Accidental Algorithms*

Leslie G. Valiant
Division of Engineering and Applied Sciences
Harvard University
valiant@deas.harvard.edu

Abstract

We provide evidence for the proposition that the computational complexity of individual problems, and of whole complexity classes, hinge on the existence of certain solvable polynomial systems that are unlikely to be encountered other than by systematic explorations for them. We consider a minimalist version of Cook's 3CNF problem, namely that of monotone planar 3CNF formulae where each variable occurs twice. We show that counting the number of solutions of these modulo 2 is $\oplus$P-complete (hence NP-hard) but counting them modulo 7 is polynomial time computable (sic). We also show a similar dichtomy for a vertex cover problem. To derive completeness results we use a new holographic technique for proving completeness results in $\oplus$P for problems that are in P. For example, we can show in this way that $\oplus$2CNF, the parity problem for 2CNF, is $\oplus$P-complete. To derive efficient algorithms we use computer algebra systems to find appropriate holographic gates. In order to explore the limits of holographic techniques we define the notion of an elementary matchgrid algorithm to capture a natural but restricted use of them. We show that for the NP-complete general 3CNF problem no such elementary matchgrid algorithm can exist. We observe, however, that it remains open for many natural #P-complete problems whether such elementary matchgrid algorithms exist, and for the general CNF problem whether non-elementary matchgrid algorithms exist.

1 Introduction

Many mathematical questions can be usefully framed in the following enumerative form. There is an enumeration E of objects, a property X that, on the surface, members of E are unlikely to have, and the conjecture is made that no object in E has property X. The Goldbach conjecture, for example, is of this form where E is the list of even numbers and X is the property of not being the sum of two primes. We believe that the question of the computational complexity of specific problems, and of the complexity classes in which they are known to be complete, can be usefully formulated in this way. The existence of objects having the unlikely property would then imply the existence of unexpectedly efficient, in particular, polynomial time algorithms.

In complexiy theory the holographic framework [25] offers such enumerations. The objects enumerated are sets of polynomial systems such that the solvability of any one member would give a polynomial time algorithm for a specific problem. The enumeration is not quite as simple as for numbers because, while basis size can be enumerated, another dimension is the choice of formulation of the combinatorial problem at hand. We argue that, aside from this issue, the situation with the P = NP question is not dissimilar to that of other unresolved enumerative conjectures in mathematics. The possibility that accidental or freak objects in the enumeration exist cannot be discounted if the objects in the enumeration have not been studied systematically.

The technical contributions of the paper are fourfold.

We first introduce a general technique for proving completeness results and apply it to show $\oplus$P-completeness for several natural problems. The $\oplus$P-completeness of a problem follows immediately if it is NP-complete and reductions exist from satisfiability that preserve numbers of solutions. However, for problems for which existence of solutions is known to be in P, proofs of $\oplus$P-completeness are more problematic. The only previous result we know is the observation in [26] that $\oplus$MonSat is $\oplus$P-complete. The new general technique is that of holographic reduction. A holographic reduction is one based on gadgets that do not preserve individual solutions in a one-to-one or many-to-one sense, but preserve some other measure such as a sum, here a sum modulo 2. Using this method we prove the $\oplus$P-completeness of several problems. One example is the parity of 2CNF for which there was no previous evidence of hardness. These reductions are efficient and simple. Besides establishing $\oplus$P-completeness they can also serve the further purpose of providing evidence for the difficulty of

*This work was supported by grants NSF-CCR-03-10882, and NSF-CCF-04-27129.

exact counting where such evidence previously did not exist or existed only through complex constructions. For example, we show that graph vertex cover parity (or equivalently monotone 2CNF parity) is hard even for planar bipartite graphs where every node in one side of the partition has degree 2 and every node in the other has degree 3. A #P-completeness result for this case was not available before (though the recent techniques of [28] may be applicable). Even for the general planar case of vertex cover (unrestricted degree, nonbipartite) only complex constructions were known before [21].

Second we develop some general holographic results for matchgates over bases of size two. Since in nontrivial computations some three-argument primitives are essential, bases of size two will require matchgates with six external nodes and a system of polynomial equations with many variables. To help in their analysis we exhibit some useful two argument functions that can be provably supported by very broad sets of bases.

Third, we focus on a lean case of the 3CNF problem that illuminates the boundary - at least as currently widely perceived - of polynomial time computation. This case is that of planar formulae in which each variable occurs at most twice. This problem without restrictions to planarity or to 3 literals per clause, has been investigated by Bubley and Dyer [2], who showed that its counting problem is #P-complete, though polynomial time approximable. Here we show that families of weighted variants are polynomial time countable. Using this we show in particular that $\#_7$Pl-Rtw-Mon-3CNF, where all variable occurrences are positive and the solutions are to be counted modulo 7, is polynomial time computable. This contrasts sharply with the completeness result, which we prove using the previously described parity preserving reductions, that $\oplus$Pl-Rtw-Mon-3CNF (equivalently $\#_2$Pl-Rtw-Mon-3CNF) is $\oplus$P-complete (and hence NP-hard). We note that intuition might have suggested that counting modulo 2 is easier than counting modulo 7.

Fourth, we develop a notion of an *elementary* encoding into matchgrids to capture a natural way of seeking polynomial time holographic algorithms. We show that 3CNF does not have such elementary encodings, and that the {0,1}-Permanent is analogously restricted. We go on to observe that this leaves open nevetheless the possibilities that these problems have non-elementary encodings into matchgrids, and that other #P complete problems such as #2CNF, have elementary encodings.

Throughout we shall define problems using the following notation. Satisfiability of general Boolean formulae is denoted by Sat. Conjuctive normal form is denoted by CNF, and if there are exactly 2 or 3 literals in each clause then by 2CNF and 3CNF. The variants in which the clauses have exactly three literals and it is required that exactly one, or exactly two literals be satisfied in each clause, is denoted by 1-in-3CNF and 2-in-3CNF, respectively. Monotone formulae will be denoted by Mon, and planar formulae [17] by Pl. We note that counting monotone 2CNF is equivalent to counting vertex covers. Hence for the abbreviation Mon-2CNF we shall sometimes use VC. Bipartite graphs will be denoted by Bip, and those in which all the nodes have degree 2 on one side and degree 3 on the other by 3/2Bip. Formulae in which each variable occurs twice (read-twice) will be denoted by Rtw. If both occurrences of every variable are positive we denote that by RtwMon. If each variable occurs once positively and once negated, we denote that by RtwOpp. Counting modulo k the number of solutions of an NP problem will be denoted by $\#_k$P (as in [22]) except that for the case $k = 2$ we also use the alternative notation $\oplus$P (from[20]). (The classes $\#_k$P are sometimes called Mod_kP.) We shall use the prefices $\#_k$ and $\oplus$ for particular problems also, to refer to the problems of counting their solutions modulo k or modulo 2.

