Holographic Algorithms*

Leslie G. Valiant
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
valiant@seas.harvard.edu

Abstract

Complexity theory is built fundamentally on the notion of efficient reduction among computational problems. Classical reductions involve gadgets that map solution fragments of one problem to solution fragments of another in one-to-one, or possibly one-to-many, fashion. In this paper we propose a new kind of reduction that allows for gadgets with many-to-many correspondences, in which the individual correspondences among the solution fragments can no longer be identified. Their objective may be viewed as that of generating interference patterns among these solution fragments so as to conserve their sum.

We show that such holographic reductions provide a method of translating a combinatorial problem to finite systems of polynomial equations with integer coefficients such that the number of solutions of the combinatorial problem can be counted in polynomial time if one of the systems has a solution over the complex numbers. We derive polynomial time algorithms in this way for a number of problems for which only exponential time algorithms were known before.

General questions about complexity classes can also be formulated. If the method is applied to a #P-complete problem then polynomial systems can be obtained the solvability of which would imply $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}$.

1 Introduction

Efficient reduction is perhaps the most fundamental notion on which the theory of computational complexity is built. The purpose of this paper is to introduce a new notion of efficient reduction, called a *holographic reduction*. In a classical reduction an instance of one problem is mapped to an instance of another by replacing its parts by certain *gadgets*. Solution fragments of the first problem will correspond in the gadgets to solution fragments of the second problem. For example, when mapping a Boolean satisfiability problem to a graph theory problem, each way of satisfying a part of the formula will correspond to a way of realizing a solution to the graph theory problem in the gadget. In classical reductions the correspondence between the solution fragments of the two problems is essentially one-to-one, or possibly many-to-one or one-to-many. In a holographic reduction the *sum* of the solution fragments of one problem maps to the *sum* of the solution fragments of the other problem for any one gadget, and does so in such a way that the sum of all

*This research was supported in part by grants NSF-CCR-98-77049, NSF-CCR-03-10882, NSF-CCF-04-27129 and by the National Security Agency (NSA) and Advanced Research and Development Activity (ARDA) under Army Research Office (ARO) contract DAAD19-01-1-0506. Preliminary versions of this paper appeared in the Proceedings of the 45th Symp. on Foundations of Computer Science, IEEE Press, (2004), 306-315, and as Electronic Colloquium on Computational Complexity, Report 99 (2005).

the overall solutions of the one will map to the sum of all the overall solutions of the other. The gadgets therefore map solution fragments *many-to-many*. The main innovation this allows is that it permits reductions in which correspondences between the solution fragments of the two problems need no longer be identifiable at all. Their effect can be viewed as that of producing interference patterns among the solution fragments, and they are called *holographic gadgets* for that reason.

A holographic reduction from a problem A to a problem B is of particular interest when for problem B the sum of the solutions is efficiently computable, since then a polynomial time algorithm for summing the solutions of A is implied. We call algorithms so derived *holographic algorithms*. In this paper we give holographic alogrithms for a number of problems for which no polynomial time algorithms were known before. We obtain these algorithms by reduction to the algorithm for finding perfect matchings in planar graphs due to Fisher, Kasteleyn and Temperley, [Fisher, 1961; Kasteleyn, 1961; Temperley and Fisher, 1961].

We consider holographic reductions and algorithms to be novel notions in algorithmic theory that do not appear to have been explored before even in disguise, and that potentially open up new approaches to the central questions of complexity theory.

The most intriguing question, clearly, is whether polynomial time holographic algorithms exist for NP- or #P-complete problems. For such a result a holographic reduction would have to be exhibited from, say, a #P-complete problem such as planar matchings, to a known polynomial time computable problem, such as planar perfect matchings. We shall show that the existence of such a reduction would be implied by the solvability of a finite system of polynomial equations that define the holographic gadgets used in the reduction. In this sense the search for fast algorithms can be semi-mechanized if computer algebra systems are invoked for solving the systems. It suffices to find a fixed set of such gadgets. We note that the search process itself is NP-hard in the size of the tested system. On the other hand one can expect that any fast algorithms so discovered would rely on algebraic relationships, possibly exotic, which have not been explored before even implicitly.

