Techniques For The Analysis Of Monotone Boolean Networks

Paul Edward Dunne

A dissertation submitted for the degree of
Doctor of Philosophy

University of Warwick
Department of Computer Science

September 1984

CONTENTS

Table Of Figures

Acknowledgements

I am indebted to Professor Michael Paterson, my supervisor, for his continual help and guidance throughout the course of this research. I would also like to thank Dr. William McColl for stimulating my interest in this area of research and for a number of valuable discussions, as well as Dr. Tony Cohn for his assistance in using the Text Formatting software.

This research was supported by the Science and Engineering Research Council of Great Britain.

Declaration

This dissertation is the result of research carried out in the Department of Computer Science at Warwick University between October 1981 and August 1984. A substantial part of the material of Chapter(3) has appeared as Theory of Computation Report No.62, (January 1984). The material of Chapter(5) and the first part of Chapter(6) formed the basis of Theory of Computation Report No.64 (March 1984). Both these reports were produced in the Department of Computer Science.

DEDICATION

For My Mother and Father

Gedachte man ir ze guote niht
von den der werlde guot geschiht
so waerez allez alse niht
swaz guotes in der werlde geschiht

Gottfried von Strassburg

Tristan, Prologue i–iv

SUMMARY

Monotone boolean networks are one of the most widely studied restricted forms of combinational networks. This dissertation examines the complexity of such networks realising single output monotone boolean functions and develops recent results on their relation to unrestricted networks. Two standard analytic techniques are considered: the inductive gate elimination argument, and replacement rules.

In Chapters(3) and (4) the former method is applied to obtain new lower bounds on the monotone network complexity of threshold functions. In Chapter(5) a complete characterisation of all replacement rules, valid when computing some monotone boolean function, is given. The latter half of the dissertation concentrates on the relation between the combinational and monotone network complexity of monotone functions, and extends work of Berkowitz and Wegener on "slice functions". In Chapter(6) the concept of "pseudo-complementation", the replacement of instances of negated variables by monotone functions, without affecting computational behaviour, is defined. Pseudo-complements are shown to exist for all monotone boolean functions and using these a generalisation of slice function is proposed. Chapter(7) examines the slice functions of some *NP*-complete predicates. For the predicates considered, it is shown that the "canonical" slice has polynomial network complexity, and that the "central" slice is also *NP*-complete. This result permits a reformulation of the $P \neq NP$? question in terms of monotone network complexity. Finally Chapter(8) examines the existence of gaps for the combinational and monotone network complexity measures. A natural series of classes of monotone boolean functions is defined and it is shown that for the "hardest" members of each class there is no asymptotic gap between these measures.

Chapter 1

Preliminaries

In this chapter we present basic definitions and the notation which will be used below. Readers already familiar with boolean network complexity may ignore the first two sections of this chapter.

1.1) Boolean Functions

Let $X_n = \{x_1,...,x_n\}$ be a set of n boolean variables. Any function $f(X_n):\{0,1\}^n \to \{0,1\}$ is called a *single output n input boolean function*. Functions $f(X_n):\{0,1\}\to\{0,1\}^m$ are called *m output n input boolean functions*, or simply multiple output boolean functions.

B_n will denote the set of all single output n input boolean functions. $B_{n,m}$ will denote the set of all m output, n input boolean functions.

Below, unless otherwise stated, "boolean function" will mean "single output boolean function".

Order relations, "$\leq$", "$<$", are defined for $f,g \in B_n$ by:

$f \leq g \iff \forall\, \alpha \in \{0,1\}^n \quad f(\alpha)=1 \Rightarrow g(\alpha)=1$
$f < g \iff f \leq g$ and $f \neq g$

Let $f \in B_n$. f is *monotone* if and only if:

M1) $\forall\, x_i \in X_n$

$$f(x_1,x_2,...,x_{i-1},0,x_{i+1},...,x_n) \leq f(x_1,x_2,...,x_{i-1},1,x_{i+1},...,x_n)$$

M_n will denote the set of all n input monotone boolean functions.

$M_{n,m}$ the set of all m output, n input monotone boolean functions.

A *monom*, m, is a monotone boolean function of the form:

$$m = x_{i1} \wedge x_{i2} \wedge ... \wedge x_{ir} \quad \text{and} \quad x_{ij} \in X_n$$

where $\wedge$ denotes boolean conjunction.

Let $f \in M_n$. A monom m is an *implicant* of f if $m \leq f$. m is a *prime implicant* of f if:

PI1) $m \leq f$

PI2) $\forall$ *monoms* m^* such that $m < m^*$, m^* is not an implicant of f.

$var(m) = \{x \in X_n \mid m \leq x\}$
$PI(f) = \{m \mid m$ *is a prime implicant of* $f\}$

A *clause* c, is a monotone boolean function of the form:

$$c = x_{i1} \vee x_{i2} \vee ... \vee x_{ir} \quad \text{and} \quad x_{ij} \in X_n$$

where $\vee$ denotes boolean disjunction.

A clause c is an *implicand* of $f \in M_n$ if $f \leq c$. c is a *prime clause* of f if and only if:

PC1) $f \leq c$

PC2) $\forall$ clauses c^* such that $c^* < c$, c^* is not an implicand of f.

$var(c) = \{x \in X_n \mid x \leq c\}$
$PC(f) = \{c \mid c$ *is a prime clause of* $f\}$

It is well known that every $f \in M_n$ may be expressed as the disjunction of its prime implicants, called *Disjunctive Normal Form* (DNF), or as the conjunction of its prime clauses, called *Conjunctive Normal Form* (CNF).

Let $f(X_n) \in M_n$. The *dual* of f ($\hat{f}(X_n)$) is the monotone boolean function defined by:

$\hat{f}(X_n) = \neg f(\neg x_1,...,\neg x_n)$

where "$\neg$" denotes negation.

Let $f \in B_n$ and $g \in B_p$ where $p \geq n$. $f(X_n)$ is a *projection* of $g(Y)$ if $\exists$ a mapping $\sigma: Y \to \{X_n, \neg x_1, \neg x_2,..., \neg x_n, 0, 1\}$ such that:

$$f(X_n) = g(\sigma(y_1), \sigma(y_2),...,\sigma(y_p))$$

1.2) Boolean Networks

Let $\Omega \subseteq B_2$. A *boolean Ω-network* T is a directed acyclic graph consisting of two disjoint sets of nodes: I is the set of nodes having in-degree equal to 0 (the *inputs* of T). Each node of I is associated with some $x \in X_n$ or with some $\neg x \in \{\neg x_1, \ldots, \neg x_n\}$ (if negation is permitted). We shall assume that for every $x \in X_n$ there is at most one input node, associated with x and at most one input node associated with $\neg x$. G is the set of nodes having in-degree equal to 2 (the *gates* of T). Each gate g is associated with some boolean operator $b \in \Omega$ (denoted by op(g) = b or g is a b-gate).

The out-degree of a node u of T is called the *fanout* of u. Nodes of T with fanout equal to 0, are called the *outputs* of T. Ω is referred to as the *basis* of T.

Below we shall make the assumption, that unless otherwise stated, any network for a boolean function has a unique output node t. For the sake of brevity, we shall informally identify the set of variables X_n with the set of network inputs I, and thus refer to "the input x_i of T" instead of "the input of T associated with x_i" and to X_n as the inputs of T.

A *monotone boolean network*, S, is a boolean Ω-network, with only unnegated inputs available, for which $\Omega = \{\wedge, \vee\}$.

Let S be a monotone boolean network and let u be a node of S. $RES(u)$ is the monotone boolean function recursively defined by:

$$RES(u) = \begin{cases} x_i & \text{if u } \textit{is the input } x_i \textit{ of } S \\ RES(u_1) \wedge RES(u_2) & \text{if u } \textit{is an } \wedge\textit{-gate} \\ RES(u_1) \vee RES(u_2) & \text{if u } \textit{is an } \vee\textit{-gate} \end{cases}$$

where u_1, u_2 are the inputs of u if u is a gate.

S *realises* or computes $f \in M_n$ if and only if $RES(t) = f$ for the output t of S. It is well known that monotone boolean networks compute exactly the class of monotone boolean functions.

$RES(u)$ may be analogously defined for arbitrary boolean networks. An Ω-network T will be called a *combinational* or *unrestricted* network if $\Omega = B_2$.

A *partial assignment* π, is an assignment of boolean constants to some subset of $\{x_1, \ldots, x_n\}$. $|\pi|$ will denote the number of variables set by π. If $f \in M_n$, $f^{|\pi}$ will denote the monotone boolean function arising from the application of π to the inputs of f. $f^{|\pi} \in M_{n-|\pi|}$, and is sometimes called a *subfunction* of f. Similarly, for monotone boolean networks, $S^{|\pi}$ will denote the network S after applying π to the inputs of S.

Let S be a monotone boolean network computing some $f \in M_n$. The *monotone dual* of S, ($\hat{S}$) is the monotone network obtained by replacing each $\wedge$-gate in S by an $\vee$-gate and each $\vee$-gate in S by an $\wedge$-gate. It may be easily verified, from the definition of dual functionand De Morgan's Laws, that $\hat{S}$ computes $\hat{f}$.

1.3) Network Complexity

Let T be an Ω-network.

$$C_\Omega(T) = |\{g \mid g \text{ is a gate in } T\}|$$

Let $f \in B_n$

$$C_\Omega(f) = \min\{C_\Omega(T) \mid T \text{ is an } \Omega\text{-network realising } f\}$$

If $\Omega = B_2$ we shall write these quantities as C(T), C(f) respectively, the latter being called the *combinational complexity* of f.

If $f \in M_n$ and $\Omega = \{\wedge,\vee\}$ we shall refer to these measures as the *monotone network size* of S and the *monotone network complexity* of f, denoting these by $C^m(S)$ and $C^m(f)$. Clearly for $f \in M_n$:

$$C(f) \leq C^m(f)$$

$C(f_1,\ldots,f_m)$ and $C^m(f_1,\ldots,f_m)$ denote the corresponding quantities for $(f_1,\ldots,f_m)$ in $B_{n,m}$.

1.4) Specific Monotone Boolean Functions

Let $X_n = \{x_1,\ldots,x_n\}$. The *symmetric* boolean functions are those functions whose value depends only on the *number* of inputs which have the value 1.

The *k-th threshold function* (T_k^n) is the monotone boolean function defined by:

$$T_k^n(X_n) = \begin{cases} 1 & \text{if at least } k \text{ inputs have the value } 1 \\ 0 & \text{otherwise} \end{cases}$$

The threshold function $T^n_{\lceil n/2 \rceil}$ is called the *majority function* and is denoted $MAJ_n(X_n)$.

The threshold functions are the monotone symmetric boolean functions.

Let $X_n^U = \{x_{ij} \mid 1 \leq i < j \leq n\}$ be a set of $n(n-1)/2$ boolean variables. The n-vertex undirected graph $G(X_n^U)$ is defined as having an edge between vertices i and j if and only if $x_{ij} = 1$.

A *k-clique* is a complete graph on k vertices. A graph (undirected or directed) has a hamiltonian circuit if there is a simple cycle which contains every vertex.

$$(n/2)\text{-}clique(X_n^U) = \begin{cases} 1 & \text{if } G(X_n^U) \text{ contains an } (n/2)\text{-clique} \\ 0 & \text{otherwise} \end{cases}$$

Let $X_n^D = \{x_{ij} \mid 1 \leq i \leq j \leq n\}$ be a set of $n(n-1)$ boolean variables with an n-vertex *directed* graph $G(X_n^D)$, defined analogously.

$$DHC(X_n^D) = \begin{cases} 1 & \text{if } G(X_n^D) \text{ contains a directed hamiltonian circuit} \\ 0 & \text{otherwise} \end{cases}$$

$UHC(X_n^U)$ is defined similarly for the Undirected Hamiltonian Circuit predicate.

1.5) Notation

Let G be an n-vertex graph.

V(G) = *Set of vertices in* G
E(G) = *Set of edges in* G

For further graph-theoretic definitions see Berge[2] or Even[11]

Let $f,g : \mathbb{N} \to \mathbb{R}^+$

G1) $f(n) = O(g(n))$ if $\exists$ constants $c, K > 0$ such that:

$$f(n) \le c.g(n) \quad \forall n \ge K$$

G2) $f(n) = \Omega(g(n))$ if $\exists$ constants $c, K > 0$ such that:

$$f(n) \ge c.g(n) \quad \forall n \ge K$$

G3) $f(n) = \Theta(g(n))$ if:

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))$$

G4) $f(n) = o(g(n))$ if:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

G5) $f(n) = \omega(g(n))$ if:

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

Chapter 2

2.1) Introduction

In recent years the study of the complexity of realising boolean functions by boolean networks has been an increasingly active research area. Such networks have been shown to be a reasonable model of computation by Fischer and Pippenger [14], who demonstrated that any function computable by a deterministic Turing machine in T steps, could be realised by a boolean network containing $O(T \log T)$ gates.[1)]

In 1949, Shannon [45], proved that all but a vanishingly small fraction of the 2^{2^n} boolean functions in B_n have combinational complexity $\Omega(2^n/n)$. However, little is known about the difficulty of realising specific boolean functions. For networks which allow any of the 16 functions in B_2 as gate operations, the best known lower bounds on explicitly defined functions, are linear. The largest of these are the bounds of $2.5n$ by Paul [38] and $3n$ by Blum [7], both the functions used involving a concept of indirect addressing. Lower bounds of $2.5n$ for various symmetric functions have been derived by Stockmeyer.[46].

The difficulty of determining the complexity of unrestricted networks for specific boolean functions has led to the consideration of restricted forms of boolean network, in the hope that these may prove more amenable to analysis. With respect to combinational networks the aims of such models are twofold: to gain insight into proof techniques for arbitrary boolean networks via lower bound techniques for the restricted model; and to determine if such models may efficiently simulate unrestricted networks. One such special model, the monotone networks, will be the main object of study in this dissertation. Before

1) All logarithms in this dissertation are to the base 2

considering the monotone network model in more detail, we shall briefly survey some of the other forms of boolean network which have been examined in the literature.

2.2) Restricted Forms Of Boolean Networks

2.2.1) Formulas

Formulas are boolean networks in which gate nodes have fanout at most one. The measure of complexity is taken to be the number of gates employed, (or sometimes the total fanout from the input nodes, which is precisely one more). A number of general lower bound techniques have been derived for this case. The methods of Khrapchenko [19] apply only to the basis $\{\wedge, \vee, -\}$ and yield lower bounds of $\Omega(n^2)$ for the n-input parity functions, (i.e. those boolean functions whose result is determined by whether the number of inputs with value 1 is odd or even). These functions require only $n-1$ gates in formulas with basis B_2. Neciporuk [33] developed methods yielding lower bounds of $\Omega(n^2/\log n)$ on the size of unrestricted formulas, by considering the number of distinct subfunctions of a boolean function $f \in B_n$. Hodes and Specker [17] and Fischer, Meyer and Paterson [13] have also derived bounding methods for arbitrary formulas, by showing that boolean functions with "small" (e.g linear) formula size must satisfy certain conditions. The arguments of Hodes/Specker lead to $\Omega(n \log^* n)$ lower bounds. Those of Fischer, Meyer and Paterson yield lower bounds of $\Omega(n \log n)$. For some symmetric functions, such as T_k^n for fixed k, the methods of Hodes/Specker have recently been sharpened by Pudlak [39] to give lower bounds of $\Omega(n \log \log n)$.

Despite the closeness of asymptotic bounds for "almost all" formulas to similar bounds for networks, formulas have complexity $\Theta(2^n/(\log n))$ [40],[26], networks $\Theta(2^n/n)$ [45], [24], it is not known if formulas can efficiently simulate

combinational networks. Paul [38] has constructed functions for which an almost quadratic gap between formula and network size is provable. The best known simulations of networks by formulas give exponential increases in size. (Paterson and Valiant [34]).

2.2.2) Planar Networks

For planar networks the underlying undirected graph is required to be planar, the complexity of such a network being the number of gates it contains. An interesting feature of this model is that it may be related to the Thompson, Brent/Kung model of VLSI chip complexity [47], [8] as shown by Savage [41]. The known lower bound methods combine information flow arguments with the planar separator construction of Lipton & Tarjan [22]. In this way lower bounds of $\Omega(n^2)$ on boolean convolution [23] and matrix multiplication [41] have been obtained. These results do not translate into lower bounds on combinational networks, as the best known simulations require a quadratic increase in the number of gates used. McColl [27] has recently derived tight asymptotic lower bounds on the planar network complexity of "almost all" boolean functions.

2.2.3) Bounded-Depth Networks

In this model arbitrary fan-in $\wedge$-gates and $\vee$-gates are available as gate operations and negation may be applied to the network inputs only. The length of any path from an input node to the output gate is bounded above by some constant k.[2)] The complexity of a depth-k network is defined to be the total number of wires used. Such networks were introduced by Lupanov [25] as a generalisation of Conjunctive and Disjunctive Normal Form. Lupanov proved asymptotic upper and lower bounds for the depth-k complexity of "almost all"

2) An alternative model, in which arbitrary boolean functions are available as gate operations, has been considered by Chandra, Fortune and Lipton [9] for multiple-output functions. Slowly growing non-linear lower bounds are obtained for certain prefix computations.

boolean functions. There has been considerable recent interest in this model, arising from the result of Furst, Saxe and Sipser [16] that parity functions cannot be realised by polynomial size bounded depth networks. Fagin, Klawe, Pippenger & Stockmeyer have extended this result by characterising those symmetric functions which require superpolynomial depth-k network size [12]. Although these results do not translate into techniques for proving lower bounds on combinational network size, [16] shows that sufficiently large lower bounds in this model lead to important results concerning the separation of complexity classes. Program Logic Arrays (PLAs) are a method of implementing arbitrary boolean functions in VLSI chips (see [30] for details). [16] also states that this model has implications for the size of PLA's realising certain boolean functions.

Each of the network models described above employs some graph-theoretic restriction in attempting to account for the complexity of specific boolean functions. All of them are functionally complete, in the sense that all boolean functions are realisable in any of these models. Monotone networks, since they compute exactly the class of monotone boolean functions, clearly do not have this property. Despite this there are several reasons for considering this restriction. Many computationally interesting functions are actually monotone or have incarnations as monotone functions, examples of the latter being Multiplication since a special case of this is boolean Convolution, which is monotone. For other monotone algebraic computations, strong lower bound techniques have been derived yielding exponential lower bounds on specific functions (Schnorr [44], Jerrum & Snir [18], Lingas [21]). Finally, it has recently been shown that sufficiently large superlinear lower bounds on the monotone network complexity of certain classes of function, imply superlinear lower bounds on combinational network size [3]. We shall return to this last point below. We

note that restricted *monotone* networks of the forms described above are also definable. In particular exponential lower bounds for the depth-3 monotone complexity of $n/2$-*clique* have been obtained by Valiant [51] and for the majority function by Yao [58]. For the planar network model McColl has shown that certain monotone functions cannot be realised by networks which are both monotone and planar.[28]

In the following section we shall describe some of the known lower bounds on monotone network complexity and discuss the proof techniques applied.

2.3) Monotone Network Complexity

Although the techniques of Schnorr [44] for analysing the complexity of monotone arithmetic computation are not known to apply to monotone boolean networks (cf Wegener [53]), a number of strong lower bounds have been derived for several computationally interesting multiple output functions. Lamagna [20] proved lower bounds of $\Omega(n \log n)$ for sorting, merging and boolean convolution. Paterson [36] and Mehlhorn & Galil [32] obtained $\Omega(n^{3/2})$ lower bounds on $N \times N \times N$ matrix multiplication (where $n = N^2$). Paterson further demonstrated that there is essentially only one optimal monotone network structure for this function, namely to use N^3 $\wedge$-gates, each computing one prime implicant of one output function, and $N^3 - N^2$ $\vee$-gates to collect the appropriate prime implicants for each specific output. Superlinear lower bounds have also been obtained for certain sets of boolean sums ($\Omega(n^{3/2})$ Wegener [54], $\Omega(n^{5/3})$ Mehlhorn [31]). The $\Omega(n \log n)$ lower bound on boolean convolution has been improved by Blum [6] ($\Omega(n^{4/3})$ on the number of $\wedge$-gates) and Weiss [57] ($\Omega(n^{3/2})$ on the number of $\vee$-gates). To date, the largest lower bound is that of $\Omega(n^2/(\log n))$ due to Wegener, for a generalisation of matrix product [55].

Common to many of the proof techniques employed is the concept of applying some "replacement rule" in combination with an inductive argument, e.g [36],[32],[31],[54],[55],[57].

The basic form of the inductive argument is easily stated. Suppose that $\{f_{i_1},\dots f_{i_n},\dots\}$ is an infinite family of monotone boolean functions, $f_{i_n} \in M_{i_n}$. To prove a lower bound of $l(n)$ on the size of monotone networks computing f_n one proceeds by first showing that $\mathbf{C^m}(f_{i_1}) \geq l(i_1)$ (Inductive base), and then under the assumption that $\mathbf{C^m}(f_{i_j}) \geq l(i_j)$ for all $i_j < n$, proving that for every optimal monotone boolean network, S, computing f_n, there exists some partial assignment π with the properties that:

a1) $f_n|^{\pi} = f_{i_j}$ and $i_j < n$

a2) $S|^{\pi}$ contains q fewer gates than S.

(Under the application of π to the inputs of S, some gates become redundant because, for example, they have constant functions as inputs.)

Now if $l(i_j) + q \geq l(n)$ the desired lower bound follows by induction on n.

The strength of this form of argument is limited in two ways. Clearly to prove even modest *linear* lower bounds it must be proved that enough gates may be eliminated, and this quantity is determined by the partial assignment used. Frequently the choice of partial assignment is limited by the need to project onto a smaller instance in the family, e.g for T_k^n with k fixed, only partial assignments which set inputs to 0 are applicable. Where this method has been applied to combinational networks, sophisticated arguments have been employed to remove the cases where insufficient gates can be directly eliminated. Thus, often it happens that the inductive step cannot be made without some knowledge of the structure of optimal networks. *Replacement rules* introduced by Paterson [36] and Mehlhorn & Galil [32], are one method of gleaning such information. A replacement rule for $f \in M_n$ is a rule of the form:

In any monotone network realising f, any node computing the function g

may be replaced by a node computing the function h, and the resulting network will still compute f.