Thus the two problems contrasted in the Abstract are $\#_2$Pl-Rtw-Mon-3CNF and $\#_7$Pl-Rtw-Mon-3CNF, where the former is also known as $\oplus$Pl-Rtw-Mon-3CNF. A second pair is $\#_2$Pl-3/2Bip-VC, which is equivalent to $\#_2$Pl-Rtw-Mon-3CNF, and is therefore $\oplus$-complete, and $\#_7$Pl-3/2Bip-VC, which we also show is polynomial time computable.

We note that read-twice formulae have been investigated before in learning [1, 12], in the context of satisfiability [14, 10, 24, 7], and, as already mentioned, also for counting [2] and parity [24, 26].

2 Parity-preserving holographic reductions

A simple example of a reduction that preserves parity but not the number of solutions is the following.

Theorem 2.1 *$\oplus$Pl-Mon-CNF is $\oplus$P-complete.*

Proof We reduce $\oplus$Pl-3CNF to $\oplus$Pl-Mon-CNF. The counting problem for the former, #Pl-3CNF, was shown to be #P-complete by parsimonious reduction from #3CNF [17, 13], and from this it follows that $\oplus$Pl-3CNF is $\oplus$P-complete. It remains only to remove negated literals. Consider any planar CNF formula F and for a variable x replace an occurrence of its negation by a new variable x^*. Also introduce another variable $x"$ and conjoin the formula with the clauses $(x + x^*)(x + x")(x^* + x")$. Then $(x + x^*)$ enforces that there are no solutions with both x and x^* false. Also, $(x + x")(x^* + x")$ enforces that when x and x^* are both true then there will be an even number of solutions (i.e. with $x" = 1$ and $x" = 0$, respectively). Clearly, if x, x^* have opposite values, reflecting the solutions of the original equation F, then there will be just one solution for the new variables, namely $x" = 1$. In order to preserve planarity, for each variable x we shall introduce a separate variable

x^* for each occurrence of x' in F, and hence also a separate variable x'' for each such x^*. ■

We shall now describe four further parity preserving constructions:

(A) To simulate $(x + y)$ by a bipartite 2CNF formula where x and y are to be represented in the same part of the partition we replace it by $(x' + u')(u' + y')$ where x' denotes the negation of x and u is a new variable. It is easy to verify that when x, y have the sets of values 00, 01, 10, 11 then $(x' + u')(u' + y')$ has 2, 1, 1, and 1 solution(s), respectively. Note that this reduction to bipartite graphs therefore preserves the parity of the number of solutions. Monotone formulae remain monotone, but in the negations of the original variables.

(B) To simulate $(x = y)$ we replace it by the chain $(x + s)(s + t)(t + y)$ where s and t are new variables. It is easy to verify that when x, y have the sets of values 00,01,10,11 this chain has 1,2,2,3 solution(s), respectively.

(C) To simulate $(x = y)$ in a read-thrice formula by a bipartite 2CNF formula in which nodes have degree 2 in one partition and degree 3 in the other, we replace it by the following formula which is bipartite and in which internal nodes have degrees 2 and 3 respectively in the two parts: $(x + x_1)(x_1 + x_2)(x_2 + s)(x_2 + t)(s + y_2)(t + y_2)(y_2 + y_1)(y_1 + y)$. It is easy to verify that when x, y have the sequences of values 00,01,10,11 this chain has 7,12,12,21 solutions, respectively.

(D) To simulate a parity gate for the variables x, y, z, we use a formula $(x + x_1)(y + y_1)(z + z_1)(x_1 + y_1)(y_1 + z_1)(z_1 + x_1)$, with x_1, y_1, z_1 as new variables. It is easy to verify that this has 1, 2, 3, or 4 solutions according to whether the number of the external variables $\{x, y, z\}$ that are true is 0, 1, 2, or 3 respectively. In other words this computes the even parity gate.

Theorem 2.2 *$\oplus$Pl-3/2Bip-Mon-2CNF and $\oplus$Pl-3/2Bip-VC are $\oplus$P-complete.*

Proof In [13] it is shown that there is a parsimonious reduction from 3CNF to planar Pl-Mon-1-in-3CNF. Clearly, by negating all the variables we can instead look at the same problem, but where we insist that exactly two variables in each clause are true instead of one, namely Pl-Mon-2-in-3CNF.

Simulate each such gate $(x + y + z)$ by a formula $(x + y)(y + z)(x + z)(x + t)(y + t)(z + t)$. This has 0, 0, 1, or 2 solutions according to whether the number of the external variables $\{x, y, z\}$ that are true is 0, 1, 2, or 3 respectively.

Now the reduction can be made to a bipartite graph by replacing every clause $(x + y)$ in the construction so far by the bipartite $(x' + u')(u' + y')$ where u is a new variable, as in (A) above. Clearly the new formula will be bipartite, and it will be monotone in the negations of the original variables. Also all nodes in one part of the node partition will have degree two.

Lastly, we can make the reduction to a 2/3-graph by replacing each node of degree $k > 3$, (which necessarily all lie in the other part of the partition) by a cyclic structure of $7k$ nodes. Of these, k nodes, say $z_1, z_2, \cdots, z_k$, represent the original node and each of these has one external link. Otherwise they are linked in a cycle, successive pairs being joined by the equality structure (C) above. Construction (C) ensures that this structure enforces equality for the values of the externally linked variables $z_1, z_2, \cdots, z_k$ that represent one original variable, and that the final graph is 3/2Bip. ■

Theorem 2.3 *#$_2$Pl-Rtw-Mon-3CNF is $\oplus$P-complete.*

Proof This follows directly from $\oplus$Pl-3/2Bip-Mon-2CNF being $\oplus$P-complete. In an instance F of $\oplus$Pl-3/2Bip-Mon-2CNF we regard each degree three node as a 3CNF gate with all variables negated, and each degree two node as the location of those (unnegated) variables. Then among solutions to F each time the variables neighboring a degree three node x all have value 1, then two solutions of x are possible, while if some neighbor of x has value 0, then x must have value 1 and only one solution is possible. Hence, the degree three node effectively computes a disjunction on the negations of its three neighboring variables. ■

3 Holographic Algorithms For Size Two Bases

We first summarize the planar matching formulation of holographic algorithms [25, 27]. We then spell out the details for the case of 2-output generators and 3-input recognizers for bases of size 2, thus generalizing the corresponding results for bases of size 1 [27]. Finally we give characterizations for some natural 2-output generators for equality and signed inequality gates.