What is the role of holographic reductions in complexity theory if it is the case that there exist no polynomial algorithms to be discovered for NP- or #P-complete problems? In that eventuality we suggest that any proof of $\mathrm{P} \neq \mathrm{NP}$ may need to explain, and not only to imply, the unsolvability of our polynomial systems. Furthermore explanations of such unsolvabilities may then stand equally in the way of any proofs of $\mathrm{P} \neq \mathrm{P}^{\#\mathrm{P}}$, $\mathrm{P} \neq \mathrm{BPP}$, $\mathrm{P} \neq \mathrm{QBP}$, $\mathrm{P} \neq \mathrm{NC2}$ and $\mathrm{P} = \mathrm{PSPACE}$. Since the solvability of a polynomial system for an explicit combinatorial constraint is a very natural mathematical problem, our approach may be viewed as one offering a restricted model of computation in which one expects mathematical questions to be resolvable, and one that we suggest may be difficult to evade.

Holographic algorithms are inspired by the quantum computational model [Deutsch, 1985; Bernstein and Vazirani, 1997]. However, they are executable on classical computers and do not need quantum computers. They can be understood best, perhaps, in terms of cancellations in classical computation. Strassen's algorithm for matrix multiplication [Strassen, 1969] offers an early striking example of the power of computations that compute extraneous terms only to cancel them later. It is known that cancellations can provide exponential speedups in computations, and in the several cases that have been analyzed, linear algebra algorithms for computing the determinant play a major role [Valiant, 1980; Jerrum and Snir, 1982; Tardos, 1987]. Further the actual cancellations that are performed by certain of these determinant algorithms can be made explicit [Valiant, 1992; Mahajan and Vinay, 1999]. Holographic algorithms offer a new source of cancellation that is not provided by linear algebra alone. Most importantly the cancellations required for the particular problem at hand can be *custom designed* into the holographic gadgets.

The substance of the holographic method as pursued here involves devising an appropriate *basis* for the reduction, and then designing *matchgates* to realize the gadgets. Matchgates have been used previously [Valiant 2000a] but only in the context of classical rather than holographic reductions. We note that the sum of solutions in matchgate constructions corresponds to the Pfaffian, which is polynomial time computable. The examples in this paper all refer to planar structures, because in that case we can use the elegant FKT route to the Pfaffian that makes the design of the gadgets easier. In principle, the FKT technique can be applied to nonplanar structures by having matchgates to simulate cross-overs. We note also that the holographic technique may be used, in principle, to reduce problems to any problem in which a quantity is known to be polynomial time computable. For example, in [Valiant 2005a] it is used in reductions to the general Pfaffian.

2 List of Problems

We first note that the range of natural graph-theoretic problems for which the number of solutions has been known to be countable in polynomial time for arbitrary inputs is very small (Jerrum [2003], Welsh [1993], Valiant [1979a,b]). The prime examples have been the Kirchhoff matrix tree algorithm for spanning trees in arbitrary graphs, and the FKT perfect matching algorithm for planar graphs. For planar graphs there is a positive result known for a further important case, intimately related to the Ising problem in physics, that is known to be obtainable from the perfect matchings problem. This is #PL-CUT—given a planar graph G and a number k the problem is to compute the number of two-colorings of the nodes of G such that exactly k edges have ends of opposite color [Kasteleyn 1967, Lovász and Plummer 1986]. The maximum k for which this number is nonzero is the well-known PL-MAXCUT problem [Orlova and Dorfman, 1972 and Hadlock, 1975].

We now list some problems for which we can provide polynomial time solutions where none apparently were known. They are motivated by their apparent proximity to known NP-, $\oplus$P- and #P-complete problems. They all have a counting, or #P, aspect, but for some we specify a decision or parity version when that is appropriate. We note that in a graph $G = (V, E)$ a subset $E' \subseteq E$ *saturates* a vertex $v \in V$ if v is the endpoint of some edge in E'.

First we consider a matching problem. Jerrum [1987] showed that counting the number of (not necessarily perfect) matchings in a planar graph is #P-complete, and Vadhan [2001] subsequently proved that this was true even for planar bipartite graphs of degree six. For degree two the problem can be solved easily and one might have conjectured that all other nontrivial cases are #P-complete. However, we have a polynomial time algorithm for the following:

#X-MATCHINGS

Input: A planar weighted bipartite graph $G = (V, E, W)$ where V has bipartition $V1$, $V2$ and the nodes in $V1$ have degree 2.
Output: The sum of the masses of all matchings of all sizes where the mass of a matching is the product of (i) the weights of all the edges present in the matching, as well as of the quantity, (ii) "$-(w_1 + \ldots + w_k)$" for all the $V2$ nodes that are not saturated, where $w_1, \ldots, w_k$ are the weights of the edges incident to that (unsaturated) node.