Now, if one wishes to show that an optimal monotone network for f, does not contain any node computing some function g, it is sufficient to show that g may be replaced by a constant function or by an input of f. Applications in this "pure" form are not always possible, but by assuming that some functions are available as additional network inputs, similar deductions may be made. Such an approach has been used by Wegener in [54], [55].

The complexity of monotone networks realising single output functions has not been widely examined. The largest lower bound known is that of $4n$ for the function, $(z \wedge T_2^n) \vee T_{n-1}^n$, by Tiekenheinrich [48]. In Chapter(3), below, we shall prove a new lower bound on T_k^n for fixed k, and in Chapter(4) slightly improve the lower bound on the majority function to $3.5n$. This compares with an upper bound of $O(n \log n)$ by Ajtai, Komlos and Szemeredi [1]. Before detailing the organisation of the remainder of this dissertation we shall briefly return to the relation between combinational and monotone complexity.

Let $f \in M_n$. Define the k-slice of f to be the function f_k:

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

Clearly f_k is the function which has value 0 when fewer than k inputs have the value 1, which equals f when exactly k inputs are true, and which has the value 1 when more than k inputs are 1. Berkowitz [3] showed that a combinational network for f may be efficiently constructed from combinational networks realising the n k-slice functions of f $(1 \leq k \leq n)$ and further proved that for a k-slice, f_k, the combinational and monotone complexities of f_k differed by at most an additive term of $O(n^2 \log n)$ (subsequently improved to $O(n \log^2 n)$ [52]). These results lead to the conclusion that a monotone boolean function f has "large" combinational complexity if and only if some k-slice has "large" monotone network complexity. These results are stated formally in Chapter(6) below and in [56].

2.4) Thesis Organisation

The work below divides into two main sections. The first, consisting of Chapters (3), (4) and (5) concentrates on the complexity of monotone networks for threshold functions and examines replacement rules in greater detail. The second part, containing chapters (7) and (8), develops the work of Berkowitz and Wegener [51] on slice functions. Chapter(6) gives some results linking these two parts. In Chapter(9) we present conclusions and some open questions.

2.4.1) Chapter 3

We consider the function T_3^n, and prove that any monotone network computing this contains at least $2.5n-5.5$ gates. This improves the previous lower bound of $2n-3$ and implies similar lower bounds for all threshold functions, T_k^n, with $3 \le k \le n-2$. The proof is in two stages: the first an inductive argument; the second a wire counting process. This second part is used to establish the lower bound for the single case where it is not possible to eliminate sufficient gates directly using partial assignments.

2.4.2) Chapter 4

We prove a general lower bound on the monotone network complexity of T_k^n which is of the form:

$$C^m(T_k^n) \ge \varphi(n,k)$$

Where $\varphi(n,k)$ is a piecewise-linear function depending on n and k.

This is used to deduce lower bounds of $3.5n$ for the n-input majority function and of $(2+r_k)n$ for T_k^n, where: $k \le n/2$, $k = \Theta(n)$ and $r_k > 1/2$ is a constant depending on k. The argument employed is a generalisation of the standard

inductive gate elimination method. Applying any partial assignment to a threshold function yields another threshold function, on fewer variables. We define a concept of the "distance" of a threshold function from MAJ_n and use this to describe the effect of any partial assignment π on a monotone network S computing T_k^n, in terms of: the distance of $(T_k^n)^{|\pi}$, the number of inputs of S set to constants by π and the number of gates eliminable from S by applying π. We call these three values the *descriptor* of π. A *reduction* (R), is a set of pairs of descriptors with the following properties:

R1) $\forall$ monotone networks S computing T_k^n, $\exists\,(d_1,d_2)$ in R such that partial assignments, π and π', applicable to S can be found with:

$$descriptor(\pi) = d_1 \text{ and } descriptor(\pi') = d_2$$

R2)

$$distance\,((T_k^n)^{|\pi}) + distance\,((T_k^n)^{|\pi'}) = 2\,distance\,(T_k^n)$$

By analysing how general reductions relate to the network size and by demonstrating the correctness of a specific reduction, we derive the lower bounds stated. The lower bounds are obtained entirely by an inductive argument, there are no "special cases" to consider, as in Chapter(3), and the approach used permits a modicum of freedom in the choice of partial assignment.

2.4.3) Chapter 5

We characterise all replacement rules of the form:

"g is replaceable by h in monotone networks computing f."

This characterisation is performed in two stages:

a) We determine the widest range of monotone boolean functions $[h_1,h_2]$ depending on f and g such that:

g is replaceable by h when computing f if and only if:

$$h_1 \leq h \leq h_2$$

b) Similarly we determine the widest range of monotone boolean functions $[g_1, g_2]$ depending on f and h such that:

g is replaceable by h when computing f if and only if:

$$g_1 \leq g \leq g_2$$

For (b) the special cases when h is a constant function are examined. These results are applied in reproving a number of specific replacement rules.

2.4.4) Chapter 6

Berkowitz proved that for any k-slice, $f_k(X_n)$, in any $\{\wedge,\vee,\neg\}$-network computing f_k having negation restricted to the inputs, all instances of $\neg x_i$ could be replaced by $T_k^{n-1}(X_n - x_i)$ and the resulting network would still compute f_k. $\{\wedge,\vee,\neg\}$-networks, which can compute any boolean function, can be converted to networks in which only inputs are negated by applying De Morgan's Laws. Such networks are at most a constant factor larger than optimal networks.

We prove that for every $f \in M_n$ there exists for each x_i, a monotone boolean function h_i depending on f and x_i, such that h_i may replace any instance of $\neg x_i$ in Ω-networks of the form above, computing f. We call such replacing functions *"pseudo-complements"* and for any given f, x_i determine the unique interval in which these must lie. Unfortunately, these results do not appear, in general, to yield an efficient simulation of combinational networks by monotone networks. We give an alternative proof of Berkowitz' result (cf Wegener [56]) and introduce a generalisation of slice functions obtaining similar, slightly weaker, translational results for these.

2.4.5) Chapter 7

This chapter considers the slice functions of some monotone boolean NP-complete predicates. The predicates examined have a special slice called the *canonical slice* which appears to be the most natural candidate for a "hard" slice function. However, Wegener has shown that the canonical slice of the $(n/2)$*-clique* function is computable by a linear-sized monotone network. We develop this result, showing that the canonical slices of Undirected Hamiltonian Circuit and related predicates (Directed Hamiltonian Circuit, Permanent) are computable by polynomial size monotone networks. In addition, we prove that if the canonical slice of a function has polynomial complexity then all slice functions "within a constant distance" may be realised by polynomial size monotone networks.

In the second section of this chapter the notion of *central slice* functions is defined. We consider the central slices of $(n/2)$*-clique* and *DHC* and show that if these slices are realisable by polynomial size monotone networks then the associated NP-complete predicates are computable by polynomial size combinational networks. The proofs involve a "padding argument" which demonstrates that all slices of these predicates may be computed by projecting from the central slice functions of slightly larger problem instances. These results effectively establish that the central slice of these and related functions is NP-hard.

2.4.6) Chapter 8

The results of Berkowitz do not preclude the possibility that some monotone function f with large monotone complexity may be efficiently realisable by a combinational network: f may have large monotone network complexity but only easy slice functions. As we observed above, the construction of Chapter(6) appears to do little to remove this possibility. In this chapter we define, for each positive integer r, a natural class $Q_{(n,r)}$ of monotone boolean functions. A standard counting argument shows that "almost all" members of $Q_{(n,r)}$ have superlinear combinational complexity. It is proved that for the "hardest" members of each class, there is no asymptotic gap between combinational and monotone network complexity. In addition we obtain the stronger result that this relation holds *for all* members of the class $Q_{(n,2)}$. These results are developed by extending them to multiple output functions and to broader classes of monotone boolean functions.

Chapter 3

A Lower Bound On T_k^n

3.1) Introduction

In this chapter the following result is proved:

$$C^m(T_k^n(X_n)) \geq 2.5n - 5.5 \quad \text{for } n \geq k \text{ and } 3 \leq k \leq n-2 \qquad (3a)$$

It is sufficient to consider only the case $k = 3$, since for $4 \leq k \leq n/2$ it will be clear that the same proof is applicable, and for $n/2 \leq k \leq n-2$, the relation:

$$\widehat{T_k^n} = T_{n-k+1}^n$$

establishes the result by duality.

In this section we shall outline the proof technique employed.

In common with previous lower bounds on the *combinational complexity* of functions in B_n the methods used combine an inductive analysis of optimal monotone networks with a counting argument (cf Paul [38], Blum [7]). The inductive stage consists of selecting an arbitrary network input, x_i say, and proceeds by a case analysis on the fanout of this input and the type of gates it enters. By setting x_i equal to 0 it is possible to eliminate 3 gates, except when x_i enters exactly 2 $\vee$-gates. Since the resulting network computes T_3^{n-1} of $X_n - \{x_i\}$ this is (more than) sufficient to prove the lower bound for these cases. However, because the analysis applies to *any* network input, i.e. not only those which enter gates at a maximal distance from the output node, it follows that the only network structures from which sufficient gates cannot be directly eliminated are those in which *every* network input enters exactly 2 $\vee$-gates. After showing that one special case of this can be handled inductively, we establish some properties of networks of this type which compute T_3^n and deduce the lower bound via a wire counting argument. In the style of Paul [38] this argument reasons about the existence of gates which may be quite distant from the network inputs. The technique is however different from Paul.

As observed above (Chapter 2, Section 2.3), for the inductive stage only partial assignments which set inputs to 0 are useable. To prove similar or larger bounds by setting an input to 1 would require at least $n/2$ gates to be eliminated. The functions analysed by Blum and Paul and the Congruence functions of Stockmeyer [46] are not constrained in this way.

The remainder of this chapter is organised as follows. In Section(3.2) a new replacement rule, for monotone networks computing T_k^n, is proved. This rule will be used in the inductive analysis to deal with the case where some input has fanout equal to 1. Section(3.3) gives the first part of the lower bound proof consisting of the inductive stage and a preliminary wire counting argument which is sufficient to prove a lower bound of $2\frac{1}{3}n$ on T_3^n. In Section(3.4) this wire counting argument is improved to yield a lower bound of $2.5n$. In Section(3.5) an upper bound of kn on the monotone network complexity of T_k^n, for fixed k, is derived.

3.2) Preliminary Results

Lemma 3.1

Let S be an optimal monotone network computing $T_k^n(X_n)$ at some node **t**. S does not contain any gate g for which:

$$T_{k_1}^n \le RES(g) \quad \forall\, 1 \le k_1 < k \text{ and } k \ge 2$$

Proof

Suppose S contains a gate g such that:

$$T_{k_1}^n \le RES(g)$$

for some k_1 as above.

We shall show that S is not optimal.

Let $\hat{S}$ denote the monotone dual network of S. This network computes T_{n-k+1}^n. Let $\widehat{RES}(g)$ be the dual function of $RES(g)$ computed in $\hat{S}$. Clearly:

$$\widehat{RES}(g) \le T_{n-k_1+1}^n$$

By a result of Mehlhorn and Galil [32], (see Chapter(5), Fact(5.1) below), g in $\hat{S}$ is replaceable by the constant 0. Thus, by duality, g in S is replaceable by the constant 1. It follows that S was not optimal.

□

Lemma 3.2

There is an optimal monotone network S computing T_k^n, such that every input x_i of S which has fan-out equal to 1, enters an $\wedge$-gate.

Proof

We show how to restructure S to a network S* satisfying the lemma.

Let x_i be an input of S having fan-out equal to 1 and entering an $\vee$-gate g whose other input is some function f. Observe that:

$$f \le T_k^n(X_n)$$

For suppose $\{x_{p_1},...,x_{p_{k-1}}\}$ is a subset of X_n such that the monom formed by $\wedge$-ing the variables in this set is an implicant of f. The partial assignment:

$$x_{p_j} = 1 \quad \forall\ 1 \le j \le k-1$$

leaves S independent of x_i, but under this assignment S should compute $T_1^{n-k+1}(X_n - \bigcup_{j=1}^{k-1} \{x_{p_j}\})$, which depends on x_i. This contradiction establishes every prime implicant of f is an implicant of T_k^n.

But now, since $g \not\geq t$, S can be restructured as follows:

1) Replace gate g in S by the input x_i.

2) Add one $\vee$-gate to S with inputs f and the output of t

Clearly the new network contains no more gates than S, and computes T_k^n. If g has only a single $\vee$-gate as successor then the steps above may be repeated. Eventually the fan-out of x_i must increase or x_i must enter an $\wedge$-gate. As the fan-out of other inputs is not affected, this process may be applied repeatedly until the lemma is true for all inputs.

□

Lemma 3.3

Let S be any monotone network which computes T_k^n (where $n > k$). Let x_i be any input of S which enters exactly 2 $\vee$-gates, whose other inputs are f_1, f_2.

$\forall\ r \quad 2 \le r \le k-1$

If $\exists$ monom m_1 over $X_n - \{x_i\}$ such that:

$$m_1 \le f_1 \wedge T_{k-r}^n(X_n - \{x_i\})$$

Then *not* $\exists$ any monom m_2 over $X_n - \{x_i\}$ such that:

$$m_2 \le f_2 \wedge T_{r-1}^n(X_n - \{x_i\})$$

Proof

Suppose m_1 and m_2 are two such monoms. The partial assignment $x_j = 1 \ \forall\ x_j \in var(m_1) \cup var(m_2)$ leaves S independent of x_i. But under the assignment S should compute $T_{k-q}^{n-q}(X_n - var(m_1) - var(m_2))$ (where $q = |var(m_1) \cup var(m_2)|$) and this depends on x_i since $q \le k-1$. Contradiction.

□

3.3) A $2\frac{1}{3}n$-Lower Bound on T_3^n

Theorem 3.1

$$C^m(\,T_3^n\,(X_n\,))\ \geq\ 2\frac{1}{3}n\ -5$$

Proof

We proceed by induction on $n \geq 3$

Base n=3

Obvious

Inductive Step

Assume the theorem is true for all values $<n$ and prove it holds for n.

Let S be an optimal monotone network computing $T_3^n(X_n)$ at a unique node t. Select some input x_i of S. It will be shown that by setting $x_i=0$, 3 gates may be eliminated, except when x_i enters exactly 2 $\vee$*−gates*. We proceed by a case analysis on the fan-out of x_i. It is assumed that S has been subjected to the process of Lemma(3.2) and thus any input having fan-out equal to 1 enters an $\wedge$*−gate* in S.

Case 1 fanout(x_i)≥3

Setting $x_i=0$ eliminates at least 3 gates. The resulting network S' computes T_3^{n-1} $(X_n-\{x_i\})$. By the inductive hypothesis:

$$C^m(T_k^n)\ =\ C^m(S)\ \geq\ 3 + C^m(\,T_3^{n-1}\,)\ \geq\ 2\frac{1}{3}n\ -5$$

Case 2 fanout(x_i)=1

x_i enters some $\wedge$-gate, g say. Let h be the gate which supplies the other input of g. It is easy to see that:

a1) $g \neq$ t

a2) $T_2^{n-1}(X_n-\{x_i\}) \leq RES(h)$

Setting $x_i=0$ eliminates g and its (by (a1)) successor. The resulting network computes $T_3^{n-1}(X_n-\{x_i\})$, but still contains gate h, with $T_2^{n-1}(X_n-\{x_i\}) \leq RES(h)$. From Lemma(3.1) the gate h may be replaced by 1 in this network. Thus setting $x_i = 0$ eliminates 3 gates.

Case 3 fanout(x_i)=2

3.1) x_i enters at least one $\wedge$*−gate*.

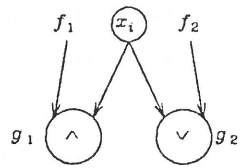


Figure 3.1

If x_i is set to 0, the gates g_1, g_2 and all the successors of the $\wedge$*−gate* may be eliminated. The $\wedge$*−gate* must have at least one successor as it cannot be the output gate.

3.2) x_i enters 2 $\vee$*−gates*

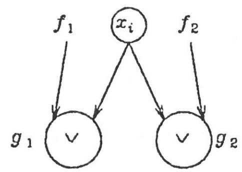


Figure 3.2

Therefore the only case for which insufficient gates can be eliminated directly by setting an input to 0, is when *every* input, x_i, enters exactly 2

$\vee$-gates.

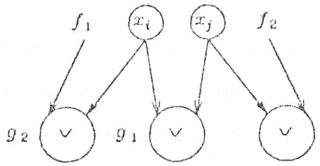


Figure 3.3

Suppose S is a network for which the inductive step fails. If g is a gate both of whose inputs are variables (Fig(3.3)), then g has fanout exactly 1 and enters an $\vee$-gate. For if g_1 has fan-out>1 or enters an $\wedge$-gate, then at least 5 gates may be eliminated by setting $x_i = x_j = 0$. This would be sufficient to prove the result.

To summarise it may now be assumed that:

A1) Every network input enters exactly 2 $\vee$-gates, g_1, g_2

A2) From Lemma(3.3): for at most one of the functions $f_1 f_2$ which enter these gates is it true that:

There exists x_k such that $x_k \le f_i$ $(i = 1$ or $2)$

3) If g_1 has inputs x_i and x_j then g_1 has only one immediate successor and this is an $\vee$-gate.

For any T_2^n network which is not of this form sufficient gates can be eliminated to apply the inductive argument.

The lower bound for the remaining case is derived by a wire counting argument. Let:

$OUT(Q) = |\{$ *The set of wires out of a set of nodes* $Q \}|$

$T = \{\vee$*-gates* $g \mid RES(g) = x_i \vee f,\ x_i$ *is an input of* g and $\exists\, x_j \ne x_i$ *such that* $x_j \le f \}$

$R = \{\vee$*-gates* $g \mid x_i$ *is an input of* $g, g \notin T \}$

$T_1 = \{\vee$*-gates* $g \in T \mid x_i, x_j$ *are inputs of* $g \}$

$T_2 = T - T_1$

$M = \{\vee$*-gates* $g \mid g$ *is the unique successor of some* $h \in T_1, g \notin T_2 \}$

$U = \{\vee$*-gates* $g \in T_2 \mid g$ *is the unique successor of some* $h \in T_1 \}$

$E = \{g \mid g \notin T \cup R \cup M \}$

We can observe the following:

B1) $OUT(X_n) = 2n$ (By analysis above).

B2) $OUT(R) \ge |R|$ (By optimality of S).

B3) $OUT(T) \ge |T|$ (By optimality of S).

B4) $OUT(T_1) = |T_1| = |U| + |M|$ (By (A3))[1].

B5) $OUT(E) \ge |R|$ (By (A2), as each gate in R must have one input from a gate not in $R \cup T \cup M$).

B6) $2|T_1| + |T_2| + |R| = OUT(X_n)$

B7) $|T_1| + |T_2| = |T|$ (By definition).

B8) $OUT(M) \ge |M|$ (By optimality).

Now, it is clear that for any network S:

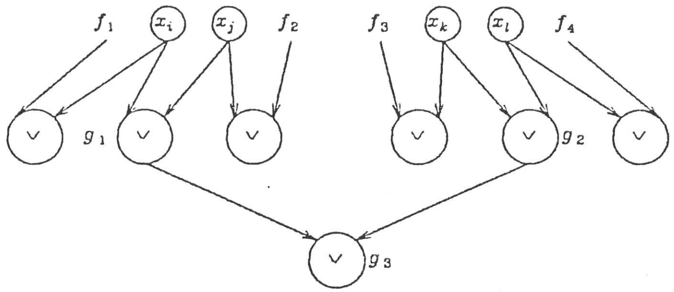
$$C^m(S) = 1/2\ OUT\,(X_n \cup G)$$

The analysis above and (B1)-(B8) are sufficient to establish a lower bound of $2\frac{1}{3}n$ for T_2^n. To avoid unnecessary repetition, this derivation is given only for the improved bound of Theorem(3.2), below.

□

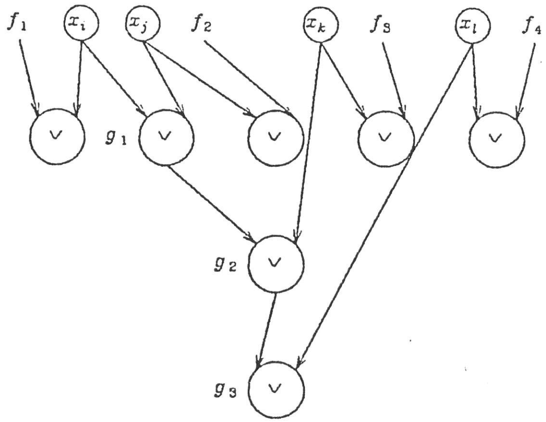
To prove $2.5n$ we improve the lower bound given by (B2).

[1] (B4) holds as each gate in U has only one input from a gate in T_1. Although a gate in M may have two inputs from gates in T_1, since T_1 gates have *fanout* $=1$, by (A3), S may be restructured in this case so that each gate in M has only one input from a T_1 gate. (Fig(3.4))



$g_1, g_2 \in T_1$, $g_3 \in M$

Restructures to:



$g_1 \in T_1$, $g_2 \in U$, $g_3 \in T_2$

Figure 3.4

3.4) An Improved Wire Counting Argument

In this section the relation (B2) above, is improved by showing that:

$$OUT(R) \geq |R| + |U|$$

As will be demonstrated below, this and the previous analysis will establish a lower bound of $2.5n - 5.5$ on T_3^n.

Definition 3.1

Let S compute T_3^n. A *U-configuration* is a subnetwork α of S consisting of 5 gates $\{g_i, g_j, g_k, g_4, g_5\}$ arranged as below: (Figure(3.5)).