A generator G over a 2-basis with 2 symbolic outputs is an undirected graph with weights from a field F that has four distinguished output nodes $\{1, 2, 3, 4\}$ and governing equations:

$$u_{ijkl} = \sum q_{rs}(\mathbf{b}_{rs})_{ijkl} \tag{1}$$

Here u_{ijkl} is defined for $(i, j, k, l) \in \{0, 1\}^4$ and equals the value of the perfect matching polynomial (PerfMatch) for graph G, when the output nodes corresponding to the parameters i, j, k, l that have value one are deleted. (Note that if G has an even number of nodes then $u_{ijkl} = 0$ whenever an odd number of i, j, k, l are zero, and vice versa.) The elements q_{rs} for $(r, s) \in \{0, 1\}^2$ represent the

combinatorial function that is being generated, which can be written as $q_{00}\mathbf{n}\otimes\mathbf{n}+q_{01}\mathbf{n}\otimes\mathbf{p}+q_{10}\mathbf{p}\otimes\mathbf{n}+q_{11}\mathbf{p}\otimes\mathbf{p}$. This in turn means that negative symbols are emitted in both directions with weight q_{00}, etc. (For example, for a generator of two equal bits the signature of the gate is $\mathbf{q}=(q_{00},q_{01},q_{10},q_{11})=(1,0,0,1)$.) Now the basis $\mathbf{b}$ is specified as $\{\mathbf{b}_{rs}\mid(r,s)\in\{0,1\}^2\}$ where $\mathbf{b}_{rs}$ denotes the weighted 16-vector $((\mathbf{b}_{rs})_{ijkl}|(i,j,k,l)\in\{0,1\}^4)$ that describes $\mathbf{n}\otimes\mathbf{n}$, $\mathbf{n}\otimes\mathbf{p}$, $\mathbf{p}\otimes\mathbf{n}$, or $\mathbf{p}\otimes\mathbf{p}$, in each of the four cases that $(r,s)=(0,0),(0,1),(1,0)$ or $(1,1)$. Further, we shall represent the 2-basis as $\mathbf{n}=(n_{00},n_{01},n_{10},n_{11})$ and $\mathbf{p}=(p_{00},p_{01},p_{10},p_{11})$.

A recognizer R over a 2-basis with 3 symbolic inputs is an undirected graph with weights from a field F that has six distinguished input nodes, and governing equations:

$$q_{ijk}=\sum u_{rstuvw}(\mathbf{b}_{ijk})_{rstuvw} \tag{2}$$

Here u_{rstuvw} is defined for $(r,s,t,u,v,w)\in\{0,1\}^6$ and equals the perfect matching (PerfMatch) polynomial for that graph when the input nodes corresponding to the parameters r,s,t,u,v,w that have value one are deleted. Now $\mathbf{b}_{ijk}$ for each $(i,j,k)\in\{0,1\}^3$ will denote the weighted 64-vector $((\mathbf{b}_{ijk})_{rstuvw}|(r,s,t,u,v,w)\in\{0,1\}^6)$ that describes $\mathbf{n}\otimes\mathbf{n}\otimes\mathbf{n}$, $\mathbf{n}\otimes\mathbf{n}\otimes\mathbf{p}$, $\mathbf{n}\otimes\mathbf{p}\otimes\mathbf{n}$, $\mathbf{n}\otimes\mathbf{p}\otimes\mathbf{p}$, $\mathbf{p}\otimes\mathbf{n}\otimes\mathbf{n}$, $\mathbf{p}\otimes\mathbf{n}\otimes\mathbf{p}$, $\mathbf{p}\otimes\mathbf{p}\otimes\mathbf{n}$, or $\mathbf{p}\otimes\mathbf{p}\otimes\mathbf{p}$ according to whether (i,j,k) = (0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1). As before, the 2-basis is $\mathbf{n}=(n_{00},n_{01},n_{10},n_{11})$ and $\mathbf{p}=(p_{00},p_{01},p_{10},p_{11})$. The elements q_{ijk} for $(i,j,k)\in\{0,1\}^3$ represent the combinatorial function that is being recognized. For example, for the 3CNF gate we need signature $\mathbf{q}=(q_{000},q_{001},q_{010},q_{011},q_{100},q_{101},q_{110},q_{111})$ = (0, 1, 1, 1, 1, 1, 1, 1). When the signature depends only on the number of 1's in the subscripts of $\mathbf{q}$ then it is a symmetric signature and is abbreviated using square parentheses as $[q_{000},q_{001},q_{011},q_{111}]$, which for 3CNF is therefore [0, 1, 1, 1].

For an instance I of a given computational problem we construct a matchgrid H(I) from generator and recognizer matchgates whose signatures capture the combinatorial constraints that characterize the solutions of I, and join these up by single edges joining outputs of generators to inputs of recognizers. By definition, the Holant of H(I) will then equal the sum of all the solutions of I. The Holant Theorem [25, 27] states that the Holant of H(I) equals the PerfMatch polynomial evaluated for H(I). The PerfMatch polynomial, in turn, will equal the square root of the determinant of the skew symmetric matrix obtained from the adjacency matrix of H(I) by setting signs according to the procedure of Fisher-Kasteleyn-Temperley [16]. Also the determinant of planar positive definite matrices can be computed in time $O(n^{\omega/2})$ [18] where ω is the exponent of matrix multiplication, and hence the determinant of a planar matrix A can be so computed by computing $\det(AA^T)$. From this it follows that the *core* holographic algorithm will always take time $O(n^{\omega/2})<O(n^{1.19})$ [9]. (Of course, in applications, the inputs to it may be larger than for the original problem, for example, if a planarizing transformation is invoked. Also, the core procedure may be called many times if some interpolation operation is employed.)

We shall now observe that for certain natural signatures some general statements can be made about generators governed by equation system (1) above: $u_{ijkl}=\sum q_{rs}(\mathbf{b}_{rs})_{ijkl}$. The output nodes will be labelled 1, 2, 3, 4 in anticlockwise order in the planar embedding. The equation system can be expanded as follows.