One instance of this is where every $V2$ node has degree 4 and every edge has weight one. Then computing #X-MATCHINGS gives the number of matchings, but each weighted by $(-4)^k$ where k is the number of unsaturated $V2$ nodes. Computing this mod 5, for example, gives the number

of matchings mod 5. Another instance is where every V2 node has degree three and every edge weight 1. Then #X-MATCHINGS is the sum of the matchings, each weighted by $(-3)^k$ where k is the number of V2 nodes not saturated by that matching.

Now we consider a coloring problem. A *functional orientation of an undirected multigraph* G is an assignment of directions to a set of edges so that there is exactly one edge directed away from each node of G. (Note that if two nodes are connected by two edges then these can both have (opposite) directions. Also any single edge may be assigned two opposite directions. The edges of G that are not assigned a direction remain undirected.)

PL-FO-2-COLOR

Input: A planar multigraph graph $G = (V, E)$ of maximum degree 3.
Output: 1 iff there is some coloring of the nodes with two colors and a functional orientation of G such that every edge that joins two nodes of the same color is directed in at least one direction by the functional orientation.
Comment: The problem of (2,1)-coloring with defects is that of 2-coloring a graph so that no node is adjacent to more than one other node of the same color. This is NP-complete for planar graphs of degree 5 [Cowen, Goddard and Jesurum, 1997]. It can be deduced that PL-FO-2-COLOR is NP-complete for degree 10 by means of the following reduction. For an instance of (2,1)-coloring one replaces each edge by a pair of edges between the same pair of nodes. Then if these nodes are given the same color the rules of PL-FO-2-COLOR ensure that the two edges are both oriented and in opposite directions. But then no other neighbor of either of the nodes can have the same color because the corresponding statement for those would imply that there are two edges directed away from that common node.

Our next two problems can be viewed as planar formula problems in the sense of Lichtenstein [1982]: A planar formula is a planar graph where a node can represent a clause or a variable, and an edge links a node representing a variable with a node representing a clause in which it occurs, either negated or not negated.

$\oplus$PL-EVEN-LIN-2

Input: A planar formula where each clause is a linear equation over $GF[2]$ with an even number of occurrences of variables, a subset of the clauses that are considered compulsory to satisfy, a setting to a subset of the variables on the outer face to constants, and an integer k.
Output: The parity of the number of solutions that satisfy exactly k of the equations, including all of the compulsory ones, and the boundary conditions.
Comment: This generalizes $\oplus$PL-CUT, which is the same problem restricted to equations with just two variables and no compulsory equations and can be solved by classical reduction to FKT. The nonplanar version with two variables is NP-and $\oplus$P-complete via known parsimonious reductions, and strong hardness of approximation results are also known [Hastad, 2001]. For odd length equations the corresponding planar problem is $\oplus$P-complete since the corresponding nonplanar problem can be reduced to it using the construction of crossovers from Jerrum [1987]. This construction requires the equations in the crossovers to be compulsory, and without such compulsory equations the completeness of the problem is apparently unresolved.

 # PL-3-NAE-SAT

Input: A planar formula F consisting of NOT-ALL-EQUAL gates of size 3.
Output: The number satisfying assignments of F.
Comment: For connectives other than NOT-ALL-EQUAL (e.g., OR, EXACTLY-ONE) for which the unrestricted decision problem is NP-complete, the corresponding planar decision and counting problems are, in general, NP- and #P-complete, respectively [Hunt, Marathe, Radhakrishnan and Stearns, 1998]. The existence problem for monotone PL-3-NAE-SAT is reducible to the Four Color Theorem and, therefore, always has a solution [Barbanchon, 2004]. Note, however, that the counting problem for the 4-colorings of planar graphs is #P-complete [Vertigan and Welsh, 1992]

PL-NODE-BIPARTITION

Input: A planar graph $G = (V, E)$ of maximum degree 3.
Output: The cardinality of a smallest subset $V' \subseteq V$ such that the deletion of V' and its incident edges results in a bipartite graph.
Comment: This problem is known to be NP-complete for maximum degree 6 [Krishnamoorthy and Deo, 1977]. See Lewis and Yannakakis [1980] for a general approach to such "node deletion" problems. We note that numerous other planar NP-complete problems, such as Hamiltonian cycles and minimum vertex covers are NP-complete already for degree 3 (e.g., Garey, Johnson and Stockmeyer [1976], and Garey and Johnson [1977]).