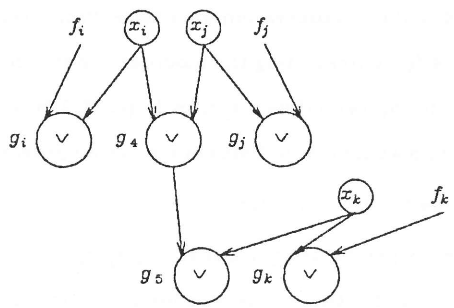


Figure 3.5

Lemma 3.4

Let $P=\{i,j,k\}$ and let S be an optimal monotone network computing T_3^n. S may be restructured to a monotone network S* which is no larger than S, computes T_3^n and satisfies:

(*) For each U-configuration in S*, there exists some $p \in P$ such that every path from g_p to an $\wedge$-gate splits, i.e there exists a gate u on a path from g_p to an $\wedge$-gate h such that fan-out(u)>1.

Proof

Suppose S does not satisfy the lemma. Let α be any *U-configuration* for which (*) is false.

Let h_i, h_j, h_k be the first $\wedge$-gate encountered on paths from g_i, g_j, g_k. (Note that there can only be one "first" $\wedge$-gate on each path as no path splits). All the gates on the paths $[g_p, h_p)$ are $\vee$-gates. Let F_i, F_j, F_k be the function $\vee$-ed with x_i, x_j, x_k on these paths. Let B_i, B_j, B_k be the function fed to the other input of h_i, h_j, h_k, so that $RES(h_p) = B_p \wedge (F_p \vee x_p)$

We perform one modification.

C) If $x_p \leq B_p$ then compute $(x_p \vee F_p) \wedge B_p$ by using one $\wedge$-gate to compute $F_p \wedge B_p$ and $\vee$ the result with x_p. h_p and g_p can then be eliminated. (Fig(3.6))

Thus we may assume that:

$$\forall p \in \{i,j,k\} \quad x_p \nleq B_p$$

We now prove three properties of this subnetwork.

Property 1

h_i, h_j and h_k are distinct.

Proof

Suppose, wlog, that $h_i = h_j$, so that $B_i = x_j \vee F_j$ and $B_j = x_i \vee F_i$. Consider the assignment $x_k = 1$. By arguments similar to the proof of Lemma(3.3) it is easy to see that $x_k x_l \nleq F_i \vee F_j \quad \forall x_l \in X_n - \{x_i, x_j\}$. Thus $S^{|x_k=1}$ depends on x_i, x_j only via h_i. This implies that all gates whose result depends on x_p, other than those on the path $[g_p, h_p)$ are descendants of h_i ($p = i$ or j). But:

$$RES(h_i) = (x_j \vee F_j) \wedge (x_i \vee F_i)$$

and the only prime implicants of this function involving x_i or x_j have the form $x_i x_j$ or $x_i x_p x_q$ or $x_j x_p x_q$ where $p \neq q$. Therefore $S^{|x_k=1}$ cannot compute

$T_2^{n-1}(X_n - \{x_k\})$ and this contradiction establishes the property.

Property 2

Let g be a gate of S such that:

b1) $x_i x_j x_k \leq RES(g)$

b2) $\forall p \in \{i\ j, k\}$ g is not a descendant of any gate on a path $[g_p, h_p]$.

Then: $x_i \vee x_j \vee x_k \leq RES(g)$

Proof

All such gates are descendants of g_5. Partition these descendants into sets according to their distance from g_5, e.g. By breadth-first search rooted at g_5. The proof proceeds by induction on d, the distance of sets from g_5.

Base $d = 0$

Obvious, as the only gate involved is g_5 itself.

Inductive Step

We assume that Property(2) is true for all gates at distance less than d from g_5 and prove it holds for all gates at distance d. Let g be a gate at distance d from g_5 such that $x_i x_j x_k \leq RES(g)$. Let g' and g'' be the inputs of g, both of which satisfy (b2).

Case 1

g is an $\vee$-gate. Then $x_i x_j x_k \leq RES(g')$ or $x_i x_j x_k \leq RES(g'')$, wlog suppose the former. Since the distance of g' from g_5 is less than d, by the inductive hypothesis, $x_i \vee x_j \vee x_k \leq RES(g')$, and so by monotonicity
$x_i \vee x_j \vee x_k \leq RES(g)$.

Case 2

g is an $\wedge$-gate. In this case $x_i x_j x_k \leq RES(g')$ and $x_i x_j x_k \leq RES(g'')$ and Case(2) follows by a similar argument.

Property 3

For all $p \in \{i\ j, k\}$ $x_i x_j x_k \not\leq B_p$

Proof

Suppose, wlog, that $x_i x_j x_k \leq B_i$. The gate which computes B_i must be a descendant of h_j or h_k. To see this recall that $h_i \neq h_j$ and $h_i \neq h_k$ (Property(1)), and so if this observation were false, Property(2) would apply and $x_i \vee x_j \vee x_k \leq B_i$ contradicting the modification (C). It follows that h_i is a descendant of h_j (or h_k) and thus $x_i x_j x_k \leq B_j$ (or B_k). By repeating the argument twice a cycle in S would result. This contradiction proves the claim.

Lemma(3.4) now follows easily for:

Consider the partial assignment $X_n - \{x_i, x_j, x_k\} = 0$. Then $B_p = 0, \forall p \in \{i\ j, k\}$. (Property(3)). S under this partial assignment cannot compute $T_3^3(x_i, x_j, x_k)$ as it only depends on x_i, x_j and x_k via g_5 which computes $T_1^3(x_i, x_j, x_k)$. Contradiction. □

Corollary 3.4.1)

$$OUT(R) \geq |R| + |U|$$

Proof

Let α be any U-configuration in S. Wlog suppose a path from g_i in α splits before meeting an $\wedge$-gate. Let F_i be the function $\vee$-ed with x_i on this path before it splits. It is clear that S may be restructured in such a way that x_i enters an $\vee$-gate g whose other input is F_i with fanout$(g) \geq 2$. This may be done without increasing the size of S, and for all U-configurations in S. □

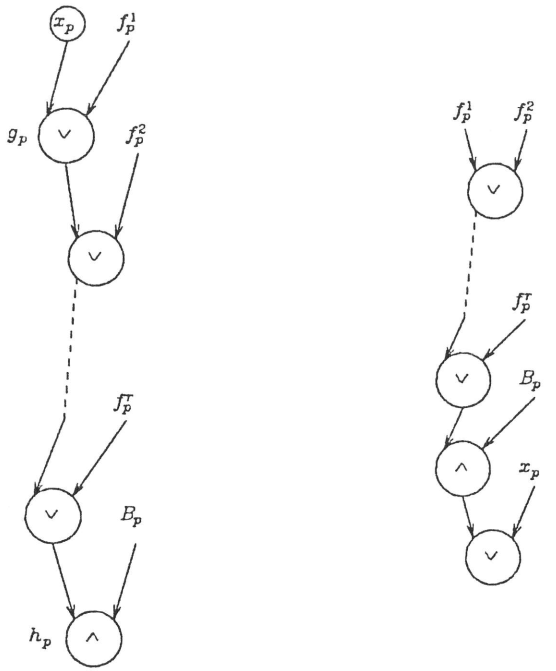


Figure 3.6

This now gives:

Theorem 3.2)

$$C^m(T_3^n) \geq 2.5n - 5.5$$

Proof

Combining the analysis of Theorem(3.1) with (B1)-(B8) and Corollary(3.3.1) yields:

$$\begin{aligned} OUT(G \cup X_n) &= OUT(R \cup E \cup T \cup M \cup X_n) \\ &\geq (|R| + |U|) + |R| + |T| + |M| + 2n \\ &\geq 4n + (|R| + |U|) + |M| - |T_1| \quad (by\ B6, B7) \\ &\geq 4n + |R| \quad (B4) \\ &\geq 5n \quad (as\ |R| \geq n\ from\ (A2)) \end{aligned}$$

Thus;

$$C^m(T_3^n) = C^m(S) \geq 2.5n - 5.5$$

and theorem follows.

□

3.5) An Upper Bound On T_k^n

The lower bound on T_k^n proved above, is possibly sub-optimal. This section presents a monotone network construction for T_k^n.

Lemma 3.5 (Adleman)[2)]

Let $k \in \mathbf{N}$

$$C^m(T_k^n) \leq kn + o(n)$$

To prove the upper bound the following combinatorial result is required.

Fact 3.1

Let:

$$\mathbf{y}_i = \langle y_{i_1}, y_{i_2}, \ldots, y_{i_k} \rangle \in \mathbf{N}^k \quad (\text{where } k \geq 2)$$

Let $\Pi_q : \mathbf{N}^k \rightarrow \mathbf{N}^k$ be the projection which sets the y_{i_q} position of $\mathbf{y}_i$ to 1. Finally let $COVER_k$ be a predicate defined on sets of k-tuples $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_s)$ by:

$$COVER_k(\mathbf{y}_1, \ldots, \mathbf{y}_s) = \begin{cases} 1 \text{ if } \forall\ 1 \leq q \leq k,\ \exists\ \mathbf{y}_i^q, \mathbf{y}_j^q \in Y \\ \quad \text{such that } \Pi_q(\mathbf{y}_i^q) = \Pi_q(\mathbf{y}_j^q) \text{ and } i \neq j \\ 0 \text{ otherwise} \end{cases}$$

Then:

$$\min_{Y \subset \mathbf{N}^k} \{ |Y| \mid COVER_k(Y) = 1 \} = k+1$$

Proof

Upper Bound

Elementary

2) This result is reported by Bloniarz [5] but no proof is given and the construction of Adleman is as yet unpublished. The method presented here was suggested by Paterson [37], the proof of its correctness is by the author.

Lower Bound

By induction on $k \geq 2$. The base $k = 2$ is immediate, so we assume the lower bound holds for all values less than k.

Let $Y = \{y_1, \ldots, y_s\}$ be any set of k-tuples such that $COVER_k(Y) = 1$. Wlog it may be assumed that:

$$\Pi_1(y_1) = \Pi_1(y_2)$$

Thus, as $COVER_k(Y) = 1$, the set of $(s-1)$ $(k-1)$-tuples

$$\{<y_{1_2}, \ldots, y_{1_k}>\} \cup \bigcup_{i=3}^{s} \{<y_{i_2}, \ldots, y_{i_k}>\}$$

must satisfy $COVER_{k-1}$. By the Inductive Hypothesis:

$$s-1 \geq k \quad => \quad s \geq k+1$$

The lower bound follows.

□

Proof of Lemma(3.5) (Outline)

For ease of exposition, suppose $n = p^k$ for some positive integer p. It is easy to see how to amend the construction below if n is not of this form. Let:

$$X_n = \bigcup_{\substack{1 \leq r_1 \leq p \\ 1 \leq r_2 \leq p \\ \cdots \\ 1 \leq r_k \leq p}} \{x_{r_1 r_2 \ldots r_k}\}$$

To avoid a plethora of subscripts $<r_1, \ldots, r_k>$ will denote $x_{r_1 \ldots r_k}$. It will be convenient to consider the elements of $\{1,2,\ldots,p\}^{k-1}$ arranged in lexicographic order. Thus $r_i = <r_i^1, r_i^2, \ldots, r_i^{k-1}>$ is the $i'th$ element. e.g $r_1 = <1,1,1\ldots,1>$

The q*-partition* of X_n is constructed as follows.

T1) X_n is partitioned into p^{k-1} *blocks*, B_i^q, where $1 \leq i \leq p^{k-1}$. Each block contains p elements of X_n.

T2) The particular elements of X_n in a block B_i^q are given by:

$$B_i^q = \bigcup_{j=1}^{p} \{<r_i^1, \ldots, r_i^{q-1} j, r_i^{q+1}, \ldots, r_i^k>\}$$

where $<r_i^1, \ldots r_i^k>$ is the $i'th$ element of $\{1,2,\ldots,p\}^{k-1}$ in the ordering described above.

The q*-partition* of X_n thus consists of p^{k-1} blocks each block being defined by a distinct $(k-1)$-tuple.

Clearly there are k possible q*-partitions* of X_n. We claim that:

$$T_k^{p^k}(X_n) = \bigvee_{q=1}^{k} T_k^{p^{k-1}}(T_1^p(B_1^q), \ldots, T_1^p(B_{p^{k-1}}^q)) \qquad (*)$$

If this assertion holds, it gives rise to a recursive construction for a monotone network computing T_k^n. Solving the underlying recurrence relation yields the upper bound stated. We justify this assertion as follows.

First observe that if fewer than k elements of X_n are assigned the value 1 then the righthand side of (*) is 0. Since the RHS is clearly monotone it is sufficient to prove that it attains the value 1 whenever exactly k members of X_n are 1.

Consider any assignment to X_n for which exactly k variables are set to 1. Let:

$$Y = \{y_1, y_2, \ldots, y_k\}$$
$$= \{<y_{1_1}, y_{1_2}, \ldots, y_{1_k}>, \ldots, <y_{k_1}, y_{k_2}, \ldots, y_{k_k}>\}$$

be the k variables of X_n which are fixed to 1. From Fact(3.1), since $|Y| < k+1$, $COVER_k(Y) = 0$. It follows that there exists some s (with $1 \leq s \leq k$) such that:

$$\{<y_{1_1}, \ldots, y_{1_{s-1}}, y_{1_{s+1}}, \ldots, y_{1_k}>, \ldots, <y_{k_1}, \ldots, y_{k_{s-1}}, y_{k_{s+1}}, \ldots, y_{k_k}>\}$$

are distinct $(k-1)$-tuples in $\{1,2,\ldots,p\}^{k-1}$. Therefore by the definition of q*-partition*:

$$y_i \in B_l^s \text{ and } y_j \in B_l^s \iff i = j$$

Thus no two y_i's (i.e variables of X_n which are set to 1) are in the same block of the s*-partition* of X_n. So:

$$T_k^{p^{k-1}} \left(T_1^p(B_1^s), T_1^p(B_2^s), \ldots, T_1^p(B_{p^{k-1}}^s) \right) = 1$$

and therefore the RHS of (*) is 1.

□

Figure(3.7) illustrates the construction for $n = 8$ and $k = 3$.

Let k=3 and n=8

X_8 = { <1,1,1> , <1,1,2> , <1,2,1> , <1,2,2> , <2,1,1> , <2,1,2> , <2,2,1> , <2,2,2> }

1-partition

B_1^1 = {<1,1,1> , <2,1,1>} ; B_2^1 = {<1,1,2> , <2,1,2>}
B_3^1 = {<1,2,1> , <2,2,1>} ; B_4^1 = {<1,2,2> , <2,2,2>}

2-partition

B_1^2 = {<1,1,1> , <1,2,1>} ; B_2^2 = {<1,1,2> , <1,2,2>}
B_3^2 = {<2,1,1> , <2,2,1>} ; B_4^2 = {<2,1,2> , <2,2,2>}

3-partition

B_1^3 = {<1,1,1> , <1,1,2>} ; B_2^3 = {<1,2,1> , <1,2,2>}
B_3^3 = {<2,1,1> , <2,1,2>} ; B_4^3 = {<2,2,1> , <2,2,2>}

Figure 3.7

Chapter 4

Lower Bounds On Arbitrary Threshold Functions

4.1) Introduction

Theorem(3.2) above yielded an improved lower bound on T_k^n when k was fixed. In this chapter a general lower bound on T_k^n, which gives larger bounds for $k = \Theta(n)$, $k \leq \lceil n/2 \rceil$, is presented. Our main result is the following:

$\forall \ 3 \leq k \leq \lceil n/2 \rceil$

$$C^m(T_k^n) \geq \max\{2n+3k, 2.5n+1.5k\} - C \qquad (4a)$$

Where C is a constant.

For the majority function, we deduce a lower bound of $3.5n$, slightly improving Bloniarz' $3n$ lower bound [5].

The remainder of this section discusses the proof technique employed. In Section(4.2) a general lower bound on T_k^n is derived and in Section(4.3) we develop the results of this section to obtain the relation (4a) above.

The approach is a generalisation of the standard inductive gate elimination argument described in Chapter(2). Three ideas are central to the proof method: extending the definition of "family of functions" as used in the Inductive step; the notion of the "distance" of T_k^n from MAJ_n; and the concept of a *reduction*. This last was briefly described in Chapter 2, Section(2.4.2).

Instead of considering a family of monotone functions $\{f_1, \ldots, f_n, \ldots\}$, in which for each n there is *at most one* n-input function, we consider families of *sets* of functions:

$$\{\{F_1\},\{F_2\},\ldots,\{F_n\},\ldots\}$$

In this way each $f \in F_n$ is an n-input function. For the Inductive step it is then sufficient to project onto a member of a smaller indexed set. (The definition should not be confused with inductive methods for multiple output functions.) This generalisation is not new, it is, for example, inherent in Weiss' lower bound method for Convolution [57]. The family we shall use is:

$$\bigcup_{n=2}^{\infty} \{ \bigcup_{k=2}^{n} \{T_k^n\} \}$$

Thus the n'th member is the set:

$$\{T_2^n, T_4^n, \ldots, T_{n-1}^n\}$$

The "distance" of T_k^n from majority is *related to* the minimum value of $|\pi|$, where π is the partial assignment such that:

$$(T_k^n)^{|\pi} = MAJ_{n-|\pi|} \text{ and } n-|\pi| \text{ is even}$$

Using these concepts the lower bound proof divides into three parts: we first show how an arbitrary reduction may be used to reason about the size of monotone networks computing T_k^n; then, assuming the correctness of a specific reduction, it is proved that a particular piecewise-linear function $\varphi(n,k)$, gives lower bounds for T_k^n. The final stage is to verify the correctness of this reduction. This is done by a case analysis on the structure of optimal networks.

4.2) Main Result

Definition 4.1

Define $\Delta(T_k^n)$ to be $n/2-k$. Δ represents the "distance" of T_k^n from MAJ_n and may be negative and non-integral.

Definition 4.2

Let S be a monotone network computing T_k^n. Let π be a partial assignment such that $S \xrightarrow{\pi} S'$, i.e $S^{|\pi} = S'$, where S' computes T_k^{n-r}. The *descriptor* of π, $\delta(\pi)$, is a triple (r, s, t) where:

$$r = |\{ \textit{Inputs of S set by } \pi \}|$$
$$s = \Delta(T_k^{n-r})$$
$$t \leq |\{ \textit{Gates deleted from S by applying } \pi \}|$$

Definition 4.3

An $\alpha\beta$-reduction for T_k^n, is a set of q descriptor pairs, $\{ <\alpha_i, \beta_i> \}$ such that:

For any S computing T_k^n, $\exists <\alpha_i, \beta_i>$ and partial assignments π, π' applicable to S for which:

$$\delta(\pi) \in \{ \alpha_i, \beta_i \} \quad (1)$$
$$\delta(\pi) = \alpha_i \iff \delta(\pi') = \beta_i \quad (2)$$
$$\forall <\alpha_i, \beta_i> \; 2\Delta(T_k^n) - (s_i + s_i') = 0 \quad (3)$$

Lemma 4.1

Let S compute T_k^n and let $\{ <\alpha_i, \beta_i> \}$ be an $\alpha\beta$-reduction for S. Let $\Delta(T_k^n) = s$. If there is a function $\varphi(n,s) \rightarrow \mathbf{Q}^+$ such that:

$$\varphi(n,s) \leq \max \begin{cases} \varphi(n-r_i, s_i) + t_i \\ \varphi(n-r_i', s_i') + t_i' \end{cases} \quad (4b)$$

$$\forall <\alpha_i, \beta_i> \equiv <(r_i, s_i, t_i), (r_i', s_i', t_i')>$$

and

$$\varphi(n,\Delta(T_1^n)) = \varphi(n,\Delta(T_{n-1}^n)) = n - Constant$$

then:

$$C^m(T_k^n) \geq \varphi(n,s)$$

Proof

By induction on n

Base

The recurrence of (4b) will terminate at $\varphi(n,\Delta(T_1^n))$ or $\varphi(n,\Delta(T_n^n))$. The conditions on φ yield the lower bound for the inductive base.

Inductive Step

We assume $\forall\, n' < n,\ \forall\, k'$ that:

$$C^m(T_{k'}^{n'}) \geq \varphi(n',s')$$

where $\Delta(T_{k'}^{n'}) = s'$

and show that this implies the result.

Thus let S be a monotone network computing T_k^n. As $\{ <\alpha_i,\beta_i> \}$ is an $\alpha\beta$-reduction, there exist partial assignments π, π', applicable to S, such that:

$$<\delta(\pi),\delta(\pi')> \; = \; <\alpha_i,\beta_i>$$

for some $1 \leq i \leq q$.

Thus:

$$C^m(T_k^n) \geq \max\begin{cases} C^m\left(T^{n-r_i}_{(\frac{n-r_i}{2}) \pm s_i}\right) + t_i \\ C^m\left(T^{n-r'_i}_{(\frac{n-r'_i}{2}) \pm s'_i}\right) + t'_i \end{cases}$$

By the inductive hypothesis:

$$C^m(T_k^n) \geq \max\begin{cases} \varphi(n-r_i,s_i) + t_i \\ \varphi(n-r'_i,s'_i) + t'_i \end{cases}$$

But:

$$\varphi(n,s) \leq \max\begin{cases} \varphi(n-r_i,s_i) + t_i \\ \varphi(n-r'_i,s'_i) + t'_i \end{cases}$$

Hence: $C^m(T_k^n) \geq \varphi(n,s)$ □

Lemma(4.1) yields a recurrence expression for the monotone network complexity of T_k^n. We do not attempt to find a general solution to this, but illustrate that a particular $\varphi(n,s)$ is given by a specified $\alpha\beta$-reduction.