$$\begin{aligned}
u_{0000}&=q_{00}n_{00}n_{00}+q_{01}n_{00}p_{00}+q_{10}p_{00}n_{00}+q_{11}p_{00}p_{00},\\
u_{0001}&=q_{00}n_{00}n_{01}+q_{01}n_{00}p_{01}+q_{10}p_{00}n_{01}+q_{11}p_{00}p_{01},\\
u_{0010}&=q_{00}n_{00}n_{10}+q_{01}n_{00}p_{10}+q_{10}p_{00}n_{10}+q_{11}p_{00}p_{10},\\
u_{0011}&=q_{00}n_{00}n_{11}+q_{01}n_{00}p_{11}+q_{10}p_{00}n_{11}+q_{11}p_{00}p_{11},\\
u_{0100}&=q_{00}n_{01}n_{00}+q_{01}n_{01}p_{00}+q_{10}p_{01}n_{00}+q_{11}p_{01}p_{00},\\
u_{0101}&=q_{00}n_{01}n_{01}+q_{01}n_{01}p_{01}+q_{10}p_{01}n_{01}+q_{11}p_{01}p_{01},\\
u_{0110}&=q_{00}n_{01}n_{10}+q_{01}n_{01}p_{10}+q_{10}p_{01}n_{10}+q_{11}p_{01}p_{10},\\
u_{0111}&=q_{00}n_{01}n_{11}+q_{01}n_{01}p_{11}+q_{10}p_{01}n_{11}+q_{11}p_{01}p_{11},\\
u_{1000}&=q_{00}n_{10}n_{00}+q_{01}n_{10}p_{00}+q_{10}p_{10}n_{00}+q_{11}p_{10}p_{00},\\
u_{1001}&=q_{00}n_{10}n_{01}+q_{01}n_{10}p_{01}+q_{10}p_{10}n_{01}+q_{11}p_{10}p_{01},\\
u_{1010}&=q_{00}n_{10}n_{10}+q_{01}n_{10}p_{10}+q_{10}p_{10}n_{10}+q_{11}p_{10}p_{10},\\
u_{1011}&=q_{00}n_{10}n_{11}+q_{01}n_{10}p_{11}+q_{10}p_{10}n_{11}+q_{11}p_{10}p_{11},\\
u_{1100}&=q_{00}n_{11}n_{00}+q_{01}n_{11}p_{00}+q_{10}p_{11}n_{00}+q_{11}p_{11}p_{00},\\
u_{1101}&=q_{00}n_{11}n_{01}+q_{01}n_{11}p_{01}+q_{10}p_{11}n_{01}+q_{11}p_{11}p_{01},\\
u_{1110}&=q_{00}n_{11}n_{10}+q_{01}n_{11}p_{10}+q_{10}p_{11}n_{10}+q_{11}p_{11}p_{10},\\
u_{1111}&=q_{00}n_{11}n_{11}+q_{01}n_{11}p_{11}+q_{10}p_{11}n_{11}+q_{11}p_{11}p_{11},
\end{aligned}$$

Theorem 3.1 *For any F there is a generator for* $\mathbf{q}$ = (0, 1, -1, 0) *over any 2-basis* $\mathbf{n}$ = ($n_{00},n_{01},n_{10},n_{11}$) *and* $\mathbf{p}$ = ($p_{00},p_{01},p_{10},p_{11}$) *provided* $n_{00}p_{11}=n_{11}p_{00}, n_{01}p_{10}=n_{10}p_{01}$, *and* $n_{01}p_{11}\neq n_{11}p_{01}$

Proof Proposition 6.3(ii) in [27] states that for four external nodes there exist planar matchgates for all $\mathbf{u}$ such that
(a) $u_{1111}=u_{0000}=u_{1100}=u_{0011}=u_{1001}=u_{0110}=u_{0101}=u_{1010}=0$,
(b) $u_{1000}u_{0111}-u_{1011}u_{0100}+u_{1101}u_{0010}-u_{1110}u_{0001}=0$, and
(c) $u_{0111}\neq 0$.

The reader can easily verify that the eight equations required for (a) are all satisfied. Equation (b) becomes $(n_{10}p_{00}-p_{10}n_{00})(n_{01}p_{11}-p_{01}n_{11})-(n_{10}p_{11}-p_{10}n_{11})(n_{01}p_{00}-p_{01}n_{00})+(n_{11}p_{01}-p_{11}n_{01})(n_{00}p_{10}-p_{00}n_{10})-(n_{11}p_{10}-p_{11}n_{10})(n_{00}p_{01}-p_{00}n_{01})=0$, which is also satisfied, as is (c). ■

Theorem 3.2 *For any F there is a generator for* **q** = [1, 0, 1] *over any 2-basis* **n** = $(n_{00}, n_{01}, n_{10}, n_{11})$ *and* **p** = $(p_{00}, p_{01}, p_{10}, p_{11})$ *provided* $n_{00}p_{11} = n_{11}p_{00}$, $n_{01}p_{10} = n_{10}p_{01}$, $n_{11}n_{11} + p_{11}p_{11} \neq 0$, and $n_{10}n_{11} + p_{10}p_{11} = n_{01}n_{11} + p_{01}p_{11} = n_{00}n_{10} + p_{00}p_{10} = n_{00}n_{01} + p_{00}p_{01} = 0$.

Proof Proposition 6.3(i) in [27] states that for four external nodes there exist planar matchgates for all **u** such that
(a) $u_{0111} = u_{1000} = u_{0100} = u_{1011} = u_{0001} = u_{1110} = u_{1101} = u_{0010} = 0$,
(b) $u_{0000}u_{1111} - u_{0011}u_{1100} + u_{0101}u_{1010} - u_{0110}u_{1001} = 0$, and
(c) $u_{1111} \neq 0$.

The reader can verify that all the eight equations of (a) are satisfied. Also, (b) becomes $(n_{00}n_{00} + p_{00}p_{00})(n_{11}n_{11} + p_{11}p_{11}) - (n_{00}n_{11} + p_{00}p_{11})(n_{11}n_{00} + p_{11}p_{00}) + (n_{01}n_{01} + p_{01}p_{01})(n_{10}n_{10} + p_{10}p_{10}) - (n_{01}n_{10} + p_{01}p_{10})(n_{10}n_{01} + p_{10}p_{01}) = 0$, which is also satisfied, as is (c). ■

We note that these results generalize the following observations for the size one bases.

Theorem 3.3 *For any F there is a generator for* **q** = (0 ,1 , -1, 0) *over any 1-basis* **n** = (n_0, n_1), **p** = (p_0, p_1).

Theorem 3.4 *For any F there is a generator for* **q**= [1, 0, 1] *over any 1-basis* **n** = (n_0, n_1), **p** = (p_0, p_1) *such that* $n_0n_1 + p_0p_1 = 0$.

For generators with two outputs equation system (1) is

$$
\begin{aligned}
u_{00} &= q_{00}n_0n_0 + q_{01}n_0p_0 + q_{10}p_0n_0 + q_{11}p_0p_0 \\
u_{01} &= q_{00}n_0n_1 + q_{01}n_0p_1 + q_{10}p_0n_1 + q_{11}p_0p_1 \\
u_{10} &= q_{00}n_1n_0 + q_{01}n_1p_0 + q_{10}p_1n_0 + q_{11}p_1p_0 \\
u_{11} &= q_{00}n_1n_1 + q_{01}n_1p_1 + q_{10}p_1n_1 + q_{11}p_1p_1
\end{aligned}
$$

Proposition 6.1 in [27] asserts that any standard signature is possible if either (a) $u_{00} = u_{11} = 0$, or (b) $u_{01} = u_{10} = 0$. Theorem 3.3 follows immediately from case (a), and Theorem 3.4 from case (b).