We now consider "ice" problems that have been widely investigated by statistical physicists. An *orientation* of an undirected graph G is an assignment of a direction to each of its edges. An "ice problem" involves counting the number of orientations such that certain local constraints are satisfied. Pauling [1935] originally proposed such a model for planar square lattices, where the constraint was that an orientation had to have two incoming and two outgoing edges at every node. The question of determining how the number of such orientations grows for various such planar *repeating* structures has been analyzed [Lieb, 1967a-d, Baxter, 1982, see also Welsh, 1993].

#PL-3-NAE-ICE

Input: A planar graph $G = (V, E)$ of maximum degree 3.
Output: The number of orientations such that no node has all the edges directed towards it or away from it.

We next turn to a covering problem. For a graph $G = (V, E)$ a *cycle* is a sequence of edges through distinct nodes that starts and ends at the same node. A *chain* is a sequence of edges through distinct nodes that starts and ends at distinct nodes. A *cycle-chain cover* in G is a set of cycles and chains that saturates every node of G. For real numbers x, y the (x, y)*cycle-chain sum* of G is the sum over all cycle-chain covers C of $x^i y^j$ where i is the number of cycles in C and j is the number of chains. For example, the (2, k) cycle-chain sum for $k = 0$ or $k = 4$ is complete for $\oplus$P for general graphs since the parity of the number of Hamiltonian cycles is reducible to it. In the planar case it is known that counting the number of Hamiltonian cycles for planar cubic graphs is #P-complete [Liskiewicz, Ogihara, and Toda, 2003]. Their proof can be adapted to show that if nodes both of degree 2 and 3 are allowed then planar Hamiltonian cycles is $\oplus$P-complete [Valiant, 05b]

#PL-3-(1,1)-CYCLECHAIN

Input: A planar regular graph $G = (V, E)$ of maximum degree 3.
Output: The (1, 1) cycle-chain sum.

As we shall further elaborate in Section 9, the proofs given there of the last four of these results imply that they can be derived also by classical reduction to #PL-CUT. However, some of these problems also have degree four variants for which such classical reductions are not apparent.

3 Evaluating Planar Matching Polynomials

We shall first describe the basic graph-theoretic notions that we shall use. A (weighted undirected) *graph* G is a triple (V, E, W) where V is the set of n nodes, labeled $\{1, \cdots, n\}$, E is the set of edges where an edge is a pair (i, j) of distinct nodes i, $j \in V$, and W is an assignment of a weight $W(i, j)$ from a field F to each edge (i, j). An edge e is *incident to* or *saturates* a node j if j is one of the pair of nodes of e. A *matching* in G is a set $E' \subseteq E$ of edges such that if e_1 and e_2 are distinct edges in E' then e_1 and e_2 saturate disjoint pairs of nodes. A matching E' *saturates* the union of the node pairs saturated by the member edges of E'. The set of nodes saturated by E' we call $satu(E')$. A matching is *perfect* if it saturates all of V.

With a graph G we associate the *perfect matching* polynomial PerfMatch(G) over $n(n-1)/2$ variables $\{x_{i,j} | 1 \leq i < j \leq n\}$ as follows:

$$\text{PerfMatch}(G) = \sum_{E'} \prod_{(i,j) \in E'} x_{i,j}$$

where the summation is over all perfect matchings E' of G. We shall also discuss the more general *matching sum* polynomial for graphs $G = (V, E, W, \Lambda)$ where Λ further specifies a labeling of each node i by a weight $\lambda_i \in F$. It is defined as:

$$\text{MatchSum}(G) = \sum_{E'} \prod_{i \notin satu(E')} \lambda_i \prod_{i,j \in E'} x_{i,j}$$

where summation is over all, not necessarily perfect, matchings in G. Clearly in the case that every $\lambda_i = 0$, PerfMatch(G) = MatchSum(G). We shall call nodes with $\lambda_i \neq 0$ *omittable* since matchings that omit them can contribute to the MatchSum.

For all polynomials we shall assume, where not otherwise specified, that the coefficients are taken from an arbitrary field F.

A remarkable fact, expressed by the following theorem, is that for *planar* graphs PerfMatch(G) can be expressed as a determinant of an easily computed matrix [Fisher, 1961; Kasteleyn, 1961, 1967; Temperley and Fisher, 1961; Jerrum, 2003]. It follows that PerfMatch(G) can be computed using standard linear algebra algorithms for the determinant.

Theorem 3.1. *There is a polynomial time computable function f that given a planar embedding of a planar graph $G = (V, E, W)$ defines $f{:}E \to \{-1, 1\}$ such that for the antisymmetric matrix M defined so that for all $i < j$*

(i) *if $(i, j) \notin E$ then $M_{i,j} = M_{j,i} = 0$, and*

(ii) *if $(i, j) \in E$ then $M_{i,j} = f(i, j)W(i, j)$ and $M_{j,i} = -f(i, j)W(i, j)$,*

it is the case that $\mathrm{PerfMatch}(G) = \mathrm{Pfaffian}(M) = \sqrt{\mathrm{Det}(M)}$.