Lemma 4.2

If:

$AB = \{ <(1, s+1/2, 4), (1, s-1/2, 3)>,$ (1)

$<(1, s+1/2, 5), (1, s-1/2, 2)>$ (2)

$<(2, s+1, 8), (2, s-1, 6)>$ (3)

$<(1, s+1/2, 3), (1, s-1/2, 4)>$ (4)

$<(1, s+1/2, 2), (1, s-1/2, 5)>$ (5)

$<(2, s+1, 6), (2, s-1, 8)> \}$ (6)

is an $\alpha\beta$-reduction for every S computing $T^n_{n/2-s}$

$$\varphi(n,s) = \begin{cases} 3.5n - |s| - C & 0 \leq |s| \leq 3/2 \\ 3.5n - 3|s| + 3 - C & |s| \geq 3/2 \end{cases}$$

satisfies:

$$\varphi(n,s) \leq \max\begin{cases} \varphi(n-r_i,s_i) + t_i \\ \varphi(n-r'_i,s'_i) + t'_i \end{cases}$$

$\forall\ <\alpha_i,\beta_i> \; := \; <(r_i,s_i,t_i),(r'_i,s'_i,t'_i)> \; \in AB$

Proof

By inspection. □

Some intuition for the choice of $\varphi(n,s)$ may be garnered from Figure(4.1). This illustrates $\varphi(n,s)$ for values of n, $n-1$ (Vertical axis) against s. In terms of the usual form of inductive argument, Fig(4.1) can be viewed as follows:

"For any monotone network S_0 which realises T_k^n, one can find partial assignments $\pi_1, \pi_2, \ldots, \pi_r$ such that:

$$(S_i)^{|\pi_{i+1}} = S_{i+1} \quad \forall\ 0 \le i < r$$

and the network S_r computes a threshold function which is "close to" majority. Then, for any T_k^n, close to majority, it is possible to choose partial assignments, π, which eliminate, on average, 3.5 gates and such that $(T_k^n)^{|\pi}$ is also close to majority."

We observe that the $\alpha\beta$-reduction **AB**, can be similarly interpreted, for a number of different $\varphi(n,s)$. One such interpretation is outlined in Section(4.3) below.

It may be noted that in some $<\alpha_i,\beta_i>$:

$$\varphi(n,s) \ge \min \begin{cases} \varphi(n-r_i,s_i) + t_i \\ \varphi(n-r_i',s_i') + t_i' \end{cases}$$

$e.g$ $\varphi(n,1/2) > \varphi(n-2,-1/2) + 6$

This imposes a strategy in inductively eliminating gates from S computing $T_{\lceil n/2\rceil - s}^n$, in that for those $<\alpha_i,\beta_i>$ having this property the step which reduces to:

$$\varphi(n,s) \le \max \begin{cases} \varphi(n-r_i,s_i) + t_i \\ \varphi(n-r_i',s_i') + t_i' \end{cases}$$

must be applied.

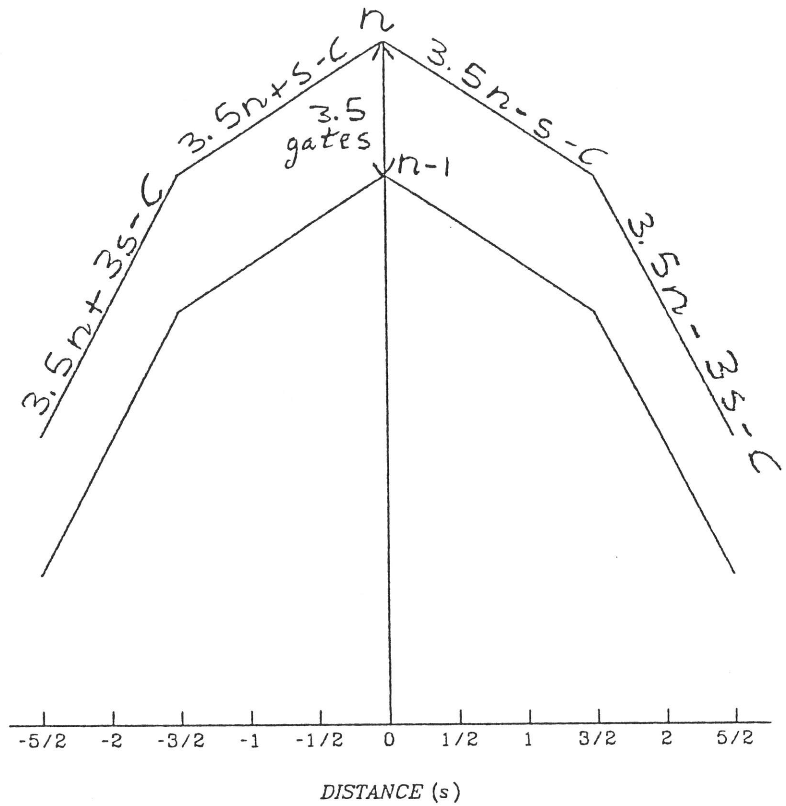


Figure 4.1

Lemma 4.3 (Schnorr [43])

If S computes T_k^n and $k \notin \{1,n\}$, then every input x_i entering a gate g at a maximal distance from the output of S has fanout≥2.

Proof

g must have inputs x_i, x_j. If the fanout of x_i equals 1, then there is a partial assignment $x_j = c$ $c \in \{0,1\}$, such that $S^{|x_j=c}$ does not depend on x_i. But $S^{|x_j=c} = T_{k'}^{n-1}$ where $k' \in \{k, k-1\}$, and this function still depends on x_i if $k \notin \{1,n\}$. Contradiction.

□

Theorem 4.1

Let S be any optimal network computing T_k^n for $1<k<n$. Then AB is an $\alpha\beta$-reduction for S.

Proof

Let g_2 be a gate of S at a maximal distance from the output. The inputs of g_2 must be distinct inputs x_i, x_j of S. We proceed by case analysis on the environment of x_i and show that in each case some $<\alpha_i, \beta_i> \in$ AB is applicable.

Case 1 fanout(x_i)=1

If $1 < k < n$, then from Lemma(4.3) this case cannot occur.

Case 2 fanout(x_i)≥3

There must exist some constant $c \in \{0,1\}$ such that setting $x_i = c$ eliminates at least 5 gates, as two of the gates entered must have the same operation. But, since the fanout of $x_i \geq 3$, setting $x_i = \neg c$ must eliminate at least 3 gates. Thus:

$$<\delta(x_i = 1), \delta(x_i = 0)> \in \{<\alpha_2, \beta_2>, <\alpha_5, \beta_5>\}$$

Case 3 fanout(x_i)=2

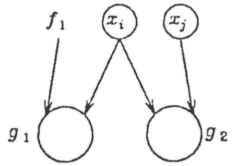


Figure 4.2

3.1) op(g_1) = op(g_2), wlog op(g_1)=$\vee$

3.1.1) fanout(g_1)$\geq$2 or fanout(g_2)$\geq$2

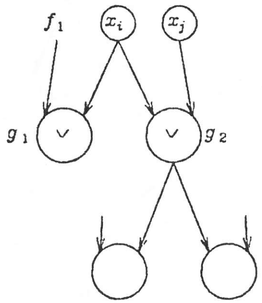


Figure 4.3

$$<\delta(x_i=1),\delta(x_i=0)> \;=\; <\alpha_2,\beta_2>$$

3.1.2) successor(g_1)=successor(g_2)

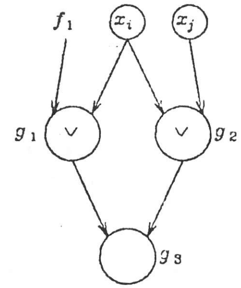


Figure 4.4

In both possible cases ($op(g_3) = \vee$ *or* $\wedge$) S is not optimal.

(3.1.1) and (3.1.2) leave only the subcase (3.1.3) where g_1 has a unique successor g_3, and g_2 has a unique successor g_4 with $g_4 \neq g_3$.

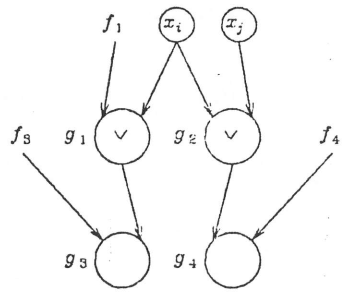


Figure 4.5

3.1.3.1) op(g_3)=op(g_1) or op(g_4)=op(g_2)

$$Then:\; <\delta(x_i=1),\delta(x_i=0)> \;=\; <\alpha_2,\beta_2>$$

Thus we need only now consider the case where op(g_3)=op(g_4)=$\wedge$

By the choice of g_2, the other function, f_4 say, input to g_4 is either some input x_k of S or the output of a gate at a maximal distance from the output of S.

3.1.4) $f_4 = x_k$

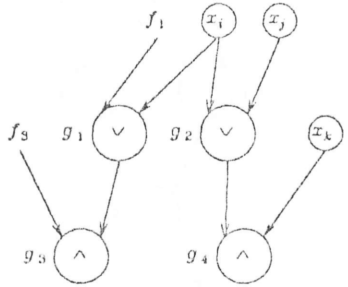


Figure 4.6

$$<\delta(x_i = 1), \delta(x_k = 0)> = <\alpha_1, \beta_1>$$

3.1.5)

f_4 is the output of a gate h_1 with inputs x_k, x_l

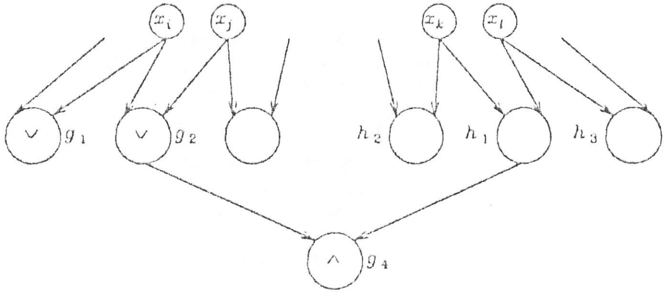


Figure 4.7

3.1.5.1) op(h_1)=$\wedge$

$$<\delta(x_k = 1), \delta(x_k = 0)> = <\alpha_5, \beta_5>$$

3.1.5.2) op(h_2)=$\wedge$ or op(h_3)=$\wedge$

$$\left.\begin{array}{c} <\delta(x_i = 1), \delta(x_k = 0)> \\ or \\ <\delta(x_i = 1), \delta(x_l = 0)> \end{array}\right\} = <\alpha_1, \beta_1>$$

3.1.5.3) op(h_q)=$\vee$ $\forall q$ $1 \le q \le 3$

$$<\delta(x_i = x_k = 1), \delta(x_i = x_j = 0)> = <\alpha_3, \beta_3>$$

This exhausts Case(3.1), since the case $op(g_1) = op(g_2) = \wedge$ follows by a dual argument.

3.2) op(g_1)$\neq op(g_2)$

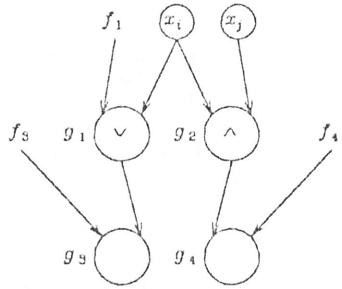


Figure 4.8

3.2.1) fanout(g_1)$\ge$2 or fanout(g_2)$\ge$2

$$<\delta(x_i = 1), \delta(x_i = 0)> \in \{<\alpha_1, \beta_1>, <\alpha_4, \beta_4>\}$$

3.2.2) successor(g_1)=successor(g_2)

In both possible cases S is not optimal

3.2.3) op(g_3)=op(g_1) or op(g_4)=op(g_2)

$$Then: <\delta(x_i = 1), \delta(x_i = 0)> \in \{<\alpha_1, \beta_1>, <\alpha_4, \beta_4>\}$$

Thus as in Case(3.1), $f_4 = x_k$ or f_4 is the output of a gate at maximal distance from the output of S.

3.2.4) $f_4 = x_k$

Similar to subcase(3.1.4) above

3.2.5)

f_4 is the output of gate h_1 with inputs x_k , x_l

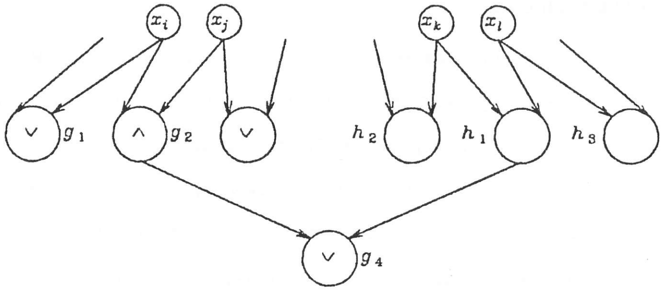


Figure 4.9

3.2.5.1) op(h_1)=∨

$$<\delta(x_k=1),\delta(x_k=0)> \; = \; <\alpha_2,\beta_2>$$

3.2.5.2) op(h_2)=∧ or op(h_3)=∧

$$<\delta(x_i=1),\delta(x_k=0)> \; = \; <\alpha_4,\beta_4>$$

or:

$$<\delta(x_i=1),\delta(x_l=0)> \; = \; <\alpha_4,\beta_4>$$

3.2.5.3) op(h_2)=op(h_3)=∨, op(h_1)=∧

$$<\delta(x_i=x_j=1),\delta(x_i=x_k=0)> \; = \; <\alpha_3,\beta_3>$$

This removes the last case. In every instance some $<\alpha_i,\beta_i>$ applies and thus AB is an $\alpha\beta$-reduction. □

4.3) Consequences Of Theorem(4.1)

Corollary 4.1

$\forall k \;\; 3 \le k \le \lceil n/2 \rceil$

$$C^m(T_k^n) \ge 2n + 3k - C \qquad C \in Q^+$$

Proof

Let $k = n/2 - s$, $s \in Q^+$. By Lemma(4.2) and Theorem(4.1)

$$C^m(T_k^n) = C^m(T_{n/2-s}^n) \ge 3.5n - 3s - C'$$

However: $s = n/2 - k$, thus

$$C^m(T_k^n) \ge 3.5n - 3(n/2-k) - C'$$
$$\ge 2n + 3k - C \qquad □$$

Theorem 4.2

$$C^m(T_k^n) \ge 4(k-3) + C^m(T_3^{n-k+3}) - C$$

Proof (Outline)

The $\alpha\beta$-reduction AB may be interpreted by saying:

"For any monotone network S computing T_k^n, ∃ some input x_i and some constant $c \in \{0,1\}$ such that setting $x_i = c$ eliminates at least 4 gates."

Choosing a suitable $\varphi(n, \Delta(T_k^n))$ leads to the theorem.

□

Corollary 4.2

If $C^m(T_3^n) = (2+\lambda)n - C$ then:

$\forall\ 3 \le k \le \lceil n/2 \rceil$

$$C^m(T_k^n) \ge \max\begin{cases} 2n + 3k - C_0 \\ (2+\lambda)n + (2-\lambda)k - C_1 \end{cases}$$

$$\ge \begin{cases} 2n + 3k - C_0 & k \ge \dfrac{\lambda n}{\lambda+1} \\ (2+\lambda)n + (2-\lambda)k - C_1 & k \le \dfrac{\lambda n}{\lambda+1} \end{cases}$$

□

Theorem 4.3

$$C^m(T_k^n) \ge \begin{cases} 2n + 3k - C_0 & k \ge n/3 \\ 2.5n + 1.5k - C_1 & k \le n/3 \end{cases}$$

Proof

From Theorem(3.2)

$$C^m(T_3^n) \ge 2.5n - 5.5$$

and the theorem follows from Corollary(4.2) with $\lambda = 1/2$.

□

Corollary(4.2) implies that improved lower bounds on T_3^n or any T_k^n with k fixed, would lead to consequent improvements in Theorem(4.3). In particular a $3n$ lower bound on T_3^n would immediately give the $3.5n$ lower bound on MAJ_n.

Chapter 5

Replacement Rules In Monotone Boolean Networks

5.1) Introduction

Replacement rules were introduced by Paterson [36] and Mehlhorn & Galil [32] and used to prove tight lower bounds on the monotone network complexity of boolean matrix multiplication. The results applied prove that in networks computing boolean matrix product, gates computing certain functions may be replaced by the constants 0 or 1 or by an input of the network.

In this chapter we investigate the following problem:

(P1) Given a pair of functions, $f, g \in M_n$, what are the monotone boolean functions h such that for any monotone network S computing f, containing a gate **u** which computes the function g, $S^{|RES(\mathbf{u}):=h}$ still computes f?

$S^{|RES(\mathbf{u}):=h}$ denotes the network S, after the gate **u** is replaced by a node computing h.

The following results are proved:

R1) For any $f \in M_n$ we derive closed form expressions for the maximal 0-replaceable and minimal 1-replaceable functions with respect to f.

R2) For any pair of functions f, $g \in M_n$ we determine closed form expressions for:

(i) min s such that g is replaceable by s in a network computing f
(ii) max s such that g is replaceable by s in a network computing f

R3) For any pair of functions $f, g \in M_n$ we determine closed form expressions for:

(i) min s such that s is replaceable by g in a network computing f
(ii) max s such that s is replaceable by g in a network computing f

Where "minimum" and "maximum" pertain to the partial order relation, "≤", of Definition(1.1.2).

Using (R2) and (R3) we obtain a complete solution for (P1).

This chapter contains four main sections. Sections(5.2), (5.3) and (5.4) obtain the relations of (R1), (R2) and (R3) respectively. Finally Section(5.5) generalises these results to deal with multiple output monotone boolean functions and gives new proofs of some known specific replacement rules.

All the expressions derived are based on the representations of f as a Conjunctive or Disjunctive Normal Form. In this way the proof methods are freed of assumptions about the structure of monotone networks computing f. Such an approach is possible since the concept of "replaceability", as outlined above, gives rise to an ordering relation between $g_1, g_2 \in M_n$ for each $f \in M_n$. Thus, $g_1 \overset{f}{=\!\!|} g_2$ if and only if "g_1 is replaceable by g_2 in monotone networks computing f". It is clear that the relation "$\overset{f}{=\!\!|}$" is reflexive and transitive and therefore defines a pre-order on the members of M_n. Our results may be viewed as establishing some properties of these pre-orders.

Beynon [4], by considering this purely algebraic formulation and by using the expressions derived below (as given in [10]), has obtained analogues of (R1), (R2) and (R3), for the wider context of arbitrary finite distributive lattices.

5.2) Replaceability By Constant Functions

Definition 5.1)

Let $f, g, h \in M_n$.

g is *h-replaceable with respect to* f ($g \overset{f}{=\!\!|} h$) iff:

$$S^{|RES(\mathbf{u}):=h} \text{ still computes } f$$

for any monotone network **S** computing f which contains a node **u** with $RES(\mathbf{u})=g$ □

The following result is the replacement rule due to Mehlhorn and Galil [32].

Fact 5.1)

$g \overset{f}{=\!\!|} 0$ iff:

$\forall\ \mathbf{m} \in PI(g)\ \neg\exists$ any monom $\mathbf{m}'$ such that

$$\mathbf{m} \wedge \mathbf{m}' \in PI(f)$$

□

A corollary of this is:

Fact 5.2)

$g \overset{f}{=\!\!|} 1$ iff:

$$\forall h \quad g \wedge h \le f \implies h \le f$$

□

In this section we characterise the largest 0-replaceable and smallest 1-replaceable functions with respect to any function f. Although the results are implied by Theorem(5.3) below, these cases are presented separately as the analysis is simpler.

Definition 5.2)

Let m be a monom, c be a clause and $f \in M_n$.

1) $\chi(m) = \vee \{ x \in X_n \mid m \not\leq x \}$

2) $Z(f) = \bigwedge_{m \in PI(f)} \chi(m)$

3) $\varphi(c) = \wedge \{ x \in X_n \mid x \not\leq c \}$

4) $U(f) = \bigvee_{c \in PC(f)} \varphi(c)$

□

Theorem 5.1) (0,1-replacements)

(A) $Z(f)$ is the unique largest 0-replaceable function with respect to f. i.e

$g \leq Z(f) \iff g$ is 0-replaceable w.r.t f.

(B) $U(f)$ is the unique smallest 1-replaceable function with respect to f. i.e

$g \geq U(f) \iff g$ is 1-replaceable w.r.t f.

Proof

Let:

$$Dross(f) = \vee \{ g \in M_n \mid g \overset{f}{=\!\!|} 0 \}$$

We shall show that:

$$Z(f) = Dross(f)$$

(i) $Z(f) \leq Dross(f)$

Suppose $m \leq Z(f)$, but that $m \not\leq Dross(f)$. Then there is some monom m' such that $m \wedge m' \in PI(f)$ and so m is a shortening of some prime implicant r of f. But:

$$m \leq Z(f) = \bigwedge_{p \in PI(f)} \chi(p) \Rightarrow m \leq \chi(r)$$

by monotonicity

But $m \not\leq \chi(r)$ if m is a shortening of r. Contradiction.

(ii) $Dross(f) \leq Z(f)$

Let $m \in PI(Dross(f))$, by Fact(5.1) there does not exist any monom m' such that $m \wedge m' \in PI(f)$. Thus:

$$\forall p \in PI(f) \quad m \leq \chi(p)$$

$$m \leq \bigwedge_{p \in PI(f)} \chi(p) \leq Z(f)$$

Thus:

$$Z(f) = Dross(f)$$

and (A) follows as $Dross(f)$ is by definition the unique largest 0-replaceable function with respect to f.

(B) It is easy to see that:

$$U(f) = \widehat{Z(\hat{f})}$$

By duality, $U(f)$ is the unique minimal 1-replaceable function with respect to f.

□

5.3) Replaceability By Arbitrary Functions (1)

We shall now consider non-constant replacements of the form $g := s$, and determine minimum and maximum solutions for these.