4 Some Accidental Algorithms

It is easy to see that Pl-Rtw-Mon-3CNF corresponds to a generator for [1, 0, 1] and a recognizer for [0, 1, 1, 1], while Pl-3/2Bip-VC (and hence Pl-3/2Bip-Mon-2CNF) corresponds to a generator for [1, 0, 1] and a recognizer for [2, 1, 1, 1]. For the former the generators correspond to variables, and the recognizers to clauses. For the latter generators correspond to degree two clauses, and the recognizers to degree three clauses. Matchgates and bases for these symmetric signatures were discovered over F_7 using the computer algebra system Singular [11] applied to the formulation of 2-bases given in the previous section.

Theorem 4.1 (*i*) *For* F_7, **n** = (1, 1, 2, 1) *and* **p** = (2, 3, 6, 2) *is such a common basis for generating* [1, 0, 1] *and recognizing* [0, 1, 1, 1]. (*ii*) *For* F_7, **n** = (1, 1, 4, 1) *and* **p** = (3, 2, 1, 3) *is a common basis for generating* [1, 0, 1] *and recognizing* [2, 1, 1, 1].

Proof First we observe that the two bases given satisfy the requirements of [1, 0, 1] generators given in Theorem 3.2. Hence it is sufficient to prove the existence of the recognizers. We need recognizers with six input nodes, and will take the three symbolic inputs as coming into (1, 2), (3, 4) and (5, 6), respectively in anticlockwise order. The set of equations (2) above specifies the governing equations. There are further equations that constrain the **u** variables to be realizable as a PerfMatch polynomial. A standard form for these is given in [27] for up to four external nodes, and for arbitrary numbers of external nodes in [3, 4]. In general the equations for six external nodes will have 32 nonzero **u** variables and many internal variables. These are difficult to handle for humans and sometimes for computers. In the current instance, we use a recognizer with just three nonzero internal variables. The edge weights are x_{16} between nodes 1 and 6, x_{45} between nodes 4 and 5, and x_{78}, between two internal nodes numbered 7 and 8, the latter adding only a multiplicative factor to the recognized values. Noting that these edges need not cross in a planar embedding and hence no crossover constructions are necessary, the standard signature of the matchgate will be characterized by the following

$$
\begin{aligned}
u_{011000} &= x_{16}x_{45}x_{78}, \\
u_{011110} &= x_{16}x_{78}, \\
u_{111001} &= x_{45}x_{78}, \\
u_{111111} &= x_{78},
\end{aligned}
$$

with the sixty remaining **u** components all being zero. Then the signature of the gate with respect to the basis basis $\mathbf{n} = (n_{00}, n_{01}, n_{10}, n_{11})$ and $\mathbf{p} = (p_{00}, p_{01}, p_{10}, p_{11})$ is given by equation system (2), which simplifies by virtue of the sparsity to:

$$
\begin{aligned}
q_{000} &= u_{011000}n_{01}n_{10}n_{00} + u_{011110}n_{01}n_{11}n_{10} \\
&\quad + u_{111001}n_{11}n_{10}n_{01} + u_{111111}n_{11}n_{11}n_{11}, \\
q_{001} &= u_{011000}n_{01}n_{10}p_{00} + u_{011110}n_{01}n_{11}p_{10} \\
&\quad + u_{111001}n_{11}n_{10}p_{01} + u_{111111}n_{11}n_{11}p_{11}, \\
q_{010} &= u_{011000}n_{01}p_{10}n_{00} + u_{011110}n_{01}p_{11}n_{10} \\
&\quad + u_{111001}n_{11}p_{10}n_{01} + u_{111111}n_{11}p_{11}n_{11},
\end{aligned}
$$

$$
\begin{aligned}
q_{011} &= u_{011000}n_{01}p_{10}p_{00} + u_{011110}n_{01}p_{11}p_{10} \\
&\quad + u_{111001}n_{11}p_{10}p_{01} + u_{111111}n_{11}p_{11}p_{11}, \\
q_{100} &= u_{011000}p_{01}n_{10}n_{00} + u_{011110}p_{01}n_{11}n_{10} \\
&\quad + u_{111001}p_{11}n_{10}n_{01} + u_{111111}p_{11}n_{11}n_{11}, \\
q_{101} &= u_{011000}p_{01}n_{10}p_{00} + u_{011110}p_{01}n_{11}p_{10} \\
&\quad + u_{111001}p_{11}n_{10}p_{01} + u_{111111}p_{11}n_{11}p_{11}, \\
q_{110} &= u_{011000}p_{01}p_{10}n_{00} + u_{011110}p_{01}p_{11}n_{10} \\
&\quad + u_{111001}p_{11}p_{10}n_{01} + u_{111111}p_{11}p_{11}n_{11}, \\
q_{111} &= u_{011000}p_{01}p_{10}p_{00} + u_{011110}p_{01}p_{11}p_{10} \\
&\quad + u_{111001}p_{11}p_{10}p_{01} + u_{111111}p_{11}p_{11}p_{11}.
\end{aligned}
$$

The two claims regarding F_7 can be verified directly by substitution. The first claim regarding a recognizer for [0, 1, 1, 1] can be verified by choosing $x_{16} = x_{45} = 1$ and $x_{78} = 2$, so that $u_{011000} = u_{011110} = u_{111001} = u_{111111} = 2$, and substituting **n** = (1, 1, 2, 1) and **p** = (2, 3, 6, 2) in the above 8 equations for **q**. The second claim for F_7 can be verified with $x_{16} = x_{45} = 1$ and x_{78} = -2. ■

Corollary 4.1 *There is a polynomial time algorithm for #$_7$Pl-Rtw-Mon-3CNF.*

Corollary 4.2 *There is a polynomial time algorithm for #$_7$Pl-3/2Bip-Mon-2CNF and #$_7$Pl-3/2Bip-VC.*

The above formulation of matchgates for bases of size 2 has therefore proved successful in yielding surprising polynomial time algorithms using a mechanical equation solver. Larger bases, or gates with larger arities can also be attempted, but these may give systems of equations that are much more challenging to solve mechanically.