In our applications we shall form graphs from fixed sets of standard components called matchgates that simulate particular combinatorial constraints, such as equality. The weights to be used in such matchgates will be elements of F obtained potentially by computationally solving systems of polynomial equations. It is therefore useful to observe that in this general setting the $\mathrm{Det}(M)$ and hence also $\mathrm{PerfMatch}(G)$ can be solved in polynomial time if the field is that of the complex numbers $\mathbb{C}$. The proof is given in Section 12.

Theorem 3.2. *Let Y be any finite subset of $\mathbb{C}$. Suppose that each element of Y can be computed to n decimal places, i.e., absolute error less than 2^{-n}, in time polynomial in n. Let $\{M_n \mid n \geq 1\}$ be a family of matrices where, for each n, M_n is $n \times n$, has every entry from Y, and has an integer valued determinant. Further, suppose that there is a polynomial time algorithm that given input $\{1^n, i, j\}$ will identify the element from Y that is the $(i,j)^{th}$ entry of M_n. Then there is a polynomial time deterministic algorithm that, given 1^n will compute the determinant of M_n.*

Computing MatchSum for planar graphs is known to be #P-complete [Jerrum, 1987]. Since matchings with omittable nodes are more expressive than those without, we might endeavor to use them wherever we can still maintain polynomial time computability. The following generalization of the above two results, which is also proved in Section 12, enables us to use omittable nodes on the outer face of a planar graph.

Theorem 3.3. *Let Y be any finite subset of $\mathbb{C}$. Suppose that each element of Y can be computed to n decimal places i.e. absolute error less than 2^{-n}, in time polynomial in n. Let $\{G_n \mid n \geq 1\}$ be a family of planar embeddings of planar graphs on n nodes with all omittable nodes on the outer face, with polynomial time identifiable weights from Y, and having an integer value of MatchSum(G_n). Then MatchSum(G_n) can be computed in polynomial time, and, in fact, in NC2.*

We note that while we emphasize the case of the field $F = \mathbb{C}$, the whole development applies equally, and without the need for these numerical considerations, if F is a finite field. In that case the consequences are for $\#_k\mathrm{P}$, the counting class corresponding to #P, but modulo k [Valiant, 1979a].

We also note that for planar structures there exist algorithms that can perform elimination on $n \times n$ matrices in $O(n^{1.5})$ rather than $O(n^3)$ steps [Lipton, Rose, and Tarjan, 1979].

4 Matchgrids and Planar Matchgates

Our overall strategy is the following. We transform an instance I of a counting problem, such as #X-MATCHINGS, to an instance Ω of what we call a matchgrid, such that the weighted sum of the perfect matchings of Ω will equal the number of solutions of I. The structure of I is reflected in the structure of Ω, with the individual components of I, nodes and edges in the case of #X-MATCHINGS, each replaced by gadgets that we call matchgates. The weight of the perfect matchings in each matchgate will equal the number of solution fragments of the #X-MATCHING problem.

We now introduce the basic concepts of the theory. We note that while our starting point is the notion of a matchgate, exactly as in [Valiant 2002a], that earlier work employed classical rather than holographic reductions. This paper can be read independently of that earlier one. However, we have attempted to keep our notation consistent with it, and reference it occasionally.

A planar matchgate Γ is a triple (G, X, Y) where G is a planar embedding of a planar graph (V, E, W) where $X \subseteq V$ is a set of *input* nodes, $Y \subseteq V$ is a set of *output* nodes, and where X, Y are disjoint. Further, as one proceeds anticlockwise around the outer face starting from one point one encounters first the input nodes labeled $1, 2, \cdots, \mid X \mid$ and then the output nodes $\mid Y \mid, \cdots, 2, 1$, in that order. The *arity* of the matchgate is $|X| + |Y|$. For $Z \subseteq X \cup Y$ we define the *standard signature of* Γ *with respect to* Z to be PerfMatch$(G - Z)$, where $G - Z$ is the graph obtained by removing from G the node set Z and all edges that are incident to Z. Further we define the *standard signature* of Γ to be the $2^{|X|} \times 2^{|Y|}$ matrix $u(\Gamma)$ whose elements are the standard signatures of Γ with respect to Z for the $2^{|X|}2^{|Y|}$ choices of Z. The labeling of the matrix is as follows: Suppose that X and Y have the labeling described, i.e., the nodes are labeled $1, 2, \ldots, |X|$ and $|Y|, \ldots, 2, 1$ in anti-clockwise order. Then each choice of Z corresponds to a subset from each of these labeled sets. If each node present in Z is regarded as a 1, and each node absent as a 0, then we have two binary strings of length $|X|$, $|Y|$, respectively, where the nodes labelled 1 correspond to the leftmost binary bit. Suppose that i, j are the numbers represented by these strings in binary. Then the entry corresponding to Z will be the one in row i and column j in the signature matrix $u(\Gamma)$.