Definition 5.3)

Let $\mathbf{M} = \{ \mathbf{m}_1, \ldots, \mathbf{m}_k \}$ be a set of monoms, and let f be a monotone boolean function. The *Prime- Implicant Extension* of $\mathbf{M}$ with respect to f ($\mathrm{IE}_f(\mathbf{M})$) is defined to be:

$$\mathrm{IE}_f(\mathbf{M}) = \{ \mathbf{p} \in \mathrm{PI}(f) \mid \exists\, \mathbf{m}_i \in \mathbf{M} \text{ with } \mathbf{p} \leq \mathbf{m}_i \}$$

The *Prime-Clause Extension* of a set of clauses $\mathbf{C} = \{ \mathbf{c}_1, \ldots, \mathbf{c}_k \}$ with respect to f ($\mathrm{CE}_f(\mathbf{C})$) is given by:

$$\mathrm{CE}_f(\mathbf{C}) = \{ \mathbf{p} \in \mathrm{PC}(f) \mid \exists\, \mathbf{c}_i \in \mathbf{C} \text{ with } \mathbf{c}_i \leq \mathbf{p} \}$$

$$A(f,g) = \bigvee_{\mathbf{m} \in \mathrm{IE}_f(\,\mathrm{PI}(g)\,)} \mathbf{m}$$

$$B(f,g) = \bigwedge_{\mathbf{c} \in \mathrm{CE}_f(\,\mathrm{PC}(g)\,)} \mathbf{c}$$

□

Theorem 5.2)

Let f, g be monotone boolean functions. Then:

1) $A(f,g)$ is the unique minimal function s such that $g \overset{f}{=\!\!\mid} s$

2) $B(f,g)$ is the unique maximal function s such that $g \overset{f}{=\!\!\mid} s$

Note: Conventionally the empty monom (clause) is 1 (0).

Proof

1) Certainly $g \overset{f}{=\!\!\mid} A(f,g)$, as by definition $\mathrm{IE}_f(\,\mathrm{PI}(g)\,)$ is the set of all prime implicants of f to which g could be extended. So suppose some function, s, exists, also satisfying these requirements and that $A(f,g) \not\leq s$ There must be some prime implicant of $A(f,g)$, $\mathbf{p}$ say, which is not an implicant of s. Now:

$$\mathbf{p} \in \mathrm{PI}(f) \quad \text{and} \quad \exists\, \mathbf{m} \in \mathrm{PI}(g) \text{ such that } \mathbf{p} \leq \mathbf{m}$$

So; $g \wedge \mathbf{p} = \mathbf{p} \in \mathrm{PI}(f)$

But; $s \wedge \mathbf{p} < \mathbf{p}$

Contradiction, as g is not always replaceable by s when computing f. (viz. Fig(5.1))

Thus $A(f,g)$ is a minimal function. To establish uniqueness, suppose s_1, s_2 are distinct i.e incomparable minimal functions. Then, since $g \overset{f}{=\!\!\mid} g$

$$g \overset{f}{=\!\!\mid} g \wedge g \;\; => \;\; g \overset{f}{=\!\!\mid} s_1 \wedge s_2$$

Thus $s_1 \wedge s_2$ ($< s_1, s_2$) is also a suitable, but smaller function. Contradiction.

2) By duality

□

Corollary 5.1)

$g \overset{f}{=\!\!\mid} h$ if and only if:

$$A(f,g) \leq h \leq B(f,g)$$

□

5.4) Replaceability By Arbitrary Functions (II)

Definition 5.4)

$PI_{rem}(f,g) = PI(f) - IE_f(PI(g))$

$PC_{rem}(f,g) = PC(f) - CE_f(PC(g))$

$$D(f,g) = \bigwedge_{m \in PI_{rem}(f,g)} \chi(m)$$

$$E(f,g) = \bigvee_{c \in PC_{rem}(f,g)} \varphi(c)$$

□

Theorem 5.3)

(A) $D(f,g)$ is the unique maximal s such that $s \overset{f}{=\!|} g$.

(B) $E(f,g)$ is the unique minimal s such that $s \overset{f}{=\!|} g$.

Proof

Again (B) will follow from (A) by duality.

1) $D(f,g) \overset{f}{=\!|} g$

By Cor(5.1) since $A(f,D(f,g)) \leq D(f,g) \leq B(f,D(f,g))$ it is sufficient to show that:

$$A(f,D(f,g)) \leq g \leq D(f,g)$$

(i) $g \leq D(f,g)$

Let $p \leq g$. Then by definition of $PI_{rem}(f,g)$:

$$\neg\exists m' \text{ such that } p \wedge m' \in PI_{rem}(f,g)$$

Thus:

$$\forall\, m \in PI_{rem}(f,g) \quad p \leq \chi(m)$$

By the definition of $D(f,g)$ this implies the result.

(ii) $A(f,D(f,g)) \leq g$

Let: $p \in PI(\, A(f,D(f,g))\,)$. Now:

$$IE_f(\, PI(D(f,g))\,) \cap IE_f(\, PI_{rem}(f,g)\,) = \{\}$$

(Since, from the proof of Th(5.1), no implicant of $D(f,g)$ can be "extended" to a member of $PI_{rem}(f,g)$)

Thus either $p \in PI(f)$, in which case p is a lengthening of some prime implicant m of g or $p=0$. In both cases $p \leq g$.

2) $D(f,g)$ is maximal

Suppose $s \nleq D(f,g)$ and $s \overset{f}{=\!|} g$. Then

$$\exists\, p \in PI(s) \text{ such that } p \nleq D(f,g)$$

Thus: $D(f,g) \leq \chi(p)$ and therefore,

$\exists\, r \in PI_{rem}(f,g)$ such that $\chi(r) \leq \chi(p)$
(By the definition of D and PI_{rem})

So $r \leq p$. Thus:

$s \wedge r = r \in PI(f)$
$g \wedge r \neq r$ (As $r \in PI_{rem}(f,g)$)
Thus s is not g-replaceable with respect to f. (cf Theorem(5.2))

Contradiction.

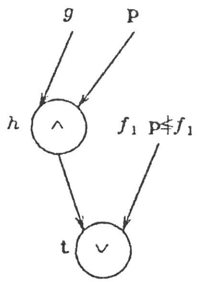
Uniqueness follows easily since: $s_1 \overset{f}{=\!|} g$ and $s_2 \overset{f}{=\!|} g$ implies that $s_1 \vee s_2 \overset{f}{=\!|} g$. □

Corollary 5.2)

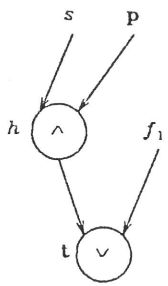
$h \overset{f}{=\!|} g$ if and only if:

$$E(f,g) \leq h \leq D(f,g)$$

□



$RES(\mathrm{t}) = f$; $RES(h) = \mathrm{p}$ *before replacement*



$RES(\mathrm{t}) \neq f$ after replacement since **p** is not a prime implicant of $RES(\mathrm{t})$

Figure (5.1)

Beynon [4] has considered a concept of "computational equivalence" within a different framework. g is said to be equivalent to h when computing f ($g \overset{f}{\models\!\dashv} h$) if and only if $g \overset{f}{=\!\!\dashv} h$ and $h \overset{f}{=\!\!\dashv} g$. In this context Cor(5.1) and Cor(5.2) yield:

Theorem 5.4

$g \overset{f}{\models\!\dashv} h$ iff

$$A(f,g) \vee E(f,g) \leq h \leq B(f,g) \wedge D(f,g)$$

□

5.5) Multiple Output Functions

Let $F = \{ f_1,\dots,f_m \}$ be any set of m monotone boolean functions over X_n.

Theorem 5.5)

$Z(F) = \bigwedge_{i=1}^{m} Z(f_i)$ is the maximal 0–replaceable function w.r.t F

$U(F) = \bigvee_{i=1}^{m} U(f_i)$ is the minimal 1–replaceable function w.r.t. F

$A(F,g) = \vee IE_F(PI(g))$ is the unique minimal function s such that g $F\!=\!|$ s

$B(F,g) = \wedge CE_F(PC(g))$ is the unique maximal function s such that g $F\!=\!|$ s

$D(F,g) = \bigwedge_{p \in \{\bigcup_{i=1}^{m} PI(f_i)\} - \{PI(g)\}} \chi(p)$ is the unique maximal s such that s $F\!=\!|$ g.

$E(F,g) = \bigvee_{c \in \{\bigcup_{i=1}^{m} PC(f_i)\} - \{PC(g)\}} \varphi(c)$ is the unique minimal function s such that s $F\!=\!|$ g

where;

$$IE_F(M) = \{p \in \bigcup_{i=1}^{m} PI(f_i) \mid \exists m_i \geq p\}$$

$$CE_F(C) = \{r \in \bigcup_{i=1}^{m} PC(f_i) \mid \exists c_i \leq r\}$$

Proof

Elementary

□

As an illustration we reprove the replacement rule due to Paterson [36] for Boolean Matrix Product. Let:

$BMP_n \ \{0,1\}^{2n^2} \text{->} \{0,1\}^{n^2}$

where each output c_{ij} is defined by:

$$c_{ij} = \bigvee_{1 \leq k \leq n} (x_{ik} \wedge y_{kj})$$

Lemma 5.1) (Paterson [36])

Let $BMP_n = \{ c_{11}, \ldots, c_{nn} \}$, where c_{ij} is as defined above. Let $1 \le i, i' \le n$ $(i \ne i')$ and $1 \le j, j' \le n$ $(j \ne j')$.

3.1) $x_{ik} \vee x_{i'k} \overset{BMP_n}{=\mid} 1$

3.2) $y_{kj} \vee y_{kj'} \overset{BMP_n}{=\mid} 1$

3.3) $x_{ik} \vee y_{kj} \overset{BMP_n}{=\mid} 1$

Proof

$$U(c_{ij}) = \widehat{Z(\hat{c}_{ij})}$$

$$= \widehat{Z\Big(\bigwedge_{1 \le k \le n} (x_{ik} \vee y_{kj}) \Big)}$$

$$= \widehat{\bigvee_{\substack{1 \le p \ne i \le n \\ 1 \le q \le n}} x_{pq} \vee \bigvee_{\substack{1 \le r \le n \\ 1 \le s \ne j \le n}} y_{rs} \vee c_{ij}}$$

$$= \bigwedge_{\substack{1 \le p \ne i \le n \\ 1 \le q \le n}} x_{pq} \wedge \bigwedge_{\substack{1 \le r \le n \\ 1 \le s \ne j \le n}} y_{rs} \wedge \hat{c}_{ij}$$

Thus:

$$U(\, BMP_n \,) = \bigvee_{\substack{1 \le i \le n \\ 1 \le j \le n}} (\wedge\, x_{pq}\, y_{rs}\, \hat{c}_{ij})$$

It is easy to see that for each of the function s in (3.1)-(3.3):

$$U(BMP) \vee s = s \;\; => \;\; U(BMP) \le s$$

□

A k-slice function of f is a function of the form:

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

The following is due to Wegener[56].

Lemma 5.2)

If g is a k-slice function then:

$$\vee\, x_i \in X_n: \; x_i \overset{g}{=\mid} x_i \wedge T_k^n(X_n)$$

Proof

Easily derived from Theorem(5.2) above. □

Corollary 5.3)

If g is a k-slice function then:

$$\vee\, x_i \in X_n: \; x_i \overset{g}{=\mid} x_i \vee T_{k+1}^n(X_n)$$

Proof

Duality. □

Chapter 6

A Replacement Rule For Non-Monotone Networks

6.1) Introduction

The results of Chapter(5) characterise all valid replacement rules for functions computed by monotone networks. In this chapter we examine a class of replacements which transform combinational networks, computing $f \in M_n$ to monotone networks. The work below is motivated by the results of Berkowitz [3] on realising "slice functions", which have been discussed briefly in the Introduction.

Definition 6.1)

Let $f \in M_n$ and let $1 \le k \le n$. The *k-slice of f* (denoted f_k) is the monotone boolean function:

$$f_k = (f \wedge T^n_k \vee) T^n_{k+1}$$

Fact 6.1) (Berkowitz)

Let $f \in M_n$

i) $C(f) \le \sum_{k=1}^{n} C(f_k) + O(n)$

ii) $C(f_k) \le C(f) + O(n)$

iii) $C^m(f_k) \le C^m(f) + O(n \log n)$

iv) $C^m(f_k) \le O(C(f_k) + n \log^2 n)$

Proof

i) Let $E^n_k(X_n)$ be the boolean function which is 1, when exactly k inputs are 1.

Since:

$$f(X_n) = \bigvee_{k=1}^{n} f \wedge E^n_k \text{ ; and } C(T^n_1,\ldots,T^n_n) = O(n)$$

and $f \wedge E^n_k = f_k \wedge (\neg T^n_{k+1})$, (i) follows.

ii) $C(T^n_k) = O(n)$

iii) $C^m(T^n_k) = O(n \log n)$ [1]

iv) (Outline)

Any combinational network T_0 can be changed to an $\{\wedge,\vee,\neg\}$-network T_1 in which negation is applied only to the network inputs. $C(T_1)$ is at most a constant multiple of $C(T_0)$. Berkowitz proved that in such networks computing f_k, any instance of $\neg x_i$ could be replaced by $T^{n-1}_k(X_n - \{x_i\})$. As the n-output monotone function:

$$\{ T^{n-1}_k(X_n - \{x_1\}),\ldots, T^{n-1}_k(X_n - \{x_n\}) \}$$

can be computed in $O(n \log^2 n)$ gates [52], so:

$$C^m(f_k) \le O(C(f_k) + n \log^2 n)$$

□

A new proof that the replacement at the heart of (iv) is correct, is given later in this chapter.

Fact(6.1) establishes that $f \in M_n$ has combinational complexity $\omega(n \log^2 n)$ if some k-slice of f has monotone network complexity $\omega(n \log^2 n)$. In addition (i) shows that if the combinational complexity of f is sufficiently large then there must be some k-slice of f with large monotone network complexity.

These results raise two questions:

Q1) In $\{\wedge,\vee,\neg\}$-networks of the form above, computing *any* $f \in M_n$, do monotone functions which can replace $\neg x_i$ always exist?

Q2) Are there other classes of monotone functions, for which results similar to Fact(6.1) can be proved?

Below, we demonstrate that both these questions can be answered affirmatively. In Section(6.2) the existence of "pseudo-complements", for each $f \in M_n$ is proved, and these are characterised. Section(6.3) introduces a generalisation of slice functions and a result analogous to Fact(6.1) is proved.

6.2) Pseudo-Complementation

Definition 6.2)

Let f be a monotone boolean function over X_n. A *pseudo-complement* for x_i is a monotone boolean function h_i, such that in any $(\wedge, \vee, \neg)$-network T computing f, in which negation is applied only to the inputs, any instance of $\neg x_i$, can be replaced by h_i and the resulting network will still compute f.

Theorem 6.1)

$\forall$ monotone f, h_i is a pseudo-complement for x_i if and only if:

$$f^{|x_i:=0} \le h_i \le f^{|x_i:=1}$$

Proof

Let f_0 denote the function computed by T after some instance, z say, of $\neg x_i$ is replaced by $f^{|x_i=0}$. Similarly let f_1 denote the function computed by T after this instance, z, is replaced by $f^{|x_i=1}$. Now since $f_0 \le f_1$ it is sufficient to prove that:

$$f_1 \le f \le f_0$$

The (monotone) boolean function computed by T after some instance of $\neg x_i$ is replaced by z may be written as:

$$g_{000} \vee x_i g_{100} \vee \neg x_i g_{010} \vee z g_{001} \vee \neg x_i z g_{011} \vee x_i z g_{101}$$

where the functions $g_{\alpha\beta\gamma}$ are such that:

1) $\forall$ m $\in$ PI($g_{\alpha\beta\gamma}$):

$$(x_i)^{\alpha} (\neg x_i)^{\beta} (z)^{\gamma} \wedge \mathrm{m}$$

is a monom[1] computed at T.

[1] Here, the definition of "monom" from Defn(1.1.4) is extended in a natural way to allow occurrences of $\neg x_i$.

where:

$$(x)^{\delta} = \begin{cases} 1 & \text{if } \delta = 0 \\ x & \text{if } \delta = 1 \end{cases}$$

2) m does not depend on x_i, $\neg x_i$ or z.

Clearly:

$$f = g_{000} \vee x_i g_{100} \vee \neg x_i \, (g_{010} \vee g_{001} \vee g_{011})$$

Now let $z := f^{|x_i=0}$ so that $f := f_0$. To prove $f \leq f_0$, it need only be shown that:

$$\neg x_i g_{001} \vee \neg x_i g_{011} \leq f_0$$

However; $f^{|x_i=0} \wedge g_{001} = g_{001}$ and $\neg x_i \, f^{|x_i=0} \, g_{011} = \neg x_i \, g_{011}$ thus $f \leq f_0$.

Now consider the replacement $z := f^{|x_i=1}$. We must show that $f_1 \leq f$. Similarly we need only prove:

$$f^{|x_i=1} g_{001} \vee \neg x_i f^{|x_i=1} g_{011} \vee x_i f^{|x_i=1} g_{101} \leq f$$

But:

$$f^{|x_i=1} \wedge g_{001} \leq g_{001} \leq f$$
$$\neg x_i \wedge f^{|x_i=1} \wedge g_{011} \leq g_{011} \leq f$$
$$x_i \wedge f^{|x_i=1} \wedge g_{101} \leq f$$

Thus $f_1 \leq f$, and the theorem follows. □

Corollary 6.1)

Let $F = \{f_1, \ldots, f_m\}$ be a set of m monotone boolean functions. Then h_i is a pseudo-complement for x_i if and only if:

$$\bigvee_{j=1}^{m} f_j^{|x_i=0} \leq h_i \leq \bigwedge_{j=1}^{m} f_j^{|x_i=1}$$

□

We note that for sets of monotone boolean functions, in general the interval of Corollary(6.1) is not well-defined. However for special cases, such as slice functions, pseudo-complements exist in this case.

Fact 6.2)

Let $f \in M_n$. For all k-slices, f_k of f, $T_k^{n-1}(X_n - \{x_i\})$ is a pseudo-complement for $\neg x_i$.

Proof

$$\begin{aligned}(f_k)^{|x_i=0} &= (f^{|x_i=0}) \wedge T_k^{n-1}(X_n - \{x_i\}) \vee T_{k+1}^{n-1}(X_n - \{x_i\}) \\ &\leq T_k^{n-1}(X_n - \{x_i\}) \\ &\leq (f^{|x_i=1}) \wedge T_{k-1}^{n-1}(X_n - \{x_i\}) \vee T_k^{n-1}(X_n - \{x_i\}) \\ &= (f_k)^{|x_i=1}\end{aligned}$$

And Fact(6.2) follows from Theorem(6.1). □

The interval which occurs in Theorem(6.1) may be informally interpreted in terms of Theorem(5.2). Observe that since negation is applied to the network inputs only, the "behaviour" of the network **T** is monotone. Thus, if each instance of $\neg x_i$ is replaced by a new variable z_i, a monotone function of $\{x_1, \ldots, z_n\}$, is computed. To compute $f(X_n)$ correctly, each z_i must be replaced by a monotone function h_i with the properties:

C1) $0 \overset{f^{|x_i=1}}{=\mid} h_i$

C2) $1 \overset{f^{|x_i=0}}{=\mid} h_i$

(i.e. h_i "appears to be" 0 or 1 when x_i is 1 or 0)

From Theorem(5.2), h_i must therefore satisfy:

$$f^{|x_i=0} \vee 0 \leq h_i \leq f^{|x_i=1} \wedge 1$$

which is the interval of Theorem(6.1).

6.3) Dissecting Transforms

Definition 6.3)

Let:

$$\Pi_r = \{ \{X^{(1)}, X^{(2)}, \ldots, X^{(r)}\} \mid \{X^{(1)}, \ldots, X^{(r)}\} \text{ is a partition of } X \text{ into } r \text{ non-empty sets} \}$$

A *dissecting transform of order* r is a mapping Δ_r defined as follows:

D1) $\Delta_r : M_n \times \Pi_r \to \Delta \subset M_n$

D2) Let $f \in M_n$, $P = \{X^{(1)}, \ldots, X^{(r)}\} \in \Pi_r$. Then

$g \in \Delta_r(f, P)$ if and only if:

$$\exists (k_1, k_2, \ldots, k_r) \text{ with } 1 \le k_i \le |X^{(i)}| = n_i$$

such that:

$$g(X) = \begin{cases} 1 \text{ if } \bigvee_{i=1}^{r} T^{n_i}_{k_i+1}(X^{(i)}) = 1 \\ f(X) \text{ if } \bigwedge_{i=1}^{r} E^{n_i}_{k_i}(X^{(i)}) = 1 \\ 0 \text{ otherwise} \end{cases}$$

Note that $g \in M_n$ since:

$$g(X) = f \wedge \bigwedge_{i=1}^{r} T^{n_i}_{k_i}(X^{(i)}) \vee \bigvee_{i=1}^{r} T^{n_i}_{k_i+1}(X^{(i)})$$

If $\overleftrightarrow{K} = \{k_1, k_2, \ldots, k_r\}$ then g is called the $(\overleftrightarrow{K}, P, r)$−block$(f)$ □.

Within the context of Defn(6.3), "slice functions" correspond to the dissecting transforms of order 1. In this section we generalise Berkowitz' results, for slice functions, to dissecting transforms.