However, Cai and Lu [6] have observed very recently that the specific signatures derived in the last result can be obtained also for size one bases using Proposition 6.2 in [27]. In particular, that proposition asserts that there exist three output matchgates with $u_{100} = u_{010} = u_{001} = u_{111} = 0$ and any combination of values for the components $u_{000}, u_{011}, u_{101}$ and u_{110}. Hence equation system (2) for recognizers becomes

$$
\begin{aligned}
q_{000} &= u_{000}n_0n_0n_0 + u_{011}n_0n_1n_1 + u_{101}n_1n_0n_1 \\
&\quad + u_{110}n_1n_1n_0, \\
q_{001} &= u_{000}n_0n_0p_0 + u_{011}n_0n_1p_1 + u_{101}n_1n_0p_1 \\
&\quad + u_{110}n_1n_1p_0, \\
q_{010} &= u_{000}n_0p_0n_0 + u_{011}n_0p_1n_1 + u_{101}n_1p_0n_1 \\
&\quad + u_{110}n_1p_1n_0, \\
q_{011} &= u_{000}n_0p_0p_0 + u_{011}n_0p_1p_1 + u_{101}n_1p_0p_1 \\
&\quad + u_{110}n_1p_1p_0, \\
q_{100} &= u_{000}p_0n_0n_0 + u_{011}p_0n_1n_1 + u_{101}p_1n_0n_1 \\
&\quad + u_{110}p_1n_1n_0, \\
q_{101} &= u_{000}p_0n_0p_0 + u_{011}p_0n_1p_1 + u_{101}p_1n_0p_1 \\
&\quad + u_{110}p_1n_1p_0, \\
q_{110} &= u_{000}p_0p_0n_0 + u_{011}p_0p_1n_1 + u_{101}p_1p_0n_1 \\
&\quad + u_{110}p_1p_1p_0, \\
q_{111} &= u_{000}p_0p_0p_0 + u_{011}p_0p_1p_1 + u_{101}p_1p_0p_1 \\
&\quad + u_{110}p_1p_1p_0,
\end{aligned}
$$

Then **n** = (1, 6) and **p** = (2, 4) is a common basis for generating [1, 0, 1] and recognizing [0, 1, 1, 1] since then the condition of Theorem 3.4 is satisfied, as is the above equation system (2) with $u_{000} = 2$ and $u_{011} = u_{101} = u_{110} = 4$. For the same reason **n** = (1, 6) **p** = (3, 5) is a common basis for generating [1, 0, 1] and recognizing [2, 1, 1, 1], the above system (2) being satisfied with $u_{000} = 5$ and $u_{011} = u_{101} = u_{110} = 6$.

Theorem 4.2 *For $\mathbb{C}$ there are common bases of size one for (i) generating* [1, 0, 1] *and recognizing* [0, 1, 1, -$\frac{4}{3}$] *and for (ii) generating* [1, 0, α] *and recognizing* [0, 1, 1, 1] *where α is a root of $x^2 + 3x + 3 = 0$.*

Proof Part (i) follows with basis **n** = (3,1), **p** = (-1,3) from equation system (2) with $u_{000} = -1/60$ and $u_{011} = u_{101} = u_{110} = 1/20$, and condition $n_0n_1 + p_0p_1 = 0$ of Theorem 3.4. Part (ii) follows in the same way by observing that for generating [1, 0, α] the condition in Theorem 3.4 becomes $n_0n_1 + \alpha p_0p_1 = 0$. Then basis **n** = (1,1), **p** = (-α/3, α + 2) with $u_{000} = 3\sqrt{3}i/4$ and $u_{011} = u_{101} = u_{110} = -\sqrt{3}i/4$ works. ■

We note that Theorems 3.1 and 3.3 correspond to read-twice formulae in which variables occur in opposite pairs (x, x'). Clearly they have fewer constraints than Theorems 3.2 and 3.4. The question arises as to whether interesting polynomial time algorithms can be derived using the former pair. When combined with a recognizer for [0, 1, 1, 1] the problem solved is $\oplus$Pl-Rtw-Opp-3CNF, which is degenerate. However, it is not clear whether in combination with recognizers for $[r_0r_1, r_2, r_3]$ for other $r_0, r_1, r_2, r_3 \in \mathbb{C}$ interesting problems can be solved.

When applying the term accidental to an algorithm we intend to point out that the algorithm arises from satisfying an apparently onerous set of constraints. We do no imply that the existence of *some* efficient algorithm for that problem is necessarily unlikely. It can be argued that all polynomial time algorithms are accidental in this sense, and one may point to efficient linear algebra algorithms for the determinant as paradigmatic examples. Indeed, one might also argue that the completeness results in this paper are no less accidental, and further that all completeness results in complexity theory are accidental in the same sense.

5 Scope and Limits of Holographic Algorithms

Holographic reductions may be applied in a variety of ways. In this section we shall restrict our discussion to their use in matchgrids as described in [25, 27] and in earlier sections of this paper. We shall refer to algorithms obtained by polynomial time reductions to matchgrids as *matchgrid algorithms*.

The principal question, clearly, is whether matchgrid algorithms exist for problems complete in NP, #P, or #$_k$P for some k. One advantage of the holographic framework is that it is a natural algebraic framework in which one expects to be able to resolve mathematical questions. We would argue that there is little intuitive guidance available as to the ultimate answer to the question posed other than through analysis. Holographic algorithms have been studied only for a short time and the failure to find complexity class collapsing algorithms to date is not a persuasive argument for strongly conjecturing a negative answer. We also note that even for the strongest positive result that might be proved this way, namely P$^{\#\mathrm{P}}$ = NC2, the search over some decades for promising approaches to proving the contrary appears to have yielded no strong candidates.

In this section we first define the notion of an elementary encoding of 3CNF into matchgrids. We shall show that 3CNF does not have such an elementary encoding. We then go on to observe that we currently know of no impediments to either (i) the encodability into matchgrids of these same problems by more general reductions, or to (ii) the existence of elementary matchgrid algorithms for some #P-complete problems.

We say that a reduction f to matchgrids is a *k(m)-oracle* reduction if for instances of size m it generates $k(m)$ matchgrids in polynomial time and from their Holants it computes solutions to the original problem also in polynomial time. The reductions in the previous section were 1-oracle reductions. Note that multi-oracle matchgrid reductions have been described [27], [4].