We note that in [Valiant, 2002a] matchgates were defined in a general, not necessarily planar, setting. In that more general case, when we compose matchgates into matchcircuits we need to keep track of sign influences explicitly, since we cannot rely on the Fisher-Kasteleyn-Temperley method. For that reason we use there the more complex notion of character, while the simpler notion of signature suffices in this paper since we restrict ourselves to planar graphs.

The treatment in [Valiant, 2002a] is more general also in the second respect that omitted nodes are allowed in matchgates and character is defined in terms of MatchSum rather than PerfMatch. We accommodate this generalization in two limited ways in the current paper. We use them in a thought experiment in the proof of Theorem 4.1. We also use them explicitly in circuits, as in Theorem 9.6, as allowed by Theorem 3.3 and Corollary 4.2, noting that the omittable nodes have to be on the outer face of the final circuit.

A *basis* of size k is a set of distinct nonzero vectors each of length 2^k with entries from a field F. Often we will have just two basis vectors that represent 0 and 1, respectively, and in that case we shall call them n and p. In this paper all bases will be of size $k = 1$, so that $n = (n_0, n_1)$ and $p = (p_0, p_1)$. The basis $\mathbf{b0} = [n, p] = [(1, 0), (0, 1)]$ we call the *standard basis*. In general, the vectors in a basis do not need to be independent

In this section we shall use as an illustrative example the basis $\mathbf{b1} = [n, p] = [(-1, 1), (1, 0)]$. The gates we describe will be used in Section 8 to implement our first holographic algorithm, one for the #X-MATCHINGS problem. We believe that this basis, though apparently somewhat specialized, is an instructive example. In later sections we shall describe bases, such as **b2**, which appear to be more broadly applicable.

In general if we have two vectors q, r, of length l, m, respectively, then we shall denote the *tensor product* $s = q \otimes r$ to be the vector s of length lm in which $s_{im+j} = q_i r_j$ for $0 \leq i < l$ and $0 \leq j < m$. Thus, for example, for the basis **b1**, $n \otimes p = (-1, 0, 1, 0)$. Clearly $\otimes$ is associative.

We say that a matchgate is a *generator* if it has zero input nodes and nonzero output nodes, and a *recognizer* if it has zero output nodes and nonzero input nodes. One can define equally naturally a *transducer* gate that has both nonzero inputs and nonzero outputs, but we do not use these for our examples. From the definition of signature it follows that for generators and recognizers the

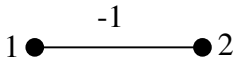

Figure 1: A generator matchgate for basis **b1** with output nodes $\{1, 2\}$ and one edge of weight -1. It generates $n \otimes n + n \otimes p + p \otimes n$.

signature is a vector.

Here we shall introduce generators and recognizers by example. A more formal treatment can be found at the beginning of Section 5. Intuitively, a generator can be viewed as emitting n and p particles along its outputs in all possible combinations, each combination with a certain value. A recognizer will absorb combinations of these particles entering via its inputs, again attaching a certain value to each combination. The overall goal is that the sum over all patterns of particles that can be generated of the product of the values of all the generators and recognizers be equal to the value of the function being computed.

We first consider generators. Suppose that a generator has graph G and m output nodes. Then, by definition, its standard signature will be a 2^m-vector. Recall that element j in this vector is the value of PerfMatch(G') where G' is G but with those output nodes removed that correspond to the index j in the manner described in the definition of standard signature. Consider the generator matchgate Γ shown in Figure 1.