Theorem 6.2)

Let $f \in M_n$, $P = \{X^{(1)}, \ldots, X^{(r)}\} \in \Pi_r$ and:

$$\rho(P, r) = \bigcup_{\substack{1 \le k_1 \le n_1 \\ 1 \le k_2 \le n_2 \\ \cdots \\ 1 \le k_r \le n_r}} \{ \langle k_1, k_2, \ldots, k_r \rangle \}$$

where: $n_i = |X^{(i)}_n|$

Then:

d1)

$$C(f) \le \sum_{\check{K} \in \rho(P,r)} C((\check{K}, P, r)\text{−block}(f)) + (r+1)n_1 n_2 \ldots n_r + O(n)$$

d2)

$$C((\check{K}, P, r)\text{−block}(f)) \le C(f) + O(n)$$

d3)

$$C^m((\check{K}, P, r)\text{−block}(f)) \le C^m(f) + O(n \log n)$$

d4)

$$C^m((\check{K}, P, r)\text{−block}(f)) \le O(C((\check{K}, P, r)\text{−block}(f)) + n \log^2 n)$$

Proof

Below c_0 and c_1 denote constants.

d1)

$$f = \bigvee_{\check{K} \in \rho(P,r)} (f \wedge \bigwedge_{i=1}^{r} E^{n_i}_{k_i}(X^{(i)}_n))$$

But:

$$f \wedge \bigwedge_{i=1}^{r} E^{n_i}_{k_i} = ((\check{K}, P, r)\text{−block}(f)) \wedge \bigwedge_{i=1}^{r} \neg T^{n_i}_{k_i+1}$$

To compute all the required $\neg(T^{n_i}_{k_i+1})$ costs $\le \sum_{i=1}^{r} c_0 . n_i = O(n)$ gates. In addition, r $\wedge$-gates are used for each block, to form the product of the required $\neg(T^{n_i}_{k_i+1})$ outputs. Finally $n_1 n_2 \ldots n_r - 1$ $\vee$-gates are used to collect all these outputs. □

d2)

This relation follows from the fact that all the necessary threshold functions can be computed using a total of $O(n)$ gates.

d3)

Similarly, all the threshold functions can be computed by a monotone network of size $O(n \log n)$.

d4)

Let T_0 be an optimal combinational network computing the $(\check{K},P,r)$-block(f). Convert T_0 to an $\{\wedge,\vee,\neg\}$-network T_1 in which negation is applied to the network inputs alone. This involves at most a constant factor increase in network size. Now let:

$X_n^{(i)} = \{x_1^i, x_2^i, \ldots, x_{n_i}^i\}$ and:

$$h_q^i = T_{k_i}^{n_i-1}(X_n^{(i)} - \{x_q^i\}) \vee \bigvee_{1 \le j \ne i \le r} T_{k_j+1}^{n_j}(X_n^{(j)})$$

It may be verified that:

$$((\check{K},P,r)\text{-block}(f))^{|x_q^i=0} \le h_q^i \le ((\check{K},P,r)\text{-block}(f))^{|x_q^i=1}$$

Thus from Theorem(6.1), h_q^i, is a pseudo-complement for x_q^i.

All the h_q^i for $1 \le i \le r$, $1 \le q \le n_i$ may be computed using at most:

$$\le \sum_{i=1}^{r} C^m(h_1^i, \ldots, h_{n_i}^i) + \sum_{i=1}^{r} C^m(T_{k_i+1}^{n_i}(X_n^{(i)}))$$

$$\le \sum_{i=1}^{r} c_0 . n_i \log^2 n_i + \sum_{i=1}^{r} c_1 . n_i \log^2 n_i$$

$$= O(n \log^2 n)$$

Thus:

$$C^m((\check{K},P,r)\text{-block}(f)) \le O(C((\check{K},P,r)\text{-block}(f)) + n \log^2 n)$$

□

Chapter 7

Slice Functions Of NP-complete Predicates

7.1) Introduction

Fact(6.1) implies that $P \neq NP$ if some k-slice of some monotone boolean NP-complete predicate has superpolynomial monotone network complexity. However this result does not indicate which, if any, slice functions of such predicates are likely to be "hard" to compute. Consider the following class of slice functions.

Definition 7.1)

Let $f \in M_n$ such that:

$$\vee \ p \in \mathrm{PI}(f) \quad |var(\mathrm{p})| = k \quad \text{for some } 1 \leq k \leq n$$

The *canonical slice* function of f ($c\text{-}sl(f)$) is the k-slice.

It can be seen that:

$$c\text{-}sl(f) = f \vee T_{k+1}^{n}$$

for those $f \in M_n$ for which the canonical slice function is well defined. For a number of monotone boolean NP-complete functions the canonical slice exists, and since this slice function seems very similar to the original function, it appears to be the most natural candidate for a "hard" slice. However, Wegener, considering the complexity of $c\text{-}sl((n/2)\text{--}clique)$, proved:

Fact 7.1) (Wegener [56])

$$C^{m}(\, c\text{-}sl((n/2)\text{--}clique\,(X_n^{U}))\,) = O(N)$$

where $N = n(n-1)/2 = |X_n^{U}|$ $\Box$

In Section(7.2), below, it is proved that this is not an isolated case, but that the canonical slices of the directed & undirected hamiltonian circuit functions

also have polynomial network complexity, as does the canonical slice of the NP-hard predicate, *Permanent*. In addition we prove that if the canonical (k) slice of f can be realised in polynomially many gates, then so can the ($k+c$)-slice of f, for any fixed c.

In Section(7.3) the *central slice* function is defined. For each of the problems cited above, the corresponding central slice functions are shown to have polynomial network complexity *if and only if* the original functions have polynomial network complexity. These results are obtained by demonstrating that every slice of these functions is a projection of the central slice of a larger instance of the problem.

In Section(7.4) similar results are proved for the non-graph theoretic predicate *SATISFIABILITY*.

7.2) Upper Bounds On Some Canonical-Slice Functions

Definition 7.2)

Let $\Pi_n = \{\sigma \mid \sigma \text{ is a permutation of } \{1,...,n\}\}$

$$Permanent\,(\mathbf{X}_n^{\mathbf{D}}) = \bigvee_{\sigma \in \Pi_n} \bigwedge_{i=1}^{n} x_{(i,\sigma(i))}$$

where:

$$x_{i,\sigma(i)} = \begin{cases} 0 & \text{if } i = \sigma(i) \\ x_{i,\sigma(i)} & otherwise \end{cases}$$

In graph-theoretic terms, $Permanent\,(\mathbf{X}_n^{\mathbf{D}})$ is the predicate which is true if and only if the vertices of $\mathbf{G}(\mathbf{X}_n^{\mathbf{D}})$ can be covered by a set of simple non-overlapping directed cycles. Valiant has shown that *Permanent* is *NP*-hard. [49]

Lemma 7.1)

Let $N_u = n(n-1)/2$ and $N_d = n(n-1)$. Then:

C1) $\mathbf{C}^{\mathbf{m}}(\ c\text{-}sl\,(UHC(\mathbf{X}_n^{\mathbf{U}}))\ ) = O(N_u^2)$

C2) $\mathbf{C}^{\mathbf{m}}(\ c\text{-}sl\,(DHC(\mathbf{X}_n^{\mathbf{D}}))\ ) = O(N_d^2)$

C3) $\mathbf{C}^{\mathbf{m}}(\ c\text{-}sl\,(Permanent\,(\mathbf{X}_n^{\mathbf{D}}))\ ) = O(N_d)$

Proof

C1)

$$c\text{-}sl\,(\ UHC(\mathbf{X}_n^{\mathbf{U}})) = (UHC(\mathbf{X}_n^{\mathbf{U}}) \wedge T_n^{N_u}) \vee T_{n+1}^{N_u}$$

On the right hand side of this expression, we may substitute for $UHC(\mathbf{X}_n^{\mathbf{U}})$, any monotone function $g(\mathbf{X}_n^{\mathbf{U}})$ which agrees with $UHC(\mathbf{X}_n^{\mathbf{U}})$ when exactly n inputs are 1. We claim:

$$g_1(\mathbf{X}_n^{\mathbf{U}}) = \bigwedge_{i=1}^{n} T_2^{n-1}(\mathbf{X}_n^{(i)}) \wedge UCON(\mathbf{X}_n^{\mathbf{U}})$$

where;

$$\mathbf{X}_n^{(i)} = \{\ x_{(1,i)}, x_{(2,i)}, \ldots, x_{(i-1,i)}, x_{(i,i+1)}, \ldots, x_{(i,n)}\ \}$$

and:

$$UCON(\mathbf{X}_n^{\mathbf{U}}) = \begin{cases} 1 & \text{if } \mathbf{G}(\mathbf{X}_n^{\mathbf{U}}) \text{ is connected} \\ 0 & \text{otherwise} \end{cases}$$

is such a function. This follows easily from the fact that:

"Any undirected n-vertex graph **G**, having exactly n edges, contains a hamiltonian circuit if and only every vertex has at least two edges incident to it and **G** is connected."

Since: $\mathbf{C}^{\mathbf{m}}(\ UCON(\mathbf{X}_n^{\mathbf{U}})\ ) = O(N_u^2)$ the upper bound of (C1) follows.

C2)

Let:

$$g_2(\ \mathbf{X}_n^{\mathbf{D}}\ ) = \bigwedge_{i=1}^{n} (\ T_1^{n-1}(\mathbf{X}_n^{out-i}) \wedge T_1^{n-1}(\mathbf{X}_n^{in-i})\ ) \wedge DCON(\mathbf{X}_n^{\mathbf{D}})$$

where:

$$\mathbf{X}_n^{out-i} = \{x_{(i,1)}, \ldots, x_{(i,i-1)}, x_{(i,i+1)}, \ldots, x_{(i,n)}\}$$
$$\mathbf{X}_n^{in-i} = \{x_{(1,i)}, \ldots, x_{(i-1,i)}, x_{(i+1,i)}, \ldots, x_{(n,i)}\}$$

and $DCON(\mathbf{X}_n^{\mathbf{D}})$ is defined analogously to $UCON(\mathbf{X}_n^{\mathbf{U}})$ for directed graphs.

$g_2(\mathbf{X}_n^{\mathbf{D}})$ may be used in the same manner as $g_1(\mathbf{X}_n^{\mathbf{U}})$ in (C1) since:

"A directed n-vertex graph, $\mathbf{G}(\mathbf{X}_n^{\mathbf{D}})$, containing exactly n edges, has a directed hamiltonian circuit if and only if each vertex is incident to at least one incoming edge, and at least one outgoing edge and **G** is connected."

C3)

Let:

$$g_3(\ \mathbf{X}_n^{\mathbf{D}}\ ) = \bigwedge_{i=1}^{n} (\ T_1^{n-1}(\mathbf{X}_n^{out-i}) \wedge T_1^{n-1}(\mathbf{X}_n^{in-i})\ )$$

with $\mathbf{X}_n^{out-i}, \mathbf{X}_n^{in-i}$ as in (C2).

Similarly, from the graph-theoretic interpretation of *Permanent*, $g_3(X_n^D)$ may be used as a substituting function.

□

The following result establishes that all the slice functions "within a constant distance" of these canonical slices also have polynomial network complexity.

Theorem 7.1)

Let $f \in M_n$ such that $c\text{-}sl(f)$ exists and is the k-slice, f_k. If $C^m(f_k) = O(n^r)$ for some constant r, then:

$$\forall\ c \geq 0: \quad C^m(f_{k+c}) = O(n^{r+c})$$

Proof (By Induction on c)

Base $c = 0$

By the assumption that $C^m(f_k) = O(n^r)$.

Inductive Step

Assume the theorem holds for all values less than c.

$$f_{k+c}(X_n) = (f(X_n) \wedge T^n_{k+c}(X_n)) \vee T^n_{k+c+1}(X_n)$$

As before, we may substitute for f on the right-hand side, any function g which agrees with f when exactly $k+c$ inputs are 1.

Let $h_i : X_n \to \{0,1\}^n$ be defined by:

$$h_i(x_1,\ldots,x_n) = \begin{cases} \{x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n\} & \text{if } x_i = 1 \\ \{0,0,0,\ldots,0,0\} & \text{if } x_i = 0 \end{cases}$$

Then:

$$f_{k+c}(X_n) = ((\bigvee_{i=1}^{n} f_{k+c-1}(h_i(X_n))) \wedge T^n_{k+c}(X_n)) \vee T^n_{k+c+1}(X_n)$$

That is: $f(X_n) = 1$ when exactly $k+c$ inputs are true if and only if for some

$k+c-1$ size subset of the true inputs, f is 1 when exactly these $k+c-1$ inputs are true. This follows from the fact that all prime implicants, p of f have $|var(p)| = k$ and from the definition of prime implicant.

However:

$$h_i(X_n) = \{x_1 \wedge x_i,\ldots,x_{i-1} \wedge x_i, 0, x_{i+1} \wedge x_i,\ldots,x_n \wedge x_i\}$$

which can be computed using $n-1$ monotone gates. By the Inductive Hypothesis:

$$C^m(f_{k+c-1}(h_i(X_n))) = O(n^{r+c-1})$$

Thus:

$$C^m(f_{k+c}(X_n)) = O(n^{r+c})$$

□

7.3) Central Slice Functions

Definition 7.3)

Let $f \in M_n$. The *central slice* of f ($Cen(f)$) is the slice function:

$$Cen(f) = (f \wedge T^{n}_{\lceil n/2 \rceil}) \vee T^{n}_{\lceil n/2 \rceil+1}$$

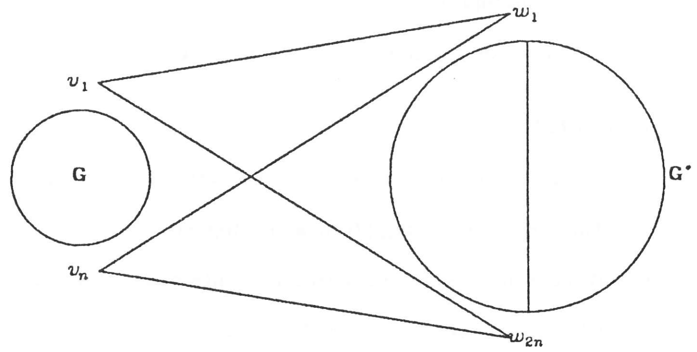
In this section it is proved that:

$$Cen((n/2)\text{-}clique\,(X_n^U))\ is\ NP\text{-}complete$$

and

$$Cen(DHC(X_n^D))\ is\ NP\text{-}complete$$

The proof methods are similar, but differ in the details of the constructions used. Both results rely heavily on the structure of the underlying predicates, and are derived by a padding argument. i.e If f_n is the $(n/2)$-clique or *DHC* function then for all valid k, the k-slice of f_n is a subfunction of $Cen(f_{rn})$ where $r=5$ for $(n/2)$-cliques and $r=7$ for *DHC*. Using Fact(6.1)(i) this yields a reduction from f_n to $Cen(f_n)$. Figure(7.1) informally illustrates the construction used for $(n/2)$-clique.



$V(G) = \{v_1, \ldots, v_n\}$; $V(G^*) = \{w_1, \ldots, w_{4n}\}$

Figure 7.1

Given $G(X_n^U)$, we extend it by adding $4n$ new vertices, to yield a graph H with the property that H contains a $(5n)/2$-clique if and only if $G(X_n^U)$ contains an $(n/2)$-clique. The number of additional edges in H must be chosen in such a way that $Cen(((5n)/2)\text{-}clique)$ is the k-slice of $(n/2)$-clique. Both the proofs are constructive in the sense that it is demonstrated how H can be built.

7.3.1) $(n/2)$–cliques

Below, we shall assume, wlog, that n is even.

Lemma 7.2)

Let $G^*(Y^U)$ be a $4n$-vertex graph with vertices $\{w_1,...,w_{4n}\}$, satisfying:

P1) The vertices $\{w_1,...,w_{2n}\}$ form a $2n$-clique in G^*.

P2) G^* contains α additional edges not contained in this clique.

P3) G^* does not contain any $((5n)/2)$-clique

If H is the $(5n)$-vertex graph formed from $G(X_n^U)$ and $G^*(Y^U)$ by adding the edges:

$$(v_i, w_j) \quad \forall\, v_i \in V(G),\ \forall\ 1 \le j \le 2n$$

Then:

$$((5n)/2)\text{–}clique\ (H) \iff (n/2)\text{–}clique\ (G)$$

Proof

<= From the construction of H

=> Since $G^*(Y^U)$ does not contain any $((5n)/2)$–*clique* any $((5n)/2)$–*clique* in H must contain at least one vertex from $V(G)$ and hence at most $2n$ vertices from G^*.

□

From Lemma(7.2) it follows that $(n/2)$–*clique* is a subfunction of $((5n)/2)$–*clique*.

Lemma 7.3)

Let H be an n-vertex graph $(n = 2m)$ such that:

H1) H contains an m-clique.

H2) H does not contain an $(m+1)$-clique

H3) $\forall\ H^*$ such that (H1) and (H2) hold for H^*:

$$|E(H^*)| \le |E(H)|$$

Then:

$$|E(H)| \ge |E(K_n)| - m$$

where K_n is the complete graph on n vertices.

Proof

Let $V(K_n) = \{w_1,...,u_n\}$. Let H be the graph formed by removing the m edges:

$$\bigcup_{i=1}^{m} \{(u_i, u_{n-i+1})\}$$

from K_n. Certainly H contains an m-clique, e.g the vertices $\{w_1,...,u_m\}$. But H cannot contain any $(m+1)$-clique as any subset $\{u_{i_1},...,u_{i_{m+1}}\}$ of $V(H)$ must contain two vertices u_i and u_j such that $i + j = n+1$ and these are not connected by an edge in H. Therefore:

$$|E(H)| \ge |E(K_n)| - m \qquad \square$$

Theorem 7.2

Let:

$$L = \{\frac{n}{4}(\frac{n}{2}-1),...,\frac{n}{2}(n-1)\}$$

For each $l \in L$

$$C^m(l\text{–}slice\ (\ (n/2)\text{–}clique\ )) = O(C^m(Cen\ (\ ((5n)/2)\text{–}clique\ )))$$

Proof

For the purpose of brevity, let:

$$e_u(n) = n(n-1)/2$$

and

$$l = e_u(n)/2 - k$$

We show that the l-slice of $(n/2)$-clique is a subfunction of $Cen\,((5n)/2)$-clique.

Let $G^*(Y^U)$ and H be the $4n$ and $5n$ vertex graphs described in the statement of Lemma(7.2). α, the number of additional edges in G^* must be chosen so that.

$$| E(H)-E(G) | = e_u(5n)/2 - e_u(n)/2 + k \qquad (1)$$

i.e so that the "threshold terms" are correctly set. Now:

$$| E(H)-E(G) | = | E(G^*) | + |\{ \textit{Edges between } G \text{ and } G^* \}|$$
$$= e_u(2n) + \alpha + 2n^2 \qquad (2)$$

Solving (1) and (2), for α, yields:

$$\alpha = 2n^2 + k$$

Now:

$$-\left\lfloor e_u(n)/2 \right\rfloor \le k \le \left\lceil e_u(n)/2 - e_u(n/2) \right\rceil$$

Thus:

$$\lfloor (7n+1)n/4 \rfloor \le \alpha \le \left\lceil 17n^2/8 \right\rceil$$

Let G^* be the $4n$-vertex graph constructed in Lemma(7.3). If β is the set of edges in G^* which are not contained in the $(2n)$-clique $\{w_1,...,w_{2n}\}$ then, from Lemma(7.3):

$$|\beta| = |E(K_{4n})| - 2n - e_u(2n)$$
$$= 6n^2 - 3n$$

Since $|\beta| > 2n^2 + k$ $\forall$ *valid* k α may be fixed by removing edges from β to yield the required slice function (the l-slice).

□

Corollary 7.1

$\exists$ *constant* q *such that* $C^m(\textit{Cen}((n/2)\textit{-clique})) = O(n^q)$
if and only if

$\exists$ *constant* r *such that* $C((n/2)\textit{-clique}) = O(n^r)$

Proof

<= By definition of the central slice function

=> Suppose $C^m(\textit{Cen}((n/2)\textit{-clique})) = O(n^q)$. From Theorem(7.2) we can construct a polynomial size monotone network S_k, for each k-slice of $(n/2)$-clique. From Fact(6.1), (i):

$$C((n/2)\textit{-clique}) \le \sum_{k=e_u(n/2)}^{e_u(n)} C^m(S_k) + O(n)$$
$$\le O(n^r) \textit{ for some constant } r$$

□

Corollary 7.2)

Cen$((n/2)$*-clique*$)$ is *NP*-complete.

Proof

Certainly *Cen*$((n/2)$*-clique*$) \in NP$. Theorem(7.2) and Corollary(7.1) yield a polynomial sized reduction from $(n/2)$-clique to *Cen*$((5n/2)$*-clique*$)$, since the graphs H and G^* used in the proof of Theorem(7.2) are both easily constructible.

□

An alternative proof of Corollary(7.2) is yielded by the following result, which also produces a more efficient construction of $(n/2)$-clique than is implied by Fact(6.1). We shall assume, for convenience, that n is a multiple of 4.

Lemma 7.4)

Let Y^U be the set of boolean variables encoding the edges of a $5n$ vertex undirected graph $H(Y^U)$.

$$(n/2)\textit{-clique}\,(X_n^U) \textit{ is a projection of } \textit{Cen}((5n)/2\textit{-clique}\,(Y^U))$$

Proof

The construction is similar to that of Theorem(7.2). Given an n-vertex undirected graph G, a $5n$-vertex undirected graph H is built with the following properties:

1) H contains a $(5n)/2$-clique if and only if G contains an $(n/2)$-clique.

2) $|E(H)| = e_u(5n)/2$

H consists of 3 graphs, connected as in Figure(7.2).

The graph $\bar{G}$ has vertex set $\{u_1,...,u_n\}$ and is the complement of G with respect to K_n. (i.e the graph such that $(u_i,u_j) \in E(\bar{G}) <=> (v_i,v_j) \notin E(G)$)

G' has vertex set $\{w_1,...,w_{3n}\}$, the vertices $\{w_1,...,w_{2n}\}$ forming a $2n$-clique. In addition G' contains $(7n^2+n)/4$ edges which are not part of this clique and G' does not contain a $(5n)/2$-clique. Finally there are edges:

$$(u_i, v_j) \quad \forall \; 1 \le i \le 2n \;,\; 1 \le j \le n$$

The existence of H for all pertinent n may be verified from Lemma(7.3).