Suppose that f is a polynomial time reduction from a class of 3CNF formulae to matchgrids over field F. Let R be a 3CNF formula with m clauses in that class, and let X be a set of n clauses $X_1, \cdots, X_n$ of R. For $1 \leq i \leq n$ let U_i be the pair of clauses $\{(x_i + x_i + x_i), (x_i' + x_i' + x_i')\}$. These will be used to enforce particular input values. (N.B. We are insisting here on exactly three literals per clause only for consistency of notation.) Let $f(R, X)$ be the set of 2^n adjacency matrices for the set of matchgrid images under f of all the formulae that can be obtained from R by replacing each clause X_i by a member of U_i. For a two argument function $L(n, m)$ we say that f is $L(n, m)$-*local* if for each R and X the following holds: the set of matrices in $f(R, X)$ are all of the same size and have identical entries in all positions except for a set $Y(R, X)$ of at most $L(n, m)$ positions. It is *local* if it is $L(n, m)$-local for some L. We say that such a transformation is *strictly local* if it is $L(n, m)$-local for some function $L(n, m)$ that is upper bounded by a polynomial $L^*(n)$ that is independent of m.

For a local reduction f, a 3CNF formula R and a set X of n of its clauses define $Z(R, X)$ to be the indices of the rows and columns that contain some $Y(R, X)$ position, and let $T(R, X)$ be the remaining indices. Then f is a *boundary* reduction if for each R and X there is a set of planar embeddings of the members of $f(R, X)$, one embedding for each combination of replacements of the set X of clauses, such that (i) the $Z(R, X)$ nodes have constant bounded degree, (ii) the embedded subgraph induced by the vertices $T(R, X)$ is identical for all members of $f(R, X)$, (iii) for all members of $f(R, X)$ the nodes $Z(R, X)$ are all mapped in the infinite outer face of the embedding of the graph induced by $T(R, X)$, and (iv) in each member of $f(R, X)$ the $Y(R, X)$ edges (all incident to vertices in $Z(R, X)$) can be partitioned into $|X|$ sets, each such set S_i corresponding to a clause X_i in X, such that the weights of S_i are functions of the choice of replacement clause for X_i and are independent of the choices of replacement clauses for X_j for any $j \neq i$.

We shall say that a matchgrid algorithm for a class of 3CNF formulae is an *elementary* matchgrid algorithm if it consists of a reduction to matchgrids that has the four properties of being (i) 1-oracle, (ii) strictly local, (iii) boundary, and (iv) over a field F of $|F| <$ poly(m) elements, where m is the size of the formula.

We note that the simplest attempt at a holographic algorithm for 3CNF would consist of fixed size gates for the variables, for the Boolean gates, for fanout gates, and for cross-overs of the connections. If this were possible then the inputs could be simulated on the periphery and requirements (i)-(iii) of elementarity would be realized. This possibility we shall exclude, at least if requirement (iv) of a polynomial bound on the number of elements of $|F|$ is also met.

Our negative results are based on the following theorem, which asserts for matchgrids the fundamental equivalent result proved for matchcircuits by Cai and Choudhary (Corollary 4.1 in [3].)

Theorem 5.1 *For any matchgrid with r external nodes there is another matchgrid with the same standard signature that has $O(r^4)$ edges of which $O(r^2)$ have weight different from 1.*

Proof The equivalence of matchgrids and matchcircuits follows from Lemmas 3.1 and 3.2 in [4]. The result is then immediate from Corollary 4.1 in [3]. ■

Theorem 5.2 *There is no elementary matchgrid algorithm for 3CNF.*

Proof Consider a Boolean circuit C with n inputs and $m \geq n-1$ gates, and consider an input vector $\mathbf{x}$ for it of n binary values. This can be mapped by standard methods to a 3CNF Boolean formula $R(\mathbf{x})$ of O(m) clauses such that $R(\mathbf{x})$ is satisfiable in 1 or 0 ways according to whether C evaluates to 1 or 0 on input vector $\mathbf{x}$. Further, for each circuit C there will be 2^n formulae $R(\mathbf{x})$ so generated, one for each $\mathbf{x} \in$

$\{0,1\}^n$, such that these circuits are identical except for n clauses each of the form $(x_i+x_i+x_i)$ or $(x_i'+x_i'+x_i')$ that enforce the actual value of the variable x_i in input vector $\mathbf{x}$. We shall designate these n clauses that encode the inputs to C as the set X. Hence, any strictly local transformation from formulae to matchgrids will map these 2^n formulae, which differ only in the n clauses X, to matchgrids that are identical to each other in every entry except for a fixed set of $L^*(n)$ entries, where $L^*(n)$ is upper bounded by a polynomial in n independent of m.

Suppose now that there existed an elementary matchgrid algorithm that maps formulae $R(\mathbf{x})$ to matchgrids whose Holant determines whether $R(\mathbf{x})$ is satisfiable, and hence whether $C(\mathbf{x})$ equals 0 or 1. Then for any circuit C the class $R(\mathbf{x})$ of formulae derived from it would map to matchgrids with identical matrices except for a fixed set of at most $L^*(n)$ entries. We regard the nodes in $T(R,X)$ as a matchgrid H, and regard those of the nodes on the outer face of the assumed embedding of H that have connections to $Z(R,X)$ nodes as the external nodes of H. There are $\mathrm{O}(L^*(n))$ such nodes if the $Z(R,X)$ nodes have constant bounded degree. We can then deduce from Theorem 5.1 that the standard signatures of the matchgrids in $f(R,X)$ are unchanged if in each the submatchgrid H is replaced by a matchgrid H^* that has only $\mathrm{O}((L^*(n))^2)$ internal edges with weights not equal to 1.

We next give an upper bound on the number of distinct sets $f(R,X)$ of matchgrids (and hence inequivalent circuits C) that can be obtained in this way by varying R. From the previous paragraph the number of distinct choices of H^* is $|F|$ to the power $O((L^*(n))^2)$. It remains to analyze the contribution of the edges external to H as constrained by part (iv) of the definition of a boundary reduction. Suppose that the first clause of X influences $|S_1|$ of the edges in $Y(R,X)$. Then the number of choices of weights for these edges is $|F|$ to the power $|S_1|$. This upper bounds the number of different encodings of a clause that can go in that position. If we want to encode just the two choices, the clauses $(x_i+x_i+x_i)$ or $(x_i'+x_i'+x_i')$, then the number of choices is $|F|$ to the power $2|S_1|$. Hence over all n members of X the possible choices will be $|F|$ to the power $2L^*(n)$. There will be a further $\mathrm{O}(L^*(n))$ edges in the exterior of H^* since the degrees of those nodes outside of H^* have constant bounded degree. Hence the number of distinct sets $f(R,X)$ that can be encoded is at most $|F|$ to the power $\mathrm{O}((L^*(n))^2)+O(L^*(n))$. We therefore conclude that if an elementary reduction exists then the number N of inequivalent circuits that can be encoded is upper bounded by $|F|$ to the power $\mathrm{O}((L^*(n))^2)$.