It has $V = \{1, 2\}$, $E = \{(1, 2)\}$, $W(1, 2) = -1$, and the input node set $X = \emptyset$, and output node set $Y = \{1, 2\}$. Then the standard signature $u(\Gamma)$ of Γ is the vector $(-1, 0, 0, 1)$ since if both output nodes are removed then PerfMatch(G') $= 1$, if neither is removed then PerfMatch(G') $= -1$, and if exactly one is removed then there is no perfect matching and PerfMatch(G') $= 0$. Now for the basis **b1** defined above it is easy to see that $n \otimes n = (1, -1, -1, 1)$, $n \otimes p = (-1, 0, 1, 0)$ and $p \otimes n = (-1, 1, 0, 0)$. The sum of these is $(-1, 0, 0, 1)$, which happens to equal the above stated standard signature of the matchgate. (Note that here we used the convention that PerfMatch of a graph with no nodes is 1. This can be avoided by using as the generator a chain of four nodes, rather than two, and again having the end nodes as output nodes.) Hence we conclude that for this gate and basis **b1** the following holds.

Proposition 4.1. *There exists a generator matchgate Γ with $u(\Gamma) = n \otimes n + n \otimes p + p \otimes n$, where (n, p) is the basis* **b1**. ■

In other words this gate generates the linear sum of the three bit combinations 00, 01, and 10 when interpreted in the basis **b1** representation. The *signature of this generator with respect to the basis* **b1**, (a notion further elaborated in Section 5, as relation 5.1) will then be (1, 1, 1, 0) since these are the coefficients of the contributions for the four bit patterns 00, 01, 10, 11, respectively. For $x \in \{n, p\}^2$ we shall denote by $valG(\Gamma, x)$ the signature element corresponding to x. Thus, for the current example, valG($\Gamma, n \otimes p$) $= 1$ and valG($\Gamma, p \otimes p$) $= 0$. (We note that since a basis **b** can be an arbitrary set the signature of a generator with respect to **b** may not be unique. When we discuss a signature any valid signature will do.)

We shall now go on to discuss recognizers. Let us suppose that these have m inputs. The purpose of such recognizers is to have PerfMatch take on appropriate values as the inputs range over the 2^m possible tensor product values $x = x_1 \otimes \ldots \otimes x_m$ where each x_i ranges independently over $\{n, p\}$. Note that x can be viewed as a 2^m vector in the standard basis. The value of PerfMatch for the recognizer matchgate Γ "evaluated at input" x will by denoted by $valR(\Gamma, x)$. More precisely,

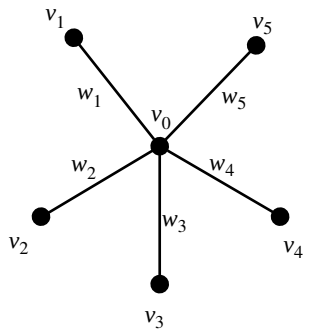

Figure 2: A recognizer matchgate for basis **b1** with input nodes v_1, $v_2, \ldots, v_5$, and edge weights w_1, $w_2, \ldots, w_5$.

if vector u is the standard signature of Γ, and x is the 2^m-vector representing x in the standard basis, then valR(Γ, x) is the inner product $u\ x$. Consider the family of recognizers Γ_k shown in Figure 2. They are defined by the star graph $G_k = (V_k, E_k, W_k)$, where $V_k = \{v_0, v_1, \ldots, v_k\}$, $E_k = \{(v_0, v_i) | 1 \leq i \leq k\}$, the input nodes are $\{v_i | 1 \leq i \leq k\}$, and the weight of edge (v_0, v_i) is w_i.

Proposition 4.2. *For all $k > 0$ and for all $w_1, \ldots, w_k \in F$ there exists a k-input recognizer matchgate Γ such that on input $x = x_1 \otimes \ldots \otimes x_k \in \{n, p\}^k$ over basis* **b1** valR(Γ, x) *equals:*

(i) $-(w_1 + \ldots + w_k)$ *if* $x_1 = \ldots = x_k = n$,
(ii) w_i *if* $x_i = p$, *and* $x_j = n$ *for every* $j \neq i$,
(iii) 0 *for the remaining* $2^k - k - 1$ *values of* $x_1, \ldots, x_k$.

Proof. We shall prove that the gate of Figure 2, where the weight of edge (v_0, v_i) is set to w_i, is such a recognizer. To see this note that the only subsets Z of the input nodes $\{v_i | 1 \leq i \leq k\}$ that can be removed that allow the PerfMatch of the remaining graph to be nonzero are those that contain exactly $k - 1$ elements, and for these PerfMatch $= w_i$ if v_i is the node omitted from Z. Hence if two or more of the inputs are $p = (1, 0)$ then PerfMatch is zero. If exactly one input is p, and this is applied at node v_i, and all the others are n, then the only nonzero contribution comes from the node v_i being omitted from Z, and this gives a contribution of $p_0 n_1^{k-1} w_i = w_i$. If all the inputs are n then there is a nonzero contribution $n_0 n_1^{k-1} w_i = -w_i$ for each possible v_i, and then the total value of PerfMatch is $-(w_1 + \ldots + w_k)$. ■

