Why is Boolean Complexity Theory Difficult?*

L.G. Valiant
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

1 Introduction

In the last decade substantial progress has been made in our understanding of restricted classes of Boolean circuits, in particular those restricted to have constant depth (Furst, Sipser, Saxe [FSS81], Ajtai [Ajt83], Yao [Yao85], Hastad [Has86], Razborov [Raz87], Smolensky [Smo87]) or to be monotone (Razborov [Raz85], Andreev [And85], Alon and Boppana [AB87], Tardos [Tar88], Karchmer and Wigderson [KW88]). The question arises, perhaps more urgently than before, as to what approaches could be pursued that might contribute to progress on the unrestricted model.

In this note we first argue that if P $\neq$ NP then any circuit-theoretic proof of this would have to be preceded by analogous results for the more constrained arithmetic model. This is because, as we shall observe, there are proven implications showing that if, for example, the Hamiltonian cycle problem (HC) requires exponential circuit size, then so does the analogous problem on arithmetic circuits. Since the set of valid algebraic identities in the latter model form a proper subset of those in the former, a lower bound proof for it should be strictly easier.

In spite of the above relationship the algebraic model is often regarded as an alternative, rather than a restriction of the Boolean model. One reason for this is that specific computations are usually understandable in one of these models, and not in both. In particular, the main power of the algebraic model derives from the possibility of cancellations, and it is usually difficult to express explicitly how these help in computing combinatorial problems. Our second aim in this note is to give an example of an algorithm, namely the Samuelson-Berkowitz method for computing

*Research supported by the National Science Foundation NSF-CCR-89-02500, the Office for Naval Research ONR-N0014-85-K-0445, the Center for Intelligent Control ARO DAAL 03-86-K-0171 and by DARPA AFOSR 89-0506.

the determinant, where the intermediate terms that are computed but ultimately cancelled by the arithmetic circuit can be exhibited explicitly in combinatorial terms. The ease of computing the determinant can be attributed to the existence of such an auxiliary set of monomials with certain computational properties. A proof of an exponential lower bound on the complexity of a polynomial that is believed to be hard, such as the permanent or the Hamiltonian circuit polynomial, would involve establishing the nonexistence of such an auxiliary set. It is difficult to imagine how such a proof might go.

Finally, we observe that in low-level complexity, the arguments giving precedence to studying the algebraic model no longer hold. A major open problem area is that of proving for some explicit problem that it cannot be computed by unrestricted Boolean circuits simultaneously in size $O(n)$ and depth $O(\log n)$. We describe, via some conjectures, one candidate approach towards proving such a lower bound for problems such as sorting. Analogous conjectures exist for the algebraic model, but resolution of those would not imply the same for the Boolean case.

2 Algebraic Structures

Let $x_1, \cdots, x_n$ be a set of indeterminates and S a set of constants. Let $\otimes$ and $\oplus$ be two binary operators. We define a *circuit* or a *straight line program* syntactically as a finite sequence of instructions of the form.

$$f_i := g_i \; op_i \; h_i \qquad i \in \{1, \cdots, C\}$$

where, for each $i, op_i \in \{\otimes, \oplus\}$, and $g_i, h_i \in \{x_1, \cdots, x_n\} \cup S \cup \{f_1, \cdots, f_{i-1}\}$. In other words a circuit is a sequence of C binary instructions, where each argument of each one is either an indeterminate, a constant, or the result of the execution of an instruction earlier in the sequence. The complexity of a circuit is C, the number of instructions.

Such a circuit can be interpreted in several ways. In this paper we shall assume that S is a commutative ring with identity. Then each instruction f_i can be identified with the polynomial that is computed at f_i, if $\otimes$ and $\oplus$ are interpreted as the ring operations in the polynomial ring $S[x_1, \cdots, x_n]$.

Among natural multivariate polynomials whose complexity in this model is of interest are Hamiltonian circuits (HC), the permanent (PERM) and the determinant (DET). These are defined over a matrix X of indeterminates $\{x_{11}, \cdots, x_{nn}\}$ where x_{ij} can be thought of as representing edge (i, j) of the complete directed graph G_n on nodes $\{1, \cdots, n\}$. They are defined as follows:

$$\begin{aligned}
PERM(X) &= \sum_{\sigma \varepsilon \Sigma} \prod_{i=1}^{n} x_{i,\sigma(i)}, \\
HC(X) &= \sum_{\substack{\sigma \varepsilon \Sigma \\ \sigma \text{ has 1 cycle}}} \prod_{i=1}^{n} x_{i,\sigma(i)}, \\
DET(X) &= \sum_{\sigma \varepsilon \Sigma} (-1)^{\#(\sigma)} \prod_{i=1}^{n} x_{i,\sigma(i)},
\end{aligned}$$

where Σ is the set of permutations $\sigma : \{1, \cdots, n\} \rightarrow \{1, \cdots, n\}$ and $\#(\sigma)$ is the number of even length cycles in σ.