Finally suppose, that we encode all circuits C of n inputs of size up to 2^n. Then each one of double exponentially many Boolean functions of n arguments will have such an encoding. But this is a contradiction, since $|F|$ is single exponential in n, and hence N is also since it is $|F|$ to the power $\mathrm{O}((L^*(n))^2$. Therefore the hypothesized elementary matchgrid algorithm cannot exist. ■

Corollary 5.1 *There is no elementary matchgrid algorithm for Pl-3CNF.*

Proof Lichtenstein [17] showed that 3CNF can be simulated by Pl-3CNF by means of crossover gadgets, and his reduction preserves numbers of solutions [13]. Hence Pl-3CNF can be used to encode circuit evaluation exactly as 3CNF can. The result then follows exactly as in the above Theorem. ■

It is natural to ask whether elementary matchgrid algorithms exist for complete problems in which circuit evaluation is encoded less directly. Some natural candidates are those #P-complete problems the reductions to which are more algebraic. Of course, we have defined elementary algorithms only for 3CNF, but corresponding definitions can be developed for other problems also. Here we shall observe that at least the most naive approaches can be excluded for the permanent.

Consider $\{0,\,1\}$-Perm$_k$ the problem of finding permanents of matrices with $\{0,\,1\}$ entries modulo a fixed integer k, or equivalently counting perfect matchings in bipartite graphs modulo k. Now the reduction of circuit evaluation via 3CNF and Pl-3CNF to this for k not a power of 2, using [22] for the last step, gives a graph in which the input variables are represented on the outer face. Also, the formula obtained from an instance of the circuit evaluation problem will always have zero or one solution. Any straightforward mapping of this graph in which the nodes and edges are replaced by constant size matchgates over a field of polynomial number of elements would produce an elementary matchgrid algorithm for 3CNF. That would contradict Theorem 5.2.

There are multitudes of #P-complete problems that are proved complete via reductions that involve multiple oracle calls and polynomial interpolation on the results [23, 15, 21, 28]. For any one of these problems one can ask in the manner of the previous paragraph whether specific classes of matchgrid algorithms are impossible. It seems that the circuit evaluation problem is more deeply hidden within these problems than in 3CNF or the Permanent.

Open Problem 1 For #2CNF (or another problem proved #P-complete by interpolation) can a natural class of matchgrid algorithms be excluded?

In the opposite direction one can ask:

Open Problem 2 Is there a #P-complete problem for which an elementary matchgrid algorithm exists?

For #P-complete problems in general, and #3CNF itself, there remains the possibility of non-elementary matchgrid

algorithms, ones in which one or more of the four properties specified in the definition of elementarity are relaxed. Thus, it is an open problem whether giving up strict locality, or having many oracle calls, or having the input variables of C map to the interior, or working in fields with exponentially many elements in terms of the input size, would, singly or in combination, allow for matchgrid algorithms.

Open Problem 3 Is there a matchgrid algorithm for #3CNF.

The nature of our lower bound argument can be understood in the following way. In traditional views of computation circuit evaluation is trivial and exponential summation or nondeterminism is apparently problematic. In the holographic view the situation is the exact converse, exponential summation or nondeterminism is trivial and circuit evaluation is problematic (though the components of it Boolean gates, constants, crossovers and fanout can be easily realized separately.) It is therefore not surprising that the circuit evaluation task would appear prominently in lower bound arguments. Note that it is implicit in Theorem 5.2 that circuit evaluation itself does not have "elementary" matchgrid algorithms. It remains open whether the evaluation of circuits or of simple P-complete cellular automaton rules (e. g. [19]) can be done by non-elementary matchgrid algorithms.

References

[1] H. Aizenstein and L. Pitt. Exact learning of read-twice dnf formulas. *Proc. of the 32th Symposium on Foundations of Computer Science*, pages 170–179, 1991.

[2] R. Bubley and M. Dyer. Graph orientations with no sink and an approximation for a hard case of #SAT. *ACM SODA*, pages 248–257, 1997.

[3] J.-Y. Cai and V. Choudhary. On the theory of matchgate computations. *ECCC*, 018, 2006.

[4] J.-Y. Cai and V. Choudhary. Results on matchgates and holographic algorithms. *ICALP 2006,* LNCS Springer-Verlag, 4051:703–714, 2006.

[5] J.-Y. Cai and V. Choudhary. Valiant's Holant theorem and matchgate tensors. *TAMC 2006,* LNCS Springer-Verlag, 3959:248–261, 2006.

[6] J.-Y. Cai and P. Lu. personal communication, June 2006.

[7] M. Cook and J. Bruck. Implementability among predicates. Technical report, California Institute ofTechnology, Pasadena, CA, 2005.

[8] S. A. Cook. The complexity of theorem proving procedures. *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[9] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. of Symbolic Computation*, 9:251–280, 1990.

[10] T. Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1-2):281–293, 2001.

[11] G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 3.0, a computer algebra system for polynomial computations. Centre for Computer Algebra, University of Kaiserslautern, 2005. http://www.singular.uni-kl.de.

[12] T. Hancock. Learning 2-DNF formulas and k-decision trees. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 199–209, Santa Cruz, California, 1991. Morgan Kaufmann.

[13] H. Hunt, M. Marathe, V. Radhakrishnan, and R. Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167, 1998.

[14] H. B. Hunt and R. E. Stearns. The complexity of very simple boolean formulas with applications. *SIAM J. Comput.*, 19(1):44–70, 1990.

[15] M. Jerrum. Two-dimensional monomer-dimer systems are computationally intractible. *J. Stat. Phys.*, 48:121–134., 1987.

[16] P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, New York, 1967.

[17] D. Lichtenstein. Planar formulae and their uses. *SIAM J. on Computing*, 11:329–343, 1982.

[18] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16(2):346–358, 1979.

[19] T. Neary and D. Woods. P-completeness of cellular automaton rule 110. *ICALP 2006,* Lecture Notes in Computer Science, Springer-Verlag 4051, 132-143, 2006.

[20] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. *Theoretical Computer Science*, pages 269–276, 1983.

[21] S. P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. on Computing*, 31:398–427, 2001.

[22] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[23] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8:410–421, 1979.

[24] L. G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM J. on Computing*, 31(4):1229–1254, 2002.

[25] L. G. Valiant. Holographic algorithms (extended abstract). *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 306–315, Oct 17-19 2004. IEEE Press.

[26] L. G. Valiant. Completeness for parity problems. In *Proc. 11th International Computing and Combinatorics Conference* Aug 16-19, Kunming, China, LNCS, volume 3959. Springer-Verlag 1-9, 2005.

[27] L. G. Valiant. Holographic algorithms, ECCC Report. TR05-099, 2005.

[28] M. Xia and W. Zhao. #3-regular bipartite planar vertex cover is #P-complete. *TAMC 2006,* LNCS, 3959:356–364, 2006. Springer-Verlag.