = \bigvee_{\text{Type (A) terms}} \boldsymbol{m}_A \vee \bigvee_{\text{Type (B) terms}} \boldsymbol{m}_B.$$

Clearly MAX is in D and by the preceding arguments on the structure of h, it must be the case that, for each $f \in D$,

$$\bigvee_{\text{Type (A) terms}} \boldsymbol{m}_A \wedge (f \vee T^n_{a+1}) \vee \bigvee_{\text{Type (B) terms}} \boldsymbol{m}_B = f.$$

This may be rewritten as

$$\left(\bigvee_{\text{Type (A) terms}} \boldsymbol{m}_A \vee \bigvee_{\text{Type (B) terms}} \boldsymbol{m}_B \right) \wedge (f \vee T^n_{a+1}) = f$$

since each Type (B) term is a prime implicant of every $f \in D$.

Therefore $\mathrm{MAX} \wedge (f \vee T^n_{a+1}) = f$. $\mathbf{PI}(f)$ is a subset of $\mathbf{PI}(\mathrm{MAX})$ (which includes all the $\boldsymbol{m}_B$) and $\mathrm{MAX} \wedge T^n_{a+1} \leq f$. The lower bound on the size of $|D|$ implies that MAX has at least $P(n) = c\,g(n) \log g(n)$ prime implicants, for otherwise not enough subsets can be formed. Recall that the function $\chi(p)$ is the disjunction of all variables in $\boldsymbol{X}_n$ which do not occur in p. Then for each $f \in D$ we have

$$\mathrm{MAX} \wedge T^n_{a+1} = \bigvee_{p \in \mathbf{PI}(\mathrm{MAX})} p \wedge \chi(p) \leq f.$$

Thus there are at least $|\mathbf{PI}(\mathrm{MAX})|\binom{n_i}{b}^{-1}$ implicants of length $a+1$ common to each $f \in D$, which satisfy the conditions of Lemma 4. From the upper bound on the maximum number of such implicants we obtain

$$P(n) - c\,g(n)\log g(n) \leq 2^b (c \log_e 2)\binom{n_i}{b} g(n) \log g(n),$$

$$g(n)\log g(n) \geq \frac{P(n)}{(c \log_e 2) 2^b \binom{n_i}{b} + 1}.$$

Using the approximation $G \log G \geq F$ implies $G \geq F/\log F$ proves the theorem. □

The result of Theorem 2 is easily seen to be expressible as given in Corollary 3.

Corollary 3. *For almost all* $f \in \mathrm{PART}(\Delta_r, \alpha)$,

$$C^{\mathrm{m}^*}(f) = \Omega\left(\frac{P(n)}{2^b \binom{n_i}{b} \log P(n)}\right).$$

Of more interest are the following special cases of Theorem 2.

Corollary 4. *If* a_i, n_i *are constant for some* $1 \leq i \leq r$ *then for almost all* $f \in \mathrm{PART}(\Delta_r, \alpha)$,

$$C^{\mathrm{m}^*}(f) = \Omega\left(\frac{P(n)}{\log P(n)}\right).$$

Note that in this case the counting argument of Shannon gives $C^{\mathrm{m}}(f) = \Omega(P(n)/\log P(n))$. So for this class of functions, providing the slice functions as extra inputs does not in general reduce monotone complexity.

The final corollary given deals with the case $r = 1$.

Corollary 5. *Let* $r = 1$ *and consider* $Q_{n,a} = \mathrm{PART}(\Delta_r, \alpha)$. *There exist functions* $f \in Q_{n,a}$ *for which*

$$C^{\mathrm{m}^*}(f) = \Omega\left(\frac{n^{a-1}}{\log n}\right) \qquad (a \text{ constant}),$$

$$C^{\mathrm{m}^*}(f) = \Omega\left(\frac{n^{a-1}}{a^a \log n}\right) \qquad (a = \mathrm{o}(n)),$$

$$C^{\mathrm{m}^*}(f) = \Omega\left(\frac{n^{k-1}}{\log n}\right) \qquad (a = n - k,\ k \text{ constant}).$$

Proof. Since $Q_{n,a}$ is a superset of $\mathrm{PART}(n)$ for certain partitions, it is sufficient to prove the result for a particular $\mathrm{PART}(\Delta_r, \alpha)$ in each case.

For the first two relations partition X_n into n/a sets of roughly equal size, and set $a_i = 1$ for each i. Applying Theorem 2 gives the results claimed.

For the final relation, partition X_n into two sets: one of size $2k$ and one of size $n - 2k$, then set $a_1 = 1$ and $a_2 = k - 1$. Again apply Theorem 2 to yield the result. □

The result of Section 3 shows that the last relation of Corollary 5 is in fact the best possible, since there it was proved that $C^{m^*}(f) = O(n^{k-1}/\log n)$ for any function in this final class.

References

[1] N. Alon and R. Boppana, The monotone circuit complexity of Boolean functions, *Combinatorica*, to appear.

[2] A.E. Andreev, A method of proving lower bounds on the complexity of monotone Boolean functions, *Dokl. Akad. Nauk* **282** (1985) 1033-1037 (in Russian).

[3] S. Berkowitz, On some relationships between monotone and non-monotone circuit complexity, Technical Report, Univ. of Toronto (1982).

[4] N. Blum, A Boolean function requiring $3n$ network size, *Theoret. Comput. Sci.* **28** (1984) 337-345.

[5] P.E. Dunne, Techniques for the analysis of monotone Boolean networks, Ph.D. Dissertation, Univ. of Warwick, September 1984; Theory of Computation Report No. 69, Dept. of Computer Science, Univ. of Warwick.

[6] P.E. Dunne, The complexity of central slice functions, *Theoret. Comput. Sci.* **44** (1986) 247-257.

[7] A.A. Razborov, Lower bounds on the monotone complexity of some Boolean functions, *Dokl. Nauk* **281**(4) (1985) 798-801 (in Russian).

[8] A.A. Razborov, A lower bound on the monotone complexity of the logical permanent, *Matematischi Zametki* **37**(6) (1985) 887-901 (in Russian).

[9] J.E. Savage, An algorithm for the computation of linear forms, *SIAM J. Comput.* **3** (1974) 150-158.

[10] C.E. Shannon, The synthesis of two-terminal switching networks, *Bell Systems Tech. J.* **28** (1949) 59-98.

[11] S. Skyum and L.G. Valiant, A complexity theory based on Boolean algebra, *J. ACM* **32** (1985) 484-502.

[12] L.G. Valiant, Negation is powerless for Boolean slice functions, *SIAM J. Comput.* **15** (1986) 531-535.

[13] I. Wegener, Boolean functions whose monotone complexity is of size $n^2/\log n$, *Theoret. Comput. Sci.* **21** (1982) 213-224.

[14] I. Wegener, On the complexity of slice functions, *Theoret. Comput. Sci.* **38** (1985) 55-68.

[15] I. Wegener, More on the complexity of slice functions, *Theoret. Comput. Sci.* **43** (1986) 201-211.

[16] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner Series in Computer Science (Wiley-Teubner, Stuttgart, 1987).