Clearly the monomials of PERM and DET correspond to cycle covers in G_n, while these of HC correspond to Hamiltonian circuits. These polynomials can be interpreted in any ring S. An important case is S = GF[2], the finite field having two elements, in which case PERM = DET since then $-1 = 1$.

Each such problem is actually a polynomial family indexed by the number of indeterminates. When we talk about the permanent we mean the family PERM$_1$ PERM$_4$ PERM$_9, \cdots$ where the subscript denotes the number of variables. The computational complexity of such a family is determined by the family of circuits, one for each polynomial, that are the smallest circuits for each member. Thus PERM is polynomial time computable if and only if there is a family of circuits, of size growing polynomially in n, that computes PERM.

Different choices of S allow for different circuits.. For example PERM is polynomial time computable for S=GF[2] but is not known to be so for any ring S whose characteristic is not a power of 2 (see [Val79]).

We shall define three complexity classes of families of polynomials in which all coefficients are integral multiples of unity: ARP is the class computable by polynomial size circuits in all rings, GFP is that computable by polynomial size circuits for S = GF[2], and GFB is that computable by polynomial size circuits for S=GF[2], where the algebra is assumed to obey the extra axiom $x^2 = x$. By definition, clearly,

$$ARP \subseteq GFP \subseteq GFB.$$

Now GFB is equivalent to the class of Boolean functions of polynomial circuit size, since the polynomial ring over GF[2] with $x^2 = x$ gives Boolean algebra over $\{0, 1\}$ and the operations $\otimes$ and $\oplus$ in GF[2] give a complete Boolean basis. Our observations here are first that some natural polynomials in this algebra have natural combinatorial interpretations, and, second, that their complexity can be related to NP-completeness.

In particular, a GFB circuit for HC computes nothing other than the parity of the number of Hamiltonian circuits in a graph. In other words if the indeterminates x_{ij} of such a circuit are set to 1 or 0 according to whether edge (i, j) is present in the graph then the circuit will compute 1 or 0 according to whether the number of Hamiltonian circuits is odd or even. Furthermore, the following is implicit in [VV86] since the randomized reductions used there can be simulated by small circuits [Adl78]:

Theorem 1 *$HC \in GFB \Rightarrow NP$ has polynomial size circuits.*

Denoting the class of Boolean functions of polynomial Boolean circuit size by pC [Val79], we conclude, therefore, that since NP $\not\subseteq$ pC $\Rightarrow$ HC $\notin$ GFB, any efforts at proving the former should be retargeted at first proving the latter or indeed any of its even more restricted versions:

$$NP \not\subseteq pC \Rightarrow HC \notin GFB \Rightarrow HC \notin GFP \Rightarrow HC \notin ARP$$

We note that $NP \not\subseteq pC$ is equivalent to the $P \neq NP$ question but with Turing uniformity removed from the definitions. It is argued in [Val79] that the nonuniform versions of such complexity questions as $P \neq NP$ are at least as natural as the uniform versions. For example, the nonuniform version of NP, there denoted by pD (for polynomial definable,) has a completeness class that includes the original Hamiltonian circuits problem, just as NP has. Our conclusion, therefore, is that unless the explanation of the possible intractability of NP is to do with uniformity, one should seek it first in the more restricted structure of ARP or GFP.

3 Cancellations in the Samuelson-Berkowitz Algorithm

Any Boolean circuit can be reexpressed efficiently as a Boolean circuit over $\otimes$ and $\oplus$ in GF[2]. In almost any efficient such circuit the computation and cancellation of unwanted terms by the $\oplus$ operations plays a central role. Unfortunately, the way in which efficient computations exploit this facility is little understood. It is difficult even to find interesting examples where the structure of the cancelled terms can be exhibited explicitly.