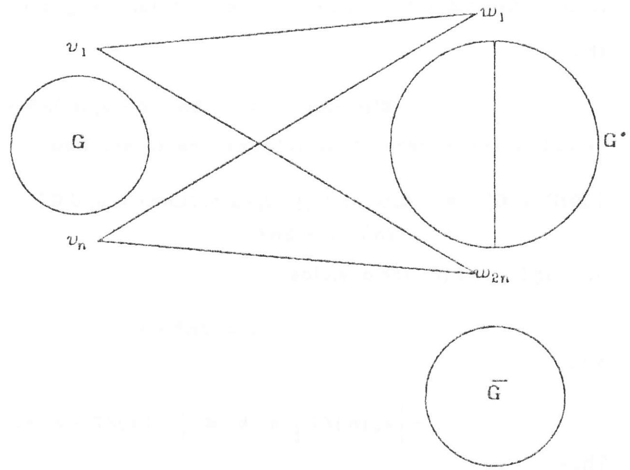


Figure 7.2

By observing that the graph $\bar{G}$ is disconnected from G and using the proof of Lemma(7.2) it is easy to see that H contains a $(5n)/2$-clique if and only if G contains an $(n/2)$-clique. H contains $|e_u(5n)/2|$ edges since:

$$\begin{aligned} |E(H)| &= |E(G)| + |E(\bar{G})| + |E(G')| + 2n^2 \\ &= e_u(n) + e_u(2n) + (7n^2+n)/4 + 2n^2 \\ &= e_u(5n)/2 \end{aligned}$$

Now consider $Cen((5n)/2\text{-}clique\,(Y^U))$. To compute $(n/2)$-clique (X_n^U) proceed by constructing the graph H and compute $Cen(E(H))$. Since $|E(H)| = e_u(5n)/2$:

$$\begin{aligned} Cen((5n/2)\text{-}clique\,(Y^U)) &<=> (5n/2)\text{-}clique\,(Y^U) \\ &<=> (n/2)\text{-}clique\,(X_n^U) \end{aligned}$$ □

7.3.2) Directed Hamiltonian Circuit

Lemma 7.5)

Let G^* be a $6n$-vertex directed graph with vertices $\{w_1,...,w_{6n}\}$, satisfying:

Q1) $\forall\ 1 \le i \le n-1, 1 \le j \le 6n$

$$(u_i, u_j) \notin E(G^*)\ ;\ (u_j, u_i) \notin E(G^*)$$

Q2) There is a directed hamiltonian path connecting the vertices $\{u_n,...,w_{6n}\}$ which commences in u_n and terminates in w_{6n}.

Q3) G^* contains α edges other than those in the hamiltonian path.

If $H(X^D_{7n})$ is the $(7n)$-vertex graph formed from $G(X^D_n)$ and G^* by adding the edges:

E1) $(v_i, u_i)\ \forall\ v_i \in V(G)$

E2) $(u_i, v_{i_k})\ \forall\ v_{i_k} \in \Gamma^+(v_i)\ \ 1 \le i \le n-1$

$(w_{6n}, v_{n_k})\ \forall\ v_{n_k} \in \Gamma^+(v_n)$

Then:

$$DHC(\ H(X^D_{7n})\ ) <=> DHC(\ G(X^D_n)\ )$$

Proof

Let: $e_d(n) = n(n-1)$, $l = e_d(n)/2 - k$ and let:

$$\Gamma^+(v_i) = \{v_j \in V(G)\ |\ (v_i, v_j) \in E(G)\}$$

<= By construction of $H(X^D_{7n})$

=> Consider any directed hamiltonian circuit in $H(X^D_{7n})$. From the construction, all the edges (v_i, u_i) must be in this circuit and for each $1 \le i \le n-1$ some edge $(u_i, v_{i_j} \in \Gamma^+(v_i))$ must be in the circuit. For the vertices $\{u_n,...,w_{6n}\}$ there must be a segment of the circuit which corresponds to a hamiltonian path through $\{u_n,...,w_{6n}\}$ in G^*, which path begins in u_n and ends in w_{6n}. Finally there must be an edge $(w_{6n}, v_{n_j} \in \Gamma^+(v_n))$. It is easy to see that replacing each pair of edges:

$$(v_i, u_i), (u_i, v_{i_j})\ by\ (v_i, v_{i_j}) \in E(G)\ for\ each\ 1 \le i \le n-1$$

$$(v_n, u_n), (w_{6n}, v_{n_j})\ by\ (v_n, v_{n_j}) \in E(G)$$

yields a directed hamiltonian cycle in $G(X^D_n)$.

□

Theorem 7.3

Let:

$$L = \{ n, n+1, ..., n(n-1) \}$$

Then:

$\forall\ l \in L$

$$C^m(\ l\text{-}slice\ (\ DHC(\ X^D_n\ )\ )) = O(\ C^m\ (\ Cen\ (DHC(\ X^D_{7n}\ ))))$$

Proof

Let H and G^* be the graphs constructed in the statement of Lemma(7.5). This lemma establishes that $DHC(X^D_n)$ is a subfunction of $DHC(X^D_{7n})$

As before α, the number of extra edges in G^*, must be chosen to set the "threshold terms" correctly. i.e So that:

$$|E(H)-E(G)| = e(7n)/2 - e(n)/2 + k$$

Now from the construction of $H(X^D_{7n})$:

$$|E(H)-E(G)| = 6n + \alpha + \sum_{v_i \in V(G)} |\Gamma^+(v_i)|$$

$$= 6n + \alpha + |E(G)|$$

Solving (for α) yields:

$\alpha = 24n^2 - 9n + k - |E(G)|$

By observing that:

$$-(n^2-n)/2 \leq k \leq (n^2-3n)/2$$

$$0 \leq |E(G)| \leq n^2-n$$

It follows that:

$$0 < \alpha \leq \lceil 24.5n^2-10.5n \rceil$$

If β is again the set of possible extra edges (which in this case must connect only vertices (u_i, u_j) where both i and j satisfy $n \leq i\, j \leq 6n$), then:

$$| \beta | = e_d(5n+1) - 5n$$
$$= 25n^2$$
$$> \textit{maximum value of } \alpha$$

The theorem follows.

□

Corollary 7.3

$$C^m(Cen(DHC(X_n^D))) = O(n^q) <=> C(DHC(X_n^D)) = O(n^r) \quad q, r \textit{ constants}$$

Proof

<= By definition of the central slice function.

=>

1) Let $S_n, S_{n+1}, \ldots, S_{n^2-n}$ be distinct monotone networks computing $Cen(DHC(X_{7n}^D))$.

2) Use the network S_k to compute the k-slice of $DHC(X_n^D)$ for a k-edge graph.

3) Compute for each k (where $n \leq k \leq n^2-n$)

$$g_k = RES(S_k) \wedge E_k^{e_d(n)}(X_n^D)$$

4) Then:

$$DHC(X_n^D) = \bigvee_{k=n}^{e_d(n)} g_k(X_n^D)$$

The total number of gates used is clearly polynomial in n.

□

Corollary 7.4)

$Cen(DHC(X_n^D))$ is NP-complete.

Proof

Again $Cen(DHC(X_n^D)) \in NP$. Theorem(7.3) and Corollary(7.3) give a polynomial sized reduction from $DHC(X_n^D)$ to $Cen(DHC(X_{7n}^D))$.

□

Both the results below are derivable from similar constructions.

Theorem 7.4)

$Cen(Permanent(X_n^U))$ *is* NP*-hard*

$Cen(UHC(X_n^U))$ *is* NP*-complete*

□

7.4) SATISFIABILITY

Definition 7.4)

Let $P = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an m clause conjunctive normal form where each clause C_i is a disjunction of some subset of the literals:

$$Z = \{z_1, z_2, \ldots, z_n, \neg z_1, \ldots, \neg z_n\}$$

P is *satisfiable* if $\exists\ \pi \in \{0,1\}^n$ such that $P^{|\{z_1,\ldots,z_n\}=\pi} = 1$.

In this section we consider the *NP*-complete problem, *SATISFIABILITY* (*SAT*) of determining whether a given conjunctive normal form, as above, is satisfiable. Results similar to Lemma(7.1) and Theorem(7.2) etc are proved for this predicate.

The definition below illustrates how *SAT* may be encoded as a monotone boolean function with $2nm$ inputs.

Definition 7.5) (From Valiant [50])

Let:

$$X_{nm} = \{x_{11}, \ldots, x_{nm}, y_{11}, \ldots, y_{nm}\}$$

be a set of $2nm$ boolean variables. X_{nm} defines an m clause *CNF* formula $P(X_{nm})$ over the set of literals Z as follows:

$$P(X_{nm}) = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$
$$z_i \text{ is a literal in } C_j \iff x_{ij} = 1$$
$$\neg z_i \text{ is a literal in } C_j \iff y_{ij} = 1$$

Thus:

$$SAT(X_{nm}) = \begin{cases} 1 & \text{if } P(X_{nm}) \text{ is satisfiable} \\ 0 & \text{otherwise} \end{cases}$$

Lemma 7.6)

$c\text{-}sl(SAT(X_{nm}))$ exists and is the m-slice

Proof

Let **p** be a monom over X_{nm} such that $|var(\mathbf{p})| < m$. Consider the *CNF*, P, that arises by setting the variables of $var(\mathbf{p})$ to 1 and the remaining variables of X_{nm} to 0. Then:

$$P = C_1 \wedge \ldots \wedge C_m$$

Since $|var(\mathbf{p})| < m$, there must exist some clause C_i of P which contains no literals, and so $C_i = 0$ (cf Theorem(5.2)). Thus P is not satisfiable and **p** cannot be an implicant of $SAT(X_{nm})$. Therefore:

$$\forall\ \mathbf{p} \in PI(SAT(X_{nm}))\quad |var(\mathbf{p})| \geq m$$

Now consider any implicant **p** of $SAT(X_{nm})$ having $|var(\mathbf{p})| > m$. It will be shown that there exists a proper subset of $var(\mathbf{p})$ which is also an implicant of $SAT(X_{nm})$, establishing that all prime implicants have size m. Again, let P be the *CNF* defined by setting the variables of **p** equal to 1 and the remaining variables of X_{nm} to 0. The clauses of P may be partitioned into 2 sets:

$$Triv(P) = \{C_i \mid C_i = z_j \vee \neg z_j \vee \ldots\} \text{ for some } 1 \leq j \leq n$$
$$NTriv(P) = \{\{C_1, \ldots, C_m\} - Triv(P)\}$$

wlog assume that $NTriv(P) = \{C_1, C_2, \ldots, C_q\}$ where $q \leq m$.

Consider any assignment π to Z which satisfies P (one must exist since **p** is an implicant of $SAT(X_{nm})$). Let $\{w_1, \ldots, w_q\}$. be the set of literals such that w_i satisfies clause C_i under π. If $\mu(w_i)$ is the corresponding element of X_{nm}, then clearly:

$$M = \{\mu(w_1), \ldots, \mu(w_q)\} \subset var(\mathbf{p})$$

If $Triv(P) \neq \{\}$ proceed as follows to extend M to a set of size m.

For each clause $C_j = (z_i \vee \neg z_i \vee \ldots)$ in $Triv(P)$, if $z_i = 1$ under π then add x_{ij} to M otherwise add y_{ij} to M.

Clearly M is still a subset of $var(p)$ and $|M| = m$ but:

$$\bigwedge_{x_{ij} \in M} x_{ij} \wedge \bigwedge_{y_{ij} \in M} y_{ij} \leq SAT(X_{n_n})$$

it follows that: $\forall$ p $\in PI(SAT(X_{n_n}))$ $|var(p)| = m$.

□

Lemma 7.3)

Let $N = 2nm$

$$C^m(c\text{-}sl(SAT(X_{nm}))) = O(N \log N)$$

Proof

$$c\text{-}sl(SAT(X_{nm})) = (SAT(X_{nm}) \wedge T_m^N) \vee T_{m+1}^N$$

As before, a substituting function $g(X_{nm})$ is defined, which agrees with $SAT(X_{nm})$ when exactly m inputs are true. Let:

$$g(X_{nm}) = \bigwedge_{i=1}^{m} (T_1^{2n}(XC_i) \vee \neg \bigvee_{i=1}^{n} (T_1^m(XZ_i) \wedge T_1^m(YZ_i))$$

where:

$$XC_i = \{x_{1i}, \ldots, x_{ni}, y_{1i}, \ldots, y_{ni}\}$$
$$XZ_i = \{x_{i1}, \ldots, x_{im}\}$$
$$YZ_i = \{y_{i1}, \ldots, y_{im}\}$$

The correctness of this substitution follows from the fact that:

"A CNF, P, with m clauses and exactly n literals is satisfiable if and only if each clause of P contains at least one literal, and should z_j be a clause of P then $\neg z_j$ is not a clause of P"

That is P is satisfiable if and only if P is a (non-zero) monom.

Note that in contrast to the previous examples $g(X_{nm})$ is *non-monotone*, however since a slice function is being computed, the translation of Fact(6.2) may be applied giving the claimed monotone network complexity.

□

The central slice of $SAT(X_{nm})$ is also NP-complete, as demonstrated by:

Theorem 7.5)

Let $L = \{m, m+1, \ldots, 2nm\}$.

$\forall\, l \in L$

$$C^m(l\text{-}slice(SAT(X_{nm}))) = O(C^m(Cen(SAT(X_{3n4m}))))$$

Proof

Let $l = nm - k$. For this construction given a m-clause CNF, P over Z, a new $4m$-clause CNF, Q over Z $\cup$ U is built, where U is a set of $4n$ new literals and:

$$P \textit{ is satisfiable} \iff Q \textit{ is satisfiable}$$

Let P^* be the $3m$-clause CNF over U defined as follows:

P1) $\forall$ clauses C_i^* of P^*, u_1 is a literal in C_i^*

P2) P^* contains α literals in addition to the u_1 literals.

Q is the CNF $P \wedge P^*$.

We claim that:

$$P \textit{ is satisfiable} \iff Q \textit{ is satisfiable}$$

For:

=> By the construction, since $u_1 = 1$ satisfies P^*

<= If Q is satisfied by some assignment, π, then every clause of Q is satisfied by π. Hence P is satisfiable, since the variables of P are disjoint from those of P^*.

Again α is chosen to project from $Cen(SAT(X_{3n4m}))$ onto $l\text{-}slice(SAT(X_{nm}))$, i.e so that:

$$nm - k = 12nm - 3m - \alpha$$

Thus:

$$\alpha = 11nm - 3m + k$$

Let β be the set of unused literals available over every clause, $|\beta| = 12nm - 3m$.

Since

$$-nm \leq k \leq nm - m$$

It follows that:

$$10nm - 3m \leq \alpha \leq 12nm - 4m \leq |\beta|$$

Thus $l\text{-}slice\,(SAT(X_{nm}))$ is a subfunction of $Cen\,(SAT(X_{3n\,4m}))$ and the theorem follows.

□

Corollary 7.5)

$$Cen\,(SAT(X_{nm}))\ is\ NP\text{-}complete$$

□

Chapter 8

Monotone Boolean Functions With Equal Combinational And Monotone Network Complexity

8.1) Introduction

The results of Berkowitz and their subsequent development by Wegener [56], leave open the possibility that a monotone function f with large monotone complexity, may have an efficient realisation by a combinational network, i.e. if f has large monotone complexity this may not imply that some slice of f is hard. The construction of "pseudo-complements" in Chapter(6), giving transformations between combinational and monotone networks for arbitrary monotone functions, appears to do little to remove this difficulty.

In this chapter we define a natural series of classes $Q_{(n,r)}$, of monotone boolean functions and show that for the "hardest" members of each class there is no asymptotic gap between the combinational and monotone complexity measures. For the special case $r = 2$, we obtain the stronger result that *for all* members of the class no such gap exists.

The remainder of this chapter is organised as follows: In Section(8.2) some preliminary results are stated. In Section(8.3) the classes $Q_{(n,r)}$ are defined and the results stated above proved. In Section(8.4) we consider the extension of our results to multiple output functions and to a wider class of monotone boolean functions.

8.2) Preliminary Results

Fact 8.1) (Improvement of Berkowitz [29])

Let k be constant

$$C^m(T_k^{n-1}(X_n-\{x_1\},\ldots,T_k^{n-1}(X_n-\{x_n\}) \leq O(n)$$

□

Fact 8.2) (Shannon [45], see Paterson [35])[1)]

Let $H \subset M_n$. Then for almost all $h \in H$:

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

□

1) This result is more usually stated in terms of $H = B_n$ or $H = M_n$. It may be easily verified that the formulation above follows from the proof in [35].

8.3) Main Result

Definition 8.1)

Let $r \in N$:

$$Q_{(n,r)} = \{ f \in M_n \mid \forall\ p \in PI(f),\ |var(p)| = n-r \}$$

Lemma 8.1)

For almost all $f \in Q_{(n,r)}$:

$$C(f) = \Omega\left(\frac{n^r}{r logn}\right)$$

Proof

The number of distinct monoms of size $n-r$ over $X_n = \{x_1,\ldots,x_n\}$ is $\Omega(n^r)$, since r is fixed. Thus:

$$|Q_{(n,r)}| = 2^{\Omega(n^r)}$$

and the Lemma follows from Fact(8.2). □

Theorem 8.1)

Let $f \in Q_{(n,r)}$ for some fixed r.

$$C(f) = O(n^{r-1}) \iff C^m(f) = O(n^{r-1})$$

Proof

<= Trivial.

=> Let T be an optimal combinational network computing f. Convert T to an $\{\wedge,\vee,\neg\}$-network T_0 with negation restricted to the network inputs. Then:

$$C(T_0) = O(n^{r-1})$$

Observe that:

$$f = f \wedge T^n_{n-r} = (f \wedge T^n_{n-r}) \vee T^n_n$$

Now proceed as follows:

S1) Compute $(f \wedge T^n_{n-r}) \vee T^n_{n-r+1}$ by a $\{\wedge, \vee, \neg\}$-network, T_1.

$$C(T_1) \leq C(T_0) + O(n) = O(n^{r-1})$$

S2) Construct a monotone network S_0, computing the n-output function:

$$\{ T^{n-1}_r(X_n - \{x_1\}), \ldots, T^{n-1}_r(X_n - \{x_n\}) \}$$

$$C^m(S_0) = O(n)$$

From Fact(6.1)

S3) Dualise S_0 so that it computes:

$$\{ T^{n-1}_{n-r}(X_n - \{x_1\}), \ldots, T^{n-1}_{n-r}(X_n - \{x_n\}) \}$$

S4) Combine the networks S_0 and T_1 by replacing each input $\neg x_i$ of T_1 by the output $T^{n-1}_{n-r}(X_n - \{x_i\})$ of S_0. From Fact(6.2), this new *monotone* network S_1 computes:

$$(f \wedge T^n_{n-r}) \vee T^n_{n-r+1}$$

and $C^m(S_1) = O(n^{r-1})$.

It remains to construct S computing f from S_1, using $O(n^{r-1})$ monotone gates. That this may be done follows from the result below:

Claim 6.1)

Let S_q be an optimal monotone network computing:

$$(f \wedge T^n_{n-r} \vee)\, T^n_{n-r+q} \qquad 1 \leq q \leq r$$

Then:

$$C^m(S_q) = O(n^{r-1})$$

Proof

By Induction on q.

Base q=1

Follows from Steps (S1)-(S4) above.

Inductive Step

We shall assume the claim holds for all values $< q \leq r$ and prove that it holds for q. Thus let S_{q-1} be an optimal network computing:

$$(f \wedge T^n_{n-r} \vee)\, T^n_{n-r+q-1}$$

We may express the function computed by S_{q-1} as:

$$(f \wedge T^n_{n-r}) \vee p_1 \vee p_2 \vee \ldots \vee p_t$$

where for all p_i: $p_i \nleq f \wedge T^n_{n-r} = f$

Let $\chi(p_i) = \vee \{ x \in X_n \mid p_i \nleq x \}$

We claim that:

$$\forall\, m \in PI(f),\ \forall\, p_i \quad m \leq \chi(p_i)$$

To see this observe that $var(m) \not\subset var(p_i)$, thus $\exists\, x \in X_n$ such that:

$$m \leq x \leq \chi(p_i)$$

S_q is the network which computes:

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

which evaluates to:

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

To compute a single $\chi(p_i)$ costs $r-q+1$ gates. To compute all the $\chi(p_i)$ and their conjunction costs $t\,(r-q+1) + t$ monotone gates. But $r-q+1$ is a constant and:

$$t \leq \binom{n}{n-r+q-1}$$
$$\leq \binom{n}{r-q+1}$$
$$\leq O(n^{r-1})$$

since $q \geq 2$. Therefore:

$$C^m(S_q) \leq C^m(S_{q-1}) + O(n^{r-1})$$

and the claim follows by the inductive hypothesis. □

By repeatedly applying the construction of Claim(8.1) to S_1 we obtain a network S_r which computes:

$$(f \wedge T^n_{n-r}) \vee T^n_n = f$$
$$\text{and } C^m(S_r) = O(n^{r-1})$$

□

Corollary 8.1)

Let $f \in Q_{(n,r)}$ for some fixed r. If $C^m(f) = \omega(n^{r-1})$ then:

$$C(f) = \Omega(C^m(f))$$

□

Corollary 8.2)

$$\forall f \in Q_{(n,2)} \quad C^m(f) = \Theta(C(f))$$

□

8.4) Extensions to Theorem(8.1)

Definition 8.2)

Let $r \in N$

$$Q^m_{(n,r)} = \{ \{ f_1, \ldots, f_m \} \mid \{ f_1, \ldots, f_m \} \subset Q_{(n,r)} \}$$

Lemma 8.2)

For almost all $F = \{f_1, \ldots, f_m\} \in Q^m_{(n,r)}$:

$$C(F) = \Omega\left(\frac{mn^r - mlogm}{log\ (mn^r - mlogm)}\right)$$

Proof

Let $aug(F)(X_n, Y)$ be the function:

$$\bigvee_{i=1}^{m} y_i \wedge f_i(X_n)$$

Clearly: $C(aug(F)) \leq C(F) + O(m)$ (*)

Let:

$$H = \{ aug(F) \mid F \in Q^m_{(n,r)} \}$$

Then:

$$|H| = \binom{|Q_{(n,r)}|}{m}$$

and Lemma(8.2) follows from Fact(8.2) and (*). □

We shall assume below that $m = \Theta(n^c)$ for some constant c, allowing the expression of Lemma(8.2) to be simplified to:

$$C(F) = \Omega\left(\frac{n^{r+c}}{(r+c)logn}\right)$$

Theorem 8.2)

Let $F \in Q^m_{(n,r)}$ for some constant r and $m = \Theta(n^c)$. Then if $C^m(F) = \omega(n^{r+c-1})$:

$$C(F) = \Omega(C^m(F))$$

Proof

Exactly as the construction of Theorem(8.1). □

Definition 8.3)

Let $t \in \mathbf{N}$

$$R_t = \bigcup_{\{f_1,\ldots,f_t\}} \{h(f_1,\ldots,f_t) \in M_n \mid f_i \in Q_{(n,r_i)}\}$$

$$R_{s,t} = \{\{h_1,\ldots,h_s\} \mid h_j \in R_t\}$$

For $h \in R_t$: $d(h) = \min_{m \in PI(h)} |var(m)|$

For $H \in R_{s,t}$: $d(H) = \min_{h_i \in H} d(h_i)$

As before we shall assume $s = \Theta(n^c)$ for some fixed c.