In this section we study the one example we have found in a nontrivial computation where the terms that are computed but ultimately cancelled have a natural characterization. The algorithm is due to Samuelson [Sam42] and adapted by Berkowitz [Ber84] (see also Eberly [Ebe85]), and computes the determinant in ARP. (Note that other standard algorithms, such as Gaussian elimination, either use division or work only in certain rings.)

The algorithm for computing $\det(X)$ is as follows. We let B_k $(0 \leq k \leq n-1)$ be the principal $(n-k) \times (n-k)$ minor of X. Then we define an $(n-k) \times 1$ matrix C_k and a $1 \times (n-k)$ matrix D_k for each k $(1 \leq k \leq n-1)$ as follows

$$B_{k-1} = \begin{pmatrix} B_k & C_k \\ D_k & X_{n-k+1,n-k+1} \end{pmatrix}$$

where X_{ii} denotes the (i,i) element of matrix X.

For each k $(1 \leq k \leq n)$ we define T_k to be the $(n+2-k) \times (n+1-k)$ matrix defined as follows:

$$(T_k)_{ij} = \begin{cases} 0 & if \quad i > j+1 \\ -1 & if \quad i = j+1 \\ X_{n-k+1,n-k+1} & if \quad i = j \\ D_k B_k^{j-i-1} C_k & if \quad i < j \end{cases}$$

It turns out that the coefficients of the characteristic polynomial are given by the $(n+1) \times 1$ matrix

$$\prod_{k=1}^{n} T_k$$

where the (1,1) term gives the determinant.

The combinatorial interpretation of this computation of the determinant can be derived by noting that it is made up of the sum of terms of the form:

$$(T_1)_{1,i_1}(T_2)_{i_1,i_2}(T_3)i_2,i_3 \cdots (T_{n-1})_{i_{n-2},i_{n-1}}(T_n)_{i_{n-1},1} \qquad (*)$$

Now the sequence of subscripts $1, i_1, i_2, \cdots i_{n-1}, 1$ imposes constraints on the products of matrix elements that occur in the term having this sequence of subscripts. From the definition of T_k it can be seen that $(T_k)_{ii} = X_{n-k+1,n-k+1}$ and $(T_k)_{ij} = D_k B_k^{j-i-1} C_k$ if $i < j$. Interpreted as paths in the underlying complete graph G_n, $(T_k)_{i,i}$ is a self-loop at node $n-k+1$, while $(T_k)_{ij}$ consists of closed walks of length $j-i+1$ going through only nodes $1, 2, \cdots, n-k+1$, and going through the last node $n-k+1$ exactly once. In both cases we can interpret $(T_k)_{ij}$ as a set of closed walks of length $j-i+1$ since a self loop is of length one.

If s_k is the size of a closed walk generated by T_k, s_k being zero if $i_{k-1} = i_k + 1$, then clearly

$$s_k = i_k - i_{k-1} + 1.$$

Since $1 = i_\circ = i_n = 1$ in $(*)$, it follows that

$$\sum_{1}^{n}(i_k - i_{k-1}) = \sum_{1}^{n}(s_k - 1) = 0.$$

Hence $\sum_1^n s_k = n$. Furthermore the sign of the term in $(*)$ is given by (-1) to the power of $p = \sum(s_k - 1)$ with summation over all k such that $s_k \geq 2$. Hence each even walk contributes a factor $(-1)^{s_k - 1} = -1$ and each odd walk a factor of $+1$. Therefore, the product $(*)$ contributes a factor

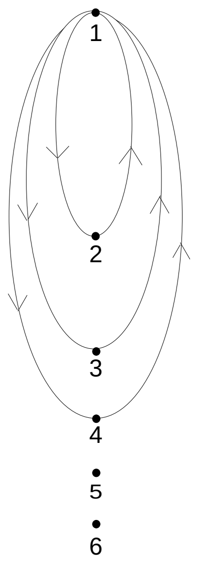
$$(-1)^{\#\text{ even length walks}}$$

which is exactly the same factor as in the determinant, if the only sets of walks counted are the sets of disjoint cycles.

To summarize, we define an m-loop in a graph with nodes $\{1, \cdots, n\}$ to be a closed walk going through node m exactly once, and through every $r > m$ zero times. A *loop-cover* of G is a set of m-loops, at most one for each m, such that the sum of the lengths of the loops is n. Note that a loop may repeat nodes and edges and that its length is the number of edges occurring in it allowing for multiplicity.

We claim that what the algorithm computes is the set of terms that correspond to loop covers of G_n, each term having sign $+$ or $-$ according to whether the loop cover has even or odd number of loops of even length.

Now a multi-set of n edges $\tilde{E}$ corresponds to more than one loop cover. For example the set of six edges shown below



has six distinct loop covers. As can be verified, two of these correspond to single loops (of size 6), three correspond to two loops (of sizes 1 and 2 respectively in all cases) and one to three loops. Since all the loops involved have even length the algorithm will assign positive sign to the three covers with an even number of them, and negative sign to the three with an odd number of them. In other words these loop covers will cancel out.

Now the terms in the determinant correspond to loop covers that consist only of cycles that are disjoint. Clearly such sets of edges have only one loop cover, and the algorithm will give it the correct sign.

From the correctness of the algorithm and our combinatorial interpretation of the terms computed by it we conclude that

Theorem 2 *If $\tilde{E}$ is any multiset of n edges in G_n then*

1. *$\tilde{E}$ is a cycle cover $\Leftrightarrow$ $\tilde{E}$ forms exactly one loop cover,*
2. *$\tilde{E}$ is not a cycle cover $\Leftrightarrow$ $\tilde{E}$ forms an even number of loop covers, exactly a half of which have an odd number of even length loops.*

We note in conclusion that the Samuelson-Berkowitz algorithm is not multilinear, in the sense that it computes powers of the variables higher than one. It is an open problem as to whether a multilinear ARP algorithm exists for the determinant.

4 Simultaneous Lower Bounds on Size and Depth

We have argued that for NP-complete problems such as Hamiltonian circuits, super-polynomial lower bounds on Boolean circuits imply similar lower bounds on more restricted algebraic models of computation. Hence one would expect that resolving the former problem should be mathematically more tractable then resolving the latter. In low-level complexity, however, this formal relationship appears to break down, and there is fuller justification for working separately on both Boolean and algebraic problems. A case in question is that of simultaneous lower bounds on size and depth. For Boolean circuits it a major open problem to find an explicit family of sets of Boolean functions such that there is no family of Boolean circuits over a complete basis computing them that has size $O(n)$ and depth $O(\log n)$ simultaneously. Here n is the number of arguments plus functions. Below we shall give a combinatorial conjecture the truth of which would imply such lower bounds for problems including shifting and sorting. Analogous conjectures have been proposed in the algebraic context [Val77] and some progress made on them ([Fri90]).

Consider a bipartite graph G with node set $X \cup Y$ where $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_n\}$ denote input variables and output functions respectively. Suppose the edges are defined implicitly by a mapping τ where $\tau(y_i) \subseteq X$ is the set of input nodes that are adjacent to y_i in the graph.

Now we define m Boolean functions $f_1(\underline{x}), \cdots, f_m(\underline{x})$ and n further Boolean functions $g_1, \cdots, g_n$, where g_i has $m + |\tau(y_i)|$ Boolean arguments, such that, with some abuse of notation, $y_i = g_i(f_1(\underline{x}), \cdots, f_m(\underline{x}), \tau(y_i))$. In other words we have m functions of the inputs, and each output y_i can be an arbitrary function of these m *common bits* and of the inputs $\tau(y_i)$ to which it is connected directly in G.

We say that G *realizes* permutation σ with m *common bits* if there exist $f_1, \cdots, f_m, g_1, \cdots, g_n$ such that for all i, $1 \leq i \leq n$

$$x_{\sigma(i)} = g_i(f_1(\underline{x}), \cdots, f_m(\underline{x}), \tau(y_i)).$$

In other words, for the fixed G and given σ one can find the appropriate Boolean functions $\{g_i\}, \{f_j\}$ such that the $\{y_i\}$ realize the permutation σ of $\{x_i\}$, for all truth assignments to the $\{x_i\}$.

Now it appears that if G is sparse, say with degree less than $n^{1-\varepsilon}$ for some $\varepsilon > 0$, and if m is small enough (e.g. $m < n/2$) then the common bits form an information bottleneck, and G cannot realize all permutations. In particular:

Conjecture 1 *If G has degree 3 then there is a permutation σ such that G does not realize σ with $n/2$ common bits.*

Conjecture 2 *If G has degree 3 then there is a cyclic shift σ such that G does not realize σ with $n/2$ common bits.*

Conjecture 1′ and 2′ *Conjectures 1 and 2 but for G with $O(n^{1+\varepsilon})$ edges for some $\varepsilon > 0$.*

Now Conjecture 2′ would imply that the following shifting problem has no circuit of size $O(n)$ and depth $O(\log n)$. The shifting problem is defined with $n + \lceil \log_2 n \rceil$ inputs $x_0, \cdots, x_{n-1}, s_1, \cdots, s_{\lceil \log_2 n \rceil}$ where the value of $\underline{s}$ defines in binary the amount s to be shifted. In other words output y_i equals $x_{i-s \bmod n}$ where s is the shift.