Theorem 8.3)

If $h \in R_t$ such that:

$$C^m(h) = \omega(n^{n-d(h)-1})$$

$$\text{Then:} \quad C(h) = \Omega(C^m(h))$$

Proof

Let $D(h) = \max_{m \in PI(h)} |var(m)|$ Then:

$$h = \bigvee_{i=d(h)}^{D(h)} h \wedge T_i^n \qquad (\dagger)$$

By definition of R_t, $d(h) = n-k$, for some $k \in \mathbf{N}$. Thus:

$$C^m(h) \le \sum_{i=d(h)}^{D(h)} C^m(h \wedge T_i^n)$$

$$\le (D(h)-d(h)+1) \max_i C^m(h \wedge T_i^n)$$

$$= O(\max_i C^m(h \wedge T_i^n))$$

But then: $C^m(h \wedge T_i^n) = \omega(n^{n-d(h)-1})$ and from Theorem(8.1):

$$C(h \wedge T_i^n) = \Omega(C^m(h \wedge T_i^n))$$

and (†) yields:

$$C(h) = \Omega(C^m(h))$$

as claimed. □

Corollary 8.3)

Let $H \in R_{s,t}$. If $C^m(H) = \omega(n^{n+c-d(H)-1})$ then:

$$C(H) = \Omega(C^m(H))$$

Proof

Combination of Theorem(8.2) and Theorem(8.3). □

Chapter 9

Conclusions

In the examination of monotone boolean function complexity above, attention has been focused on two methods central to most existing lower bound proofs in this field. The emphasis in the second part of this dissertation has been placed on developing the work of Berkowitz [3] concerning the relation between monotone and combinational network complexity via slice functions. Here we consider the wider relevance of the results proved and propose some directions for further research.

Chapters(3) and (4) establish improved linear lower bounds on the monotone network complexity of threshold functions. For the lower bound on T_3^n, the basic inductive approach was supplemented with a wire counting argument. The counting procedure does not adapt to combinational networks for several reasons: the replacement rule of Lemma(3.1) does not hold for such networks and so it is not possible to determine, as precisely, the structure of those optimal networks for which inductive gate elimination fails. Secondly the analysis in the wire counting argument relies heavily on monotonicity. A further, potential, weakness in the proof technique is that presently it does not extend to other monotone functions. However this is also a problem with the methods of Paul [38] and Blum [7] for unrestricted networks and in these cases the functions considered are slightly artificial. There does appear to be some scope for sharpening this argument in the case of higher fixed threshold functions.

In contrast the lower bound construction for T_k^n, in Chapter(4), could be adapted to combinational networks since the technique is entirely an inductive one. Stockmeyer's approach for Congruence and other symmetric functions can be interpreted as a simplified example of using reductions in this way [46].

The characterisation of all replacement rules applicable when computing monotone boolean functions, unifies the results on replacement rules for particular functions proved by several authors. A problem arises in that for proving the correctness of a specific rule, the representation in CNF and DNF is not the most tractable, and an interesting development of this work, would be to obtain similar characterisations in which the expressions resulting could be denoted in terms of partial assignments to f and g. For example, consider the functions:

$$Z^+(f) = \bigvee_{i=1}^{n} x_i \wedge f^{|x_i=0}$$

$$U^+(f) = \bigwedge_{i=1}^{n} (x_i \vee f^{|x_i=1})$$

It may be easily verified that if f is a threshold function, then $Z^+(f) = Z(f)$ and $U^+(f) = U(f)$.

Chapter(6) extended the results of Berkowitz in two ways: by showing that the existence of pseudo-complements is a property of all monotone boolean functions, although the construction of these does not, in general, permit an efficient translation from combinational to monotone networks; and by proving that slice function are a special case of the class of functions defined by dissecting transforms, for which similar translational results are provable,

The final two chapters answer some questions left open by Berkowitz and Wegener [56]. We have considered three "core" monotone boolean NP-complete problems: $(n/2)$-cliques, Hamiltonian circuit and Satisfiability. For each of these the central slice function is proved NP-complete and thus is a strong candidate for a "hard" slice function. The central slice function may possess the same projective properties for non NP-complete functions, however it seems unlikely that the more powerful result of Lemma(7.4), whereby the underlying function f is a projection of $Cen(f)$, holds in general. We conjecture the following:

Conjecture 9.1)

Let $p(n)$ be any polynomial in n of degree at most r, where r is fixed.

$(n/2)$–clique is not a monotone projection of $Cen((p(n)/2$–clique$)$

□

Conjecture 9.2)

Let $p(n)$ be as above. Let $ZCONV_n:\{0,1\}^{3n} \to \{0,1\}$ be the monotone function:

$$ZCONV_n(x_0,\ldots,x_{n-1},y_0,\ldots,y_{n-1},z_0,\ldots,z_{n-1}) = \bigvee_{i+j+k \equiv 0 (mod\ n)} x_i\ y_j\ z_k$$

$ZCONV_n$ is not a projection of $Cen(ZCONV_{p(n)})$.

□

The reasons for the first conjecture are based on the fact that the projection must arrange for *exactly half* the inputs of the central slice instance to be 1, no matter how many edges are present in the graph $G(X_n^U)$. While it is straightforward to achieve this using a non-monotone projection (by including the complement graph $\overline{G}$) there does not appear to be any similar method using monotone projections. We note that if Conjecture(9.1) is false then the question $P =?\ NP$ may be reformulated as:

$P \neq NP$ if the $(n/2)$-clique predicate has superpolynomial *monotone* network complexity.

The second conjecture is proposed since the structure of $ZCONV_n$ appears to prevent the techniques of Lemma(7.4) being applied.

However the property of Lemma(7.4) *does* hold for some computationally interesting multiple-output functions, using the following extended definition of "projection".

Definition 9.1)

Let $F = \{f_1,\ldots,f_m\} \in M_{n,m}$ and $G = \{g_1,\ldots,g_q\} \in M_{p,q}$ where $p \geq n$ and $q \geq m$

F is a projection of G if and only if:

$\exists\ \sigma: Y \to \{X_n, \neg x_1,\ldots,\neg x_n, 0, 1\}$
and a one-one mapping $\tau:\{1,\ldots,q\} \to \{1,\ldots,m\}$ such that:

$$\forall\ 1 \leq i \leq m \quad f_i(X_n) = g_{\tau(i)}(\sigma(Y))$$

Lemma 9.1)

Let $BMP_N:\{0,1\}^{2N^2} \to \{0,1\}^{N^2}$ denote the $N \times N \times N$ matrix product functions of Lemma(5.1).

BMP_N is a projection of $Cen(BMP_{2N})$

Proof (Outline)

Let A and B be any two $N \times N$ boolean matrices. Define A^* and B^* to be the $(2N) \times (2N)$ boolean matrices:

$$A^* = \begin{bmatrix} A & \neg A \\ 0 & 1 \end{bmatrix} \; ; \; B^* = \begin{bmatrix} B & \neg B \\ 0 & 1 \end{bmatrix}$$

Since for all input assignments to A and B, the total number of inputs set to 1 is $4N^2$ and since:

$$A^* B^* = \begin{bmatrix} AB & \neg(BA) \\ 0 & 1 \end{bmatrix}$$

the lemma follows. □

Let $A_1, A_2, \ldots, A_m$ be m $M \times N$ boolean matrices. In [55] Wegener introduced the monotone functions Direct Matrix Product $((mMN)-DMP)$ as a generalisation of boolean matrix multiplication.

$$(mMN)-DMP:\{0,1\}^{mMN} \to \{0,1\}^{M^m}$$

where each output $y_{h_1 h_2 \ldots h_m}$ $(1 \leq h_i \leq M)$ is defined as:

$$y_{h_1 h_2 \ldots h_m} = \bigvee_{1 \leq l \leq N} x^1_{h_1 l} \wedge x^2_{h_2 l} \wedge \ldots \wedge x^m_{h_m l}$$

where $x^i_{j,k}$ is the (j,k) entry of the matrix A_i. (Thus the output $y_{h_1 \ldots h_m}$ is true if and only if all the rows referenced have a common 1.)

Wegener defined instances of this set of functions having monotone network complexity $\Theta(n^2/\log n)$ where $n \geq M^m, mMN$

Lemma 9.2)

$$(mMN)-DMP \text{ is a projection of } Cen(((m+1)M(2N))-DMP)$$

Proof (Outline)

Let A_i^* be the $M \times (2N)$ matrix:

$$A_i^* = (A_i \ \neg A_i)$$

and let A^*_{m+1} be the $M \times (2N)$ matrix $(1 \ \ 0)$ It may be verified that:

$$(mMN)-DMP(A_1,\ldots,A_m) = ((m+1)M(2N)-DMP(A_1^*,\ldots,A^*_{m+1})$$

□

We observe that good upper bounds on $Cen(BMP_N)$ also provide good upper bounds on the combinational complexity of Boolean Matrix Product. Chapter(8) resolved some questions, on the existence of complexity gaps, for a certain class of functions. To conclude we present some open problems arising from the work above.

Open Problems

P1) The wire counting argument employed in Chapter(3), assumes that optimal networks of a particular type exist which compute T_3^n. However, it is not clear whether this assumption is valid. Thus:

Does any (optimal) monotone network exist which computes T_3^n AND is such that every network input enters exactly 2 $\vee$-gates?

A negative answer to this question would yield a $3n$ lower bound.

P2) To what extent can the lower bounds of Theorem(4.3) be improved by introducing wire counting arguments similar to that of Chapter(3)?

P3) The functions $Z(f)$ and $U(f)$ of Chapter(5) are inverse. One can thus partition M_n into a set of "cycles" of monotone functions, $\{C_1,\ldots,C_{\alpha_n}\}$, where each cycle, C_i, consists of a set of functions $\{f_1,\ldots,f_{\beta_i}\}$ with the property that:

$$Z(f_i) = Z(f_{((i+1) \bmod \beta_i)+1})$$

Little is known about the cycles C_i. Three basic questions are:

Q1) How do the values of α_n, the number of cycles present, relate to n?

Q2) What is the maximal value of β_i?

Q3) Given f and g in M_n when do f and g belong to the same cycle class?

P4) Prove or disprove Conjectures(9.1) and (9.2) above.

P5) Is $ZCONV_n$ a projection of:

$$ZCONV_n \wedge T^m_{n/2}(X_n) \wedge T^m_{n/2}(Y) \wedge T^m_{n/2}(Z) \ \vee \ T^m_{n/2+1}(X_n) \vee T^{m/2}_{n/2+1}(Y) \vee T^m_{n/2+1}(Z)$$

i.e The "natural" central dissecting transform of order 3.

P6) Consider the following $2n$-input n output function:

$$LCON(x_0,\ldots,x_{n-1},y_0,\ldots,y_{n-1}) = \{LC_0,\ldots,LC_{n-1}\}$$

where:

$$LC_k = \bigvee_{i+j = k \ (\mathrm{mod}\ n)} T^{m-1}_{n-1}(X_n - \{x_i\}) \wedge T^{m-1}_{n-1}(Y - \{y_j\})$$

It is easy to see that, by applying the results of Chapter(8):

$$C^m(LCON(X_n, Y)) = \Theta(C(LCON(X_n, Y)))$$

Now consider an optimal $\{\wedge, \vee, \neg\}$-network computing Circulant Convolution, in which negation is restricted to the inputs. If the replacements $x_i := T^{m-1}_{n-1}(X_n - \{x_i\})$ for each x_i, and the corresponding replacements for each y_j are made, then the resulting network computes $LCON(X_n, Y)$. Thus:

$$C^m(LCON(X_n, Y)) = O(C(CONV(X_n, Y)))$$

Where *CONV* is the n-output circulant convolution function.

The following open problem is therefore of interest:

Is $\mathbf{C}^{m}(LCON(\mathbf{X}_n,\mathbf{Y})) = \omega(n)$

A positive answer would establish a non-linear lower bound on the combinational complexity of multiplication.

References

[1] M.Ajtai, J.Komlos, E.Szemeredi: An $O(n\log n)$ Sorting Network; Proc. 15th STOC, (1983), pp 1-9

[2] C.Berge: Graphs and Hypergraphs; North-Holland, (1979)

[3] S.Berkowitz: Ph.D Dissertation reported in [51]

[4] W.M.Beynon: Replaceability and Computational Equivalence in Finite Distributive Lattices; Theory Of Computation Report No.61, Univ Of Warwick, (March 1984)

[5] P.Bloniarz: The Complexity of Monotone Boolean Functions and an Algorithm for Finding Shortest Paths in a Graph; Ph.D Dissertation; Technical Report No. 238, Lab. for Computer Science, MIT (January 1979)

[6] N.Blum: An $\Omega(n^{4/3})$ Lower Bound On The Monotone Network Complexity Of n-th degree Convolution; Preprint (March 1981)

[7] N.Blum: A Boolean Function Requiring $3n$ network size; Theoretical Computer Science; 28 (1984), pp 337-345

[8] R.Brent, H.T.Kung: The Area-Time Complexity Of Binary Multiplication; JACM, Vol. 28 No.3, (1981), pp 521-534

[9] A.K.Chandra, S.Fortune, R.Lipton: Lower Bounds For Constant Depth Circuits for Prefix Problems; Proc. 10th ICALP, Lecture Notes In Computer Science, Springer-Verlag, (1983), pp 109-117

[10] P.E.Dunne: Some Results On Replacement Rules In Monotone Boolean Networks; Theory Of Computation Report No.64, Univ. Of Warwick, (January 1984).

[11] S.Even: Graph Algorithms; Pitman (1979)

[12] R.Fagin, M.Klawe, N.Pippenger, L.Stockmeyer: Bounded-depth, Polynomial size Circuits for Symmetric functions; (To appear Theoretical Computer Science)

[13] M.Fischer, A.Meyer, M.S.Paterson: $\Omega(nlogn)$ Lower Bounds On the Length Of Boolean Formulas; SIAM Jnl. Of Computing, Vol. 11 No. 3, (August 1982), pp 416-426

[14] M.Fischer, N.Pippenger: Relations Among Complexity Measures; JACM, Vol.26 (1979), pp 361-381

[15] A.D.Friedman: Logical Design Of Digital Systems; Pitman (1975)

[16] M.Furst, J.B.Saxe, M.Sipser: Parity, Circuits and The Polynomial Time Hierarchy; Math. Syst. Theory; Vol. 17, (1984), pp 13-27

[17] L.Hodes, E.Specker: Lengths of Formulas and Elimination of Quantifiers I; in Contributions to Mathematical Logic, H.A.Schmidt, K.Schutte and H.-J.Thiele (editors); North-Holland, (1968) pp 175-188

[18] M.Jerrum, M.Snir: Some Exact Complexity Results For Straight-Line Computations Over Semirings; JACM Vol. 29 No.3, (July 1982), pp 874-897

[19] V.M.Khrapchenko: On The Complexity of the Realisation of the Linear Function in the Class of π-circuits; Mat. Zametki, Vol. 9 No. 1, (1971), pp 35-40, (In Russian, Translated in Math. Notes of Academy of Sciences of USSR, 9, pp 21-23)

[20] E.A.Lamagna: The Complexity Of Monotone Networks For Certain Bilinear Forms, Routing Problems, Sorting and Merging; IEEE Trans. Computers, C-28, (1979), pp 773-782

[21] A.Lingas: Lower Bounds for Straight-Line Algorithms; (from Ph.D Dissertation Univ. Of Warsaw 1979)

[22] R.Lipton, R.E.Tarjan: A Separator Theorem for Planar Graphs; SIAM Jnl. Applied Mathematics, Vol. 36, No. 2, (April 1979), pp 177-189

[23] R.Lipton, R.E.Tarjan: Applications Of A Planar Separator Theorem; SIAM Jnl of Computing, Vol.9, (1980), pp 615-627

[24] O.B.Lupanov: Ob Odnom Metode Sinteza Skhem; Izvestia VUZ (Radiofizika), 1 (1958), pp 120-140

[25] O.B.Lupanov: Implementing the Algebra of Logic Functions In Terms of Bounded Depth Formulas In The Basis Of $\wedge, \vee, \neg$; Soviet-Phys. Doklady; Vol.6 No.2, (August 1961), pp 107-108

[26] O.B.Lupanov: Complexity of Formula Realisation of Functions of Logical Algebra; Problemy Kibernetiki, 3, (1960), pp 61-80; Problems of Cybernetics, 3, (1962), pp 782-811

[27] W.F.McColl: Planar Circuits Have Short Specifications; Theory Of Computation Report No.59, Univ. of Warwick, (February 1984); (To appear: Proc. 2nd Annual Symposium on Theoretical Aspects of Computer Science; 1985)

[28] W.F.McColl: On The Planar Monotone Computation of Threshold Functions; (To appear: Proc. 2nd Annual Symposium on Theoretical Aspects of Computer Science; January 1985)

[29] W.F.McColl, M.S.Paterson: personal communication

[30] C.Mead, L.Conway: Introduction to VLSI Systems; Addison-Wesley (1980)

[31] K.Mehlhorn: Some Remarks On Boolean Sums; Acta Informatica, Vol.12, (1979), pp 371-375

[32] K.Mehlhorn, Z.Galil: Monotone Switching Networks and Boolean Matrix product, Computing, 16, (1976), pp 99-111

[33] E.I.Neciporuk: A Boolean Function; Soviet Math. Doklady, Vol.7 No.4, (1966), pp 999-1000

[34] M.S.Paterson, L.G.Valiant: Circuit Size is Nonlinear in Depth; Theoretical Computer Science, 2, (1976), pp 397-400

[35] M.S.Paterson: An Introduction to Boolean Function Complexity; Asterisque, (Journal of the French Mathematical Society), 38-39, pp 183-201

[36] M.S.Paterson: Complexity Of Monotone Networks for Boolean Matrix Product; Theoretical Computer Science, 1, (1975), pp 13-20

[37] M.S.Paterson: personal communication

[38] W.Paul: A $2.5n$-Lower Bound On The Complexity Of Boolean Functions; SIAM Jnl. of Computing, Vol.6, (1977), pp 427-443

[39] P.Pudlak: Bounds For Hodes-Specker Theorem; Logic and Machines: Decision Problems and Complexity (Proceedings); Lecture Notes In Computer Science, Vol. 171; (1983), pp 421-445

[40] J.Riordan, C.E.Shannon: The Number of Two-Terminal Series-Parallel Networks; Jnl. Math. and Phys. 21, (1942), pp 83-93

[41] J.E.Savage: Planar Circuit Complexity and the Performance of VLSI Algorithms; VLSI Systems and Computation. H.T.Kung, B.Sproull and G.Steele (Editors), Computer Science Press (1981), pp 61-68

[42] J.E.Savage: The Complexity Of Computing; John Wiley and Sons (1976)

[43] C.P.Schnorr: Zwei Lineare Untere Schranken fur die Komplexitat Boolescher Funktionen; Computing, 13, (1974), pp 155-171

[44] C.P.Schnorr: A Lower Bound on the Number of Additions In Monotone Computations of Monotone Rational Polynomials; Theoretical Computer Science, Vol.2 No.3, (1976), pp 305-317

[45] C.E.Shannon: The Synthesis of Two-Terminal Switching Circuits; Bell System Tech. Jnl, 28, (1949), pp 59-98

[46] L.Stockmeyer: On the Combinational Complexity of Certain Symmetric Boolean Functions; Math. Syst. Theory, Vol. 10, (1977), pp 323-336

[47] C.D.Thompson: Area-Time Complexity for VLSI; Proc. 11-th STOC, (1979), pp 81-88

[48] J.Tiekenheinrich: A 4n Lower Bound On the Monotone Boolean Network Complexity of a One Output Boolean Function; Inf. Proc. Letters, Vol. 18 No.4, (May 1984), pp 201-202

[49] L.G.Valiant: The Complexity Of Computing The Permanent; Theoretical Computer Science, 8, (1979), pp 189-201

[50] L.G.Valiant: Completeness Classes In Algebra; Proc. 11th STOC, (1979), pp 249-261

[51] L.G.Valiant: Exponential Lower Bounds For Restricted Monotone Circuits, Proc. 15th STOC, (1983)

[52] L.G.Valiant: personal communication via M.S.Paterson

[53] I.Wegener: A Counterexample to a Conjecture of Schnorr Referring to Monotone Networks; Theoretical Computer Science, 9, (1979), pp 147-150

[54] I.Wegener: A New Lower Bound on the Monotone Network Complexity of Boolean Sums; Acta Informatica, 13, (1980), pp. 109-114

[55] I.Wegener: Boolean Functions Whose Monotone Complexity is of Size $n^2/\log n$; Theoretical Computer Science, 21, (1982), pp 213-224

[56] I.Wegener: On The Complexity Of Slice Functions; Tech. Report, Universitat Frankfurt, July 1983 (To appear Theoretical Computer Science)

[57] J.Weiss: An $\Omega(n^{3/2})$ Lower Bound On The Complexity of Boolean Convolution; Information and Control; Vol. 59, (1983), pp 184-88

[58] A.C-C.Yao: Lower Bounds by Probabilistic Arguments; Proc 24th IEEE Symp. on FOCS, (1983)