To prove the implication we appeal to Proposition 6.2 in [Val77] which implies that for any $\varepsilon > 0$, in any n-input n-output graph family of size $O(n)$ and depth $O(\log n)$ some $n/2$ nodes and adjacent edges can be removed so that fewer than $O(n^{1+\varepsilon})$ input output pairs remain connected. Hence if such a computation graph existed for shifting (i.e. if we identified $x_1, \cdots, x_n$ of the shifter as the inputs of the graph) then we could identify the $n/2$ removed nodes as the common bits. Each setting of the shift bits $\{s_i\}$ could be seen as setting the functions $\{f_i\}$ and $\{g_i\}$ so as to contradict Conjecture 2′.

Conjecture 1′ is a weaker statement than Conjecture 2′, and might be easier to prove. Potentially it could yield a lower bound on sorting, say of $n/(2 \log n)$ numbers each of size $2 \log n$. The conjecture as stated is not quite enough but the reader can verify that it can be adapted in various ways so as to imply such a lower bound.

As our conjectures imply, little is known about the permutations that can be realized even for low degrees such as three. The degree one case can be analyzed completely. The canonical case is the identity graph $G(\tau(y_i) = x_i)$ which, with $n/2$ common bits, can realize such permutations as $y_{2i} = x_{2i-1}, y_{2i-1} = x_{2i}$ by letting $f_i = x_{2i-1}$, $y_{2i} = f_i \oplus x_{2i}, y_{2i-1} = f_i \oplus x_{2i-1}$.

The degree two case has been analysed partially by K. Kalorkoti.

References

[AB87] N. Alon and R.B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, (1987).

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proc. 19th IEEE Symp. on Foundations of Computer Science*, pages 75–83, (1978).

[Ajt83] M. Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, (1983).

[And85] A.E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone funtions. *Sov. Math. Dokl*, 31(3):530–534, (1985).

[Ber84] S.J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Proc. Letters*, 18:147–150, (1984).

[Ebe85] W. Eberly. Very fast parallel matrix and polynomial arithmetic. Technical Report 178/85, Comp. Sci. Dept., Univ. of Toronto, (1985).

[Fri90] J. Friedman. A note on matrix rigidity. Manuscript, (1990).

[FSS81] M. Furst, J.B. Saxe, and M. Sipser. Parity circuits and the polynomial time hierarchy. In *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, pages 260–270, (1981). Also Math. Systems Theory 17, (1984) 13-27.

[Has86] J. Hastad. Improved lower bounds for small depth circuits. In *Proc. 18th ACM Symp. on Theory of Computing*, pages 6–20, (1986). Also Computational Limitations for Small Depth Circuits, (1988), MIT Press, Cambridge, MA, USA.

[KW88] M. Karchmer and A Wigderson. Monotone circuits for connectivity require super-logorithmic depth. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 539–550, (1988).

[Raz85] A.A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Sov. Math. Dokl.*, 31:354–357, (1985).

[Raz87] A.A Razborov. Lower bounds for the size of circuits for bounded depth with basis $\{\wedge, \oplus\}$. *Math Notes of the Acad. of Sciences of the USSR*, 41(4):333–338, (1987).

[Sam42] P.A. Samuelson. A method of determining explicitly the coefficients of the characteristic equation. *Ann. Math. Statist.*, 13:424–429, (1942).

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *19th ACM Symp. on Theory of Computing*, pages 77–82, (1987).

[Tar88] E. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, (1988).

[Val77] L.G. Valiant. Graph-theoretic arguments in low-level complexity. *Lecture Notes in Compter Science*, 53:162–176, Springer (1977).

[Val79] L.G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symp. on Theory of Computing*, pages 249–261, (1979).

[VV86] L.G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, (1986).

[Yao85] A.C.-C. Yao. Separating the polynomial time hierarchy by oracles. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 1–10, (1985).