We note that the basic properties of a basis are unchanged if the first and second components of all of its elements are interchanged together, or if they are multiplied by arbitrary constants x and y, respectively. The former transformation can be realized by appending to every input or output node an edge of weight 1. The latter can be realized by appending to such nodes chains of length two weighted by x and y, respectively. Hence any basis $[(-x, y), (x, 0)]$ or $[(x, -y), (0, y)]$ with nonzero x and y is essentially equivalent to **b1**.

Proposition 4.3 *If there is a generator (recognizer) with certain valG(valR) values for size one basis $\{(a_1, b_1) \cdots, (a_r, b_r)\}$ then there is a generator (recognizer) with the same valG(valR) values for any basis $\{(xa_1, yb_1), \cdots, (xa_r, yb_r)\}$ or $\{(xb_1, ya_1), \cdots (xb_r, ya_r)\}$ for any $x, y \in F$.*

We define a *matchgrid* over a basis **b** to be a weighted undirected planar graph G that consists of the disjoint union of a set of g generator matchgates $B_1, \ldots, B_g$, r recognizer matchgates $A_1, \ldots, A_r$, and f *connecting edges* $C_1, \ldots, C_f$ where each C_i edge has weight one and joins an output node in

a generator matchgate with an input node of a recognizer matchgate, such that every input and output node in every constituent matchgate has exactly one such incident connecting edge.

Consider such a matchgrid $\Omega = (A, B, C)$ and assume, for simplicity, that the basis is of size two. We denote by $X = \mathbf{b}^f = (n, p)^f$ the set of 2^f possible combinations of the basis elements n, p that can be transmitted simultaneously along the f connecting edges in the matchgrid. We can break X into $X_1 \otimes \ldots \otimes X_g$ where $X_j = (n, p)^{k(j)}$ and $k(j)$ is the arity of generator B_j and refers to the connecting edges that are incident to that generator. Also if $x \in X$ then we can mirror this decomposition as $x = x_1 \otimes \ldots \otimes x_g$ where x_j is the particular set of basis elements that is transmitted from the outputs of B_j. We can also break the same X into $\bar{X}_1 \otimes \cdots \otimes \bar{X}_r$ where $\bar{X}_j = (n, p)^{l(j)}$ and $l(j)$ is the arity of the recognizer A_j and refers to the connecting edges incident to that recognizer. If $x \in X$ then this decomposition can be mirrored as $x = \bar{x}_1 \otimes \cdots \otimes \bar{x}_r$ where $\bar{x_j}$ is the set of basis elements transmitted into the inputs of A_j.

Now for each $x \in X$ each recognizer A_i will evaluate a value $\mathrm{valR}(A_i, x) = \mathrm{valR}(A_i, \bar{x}_i)$ and each generator B_j will generate the value $\mathrm{valG}(B_j, x) = \mathrm{valG}(B_j, x_j)$. The product of these values for all the generators and all the recognizers is the *value of the matchgrid* at x. The *value of the matchgrid* will be the sum of these products for the various x. This quantity we call the *Holant*:

$$\mathrm{Holant}(\Omega) = \sum_{x \in \mathbf{b}^f} \left[\prod_{1 \le j \le g} \mathrm{valG}(B_j, x_j) \right] \left[\prod_{1 \le i \le r} \mathrm{valR}(A_i, x) \right].$$

There are two views of a matchgrid, one as a directed weighted graph G and the other as a composition $\Omega = (A, B, C)$ of matchgates and connecting edges. For the former we have already defined various matching polynomials such as PerfMatch and it is these that we shall evaluate in polynomial time. For the latter it is the Holant that expresses the basic intention of the matchgrid, that of performing a weighted sum of potentially exponentially many solutions, indexed by the set X, that obey the local constraints expressed in the matchgates.

The central relationship that is necessary for a holographic algorithm is that the potentially exponential summation that the Holant defines be computable in polynomial time. The following is a paradigmatic expression of this. The reader should note that for the standard basic *valG* = *valR*, and the Theorem follows immediately. More surprising, and at the heart of our holographic technique, is the fact is that the result holds for *all* bases:

Theorem 4.1. *For any matchgrid* Ω *over any basis* $\mathbf{b}$ *if* Ω *has weighted graph* G *then*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G).$$

Proof. The result is a consequence of linearity. The following is a mechanistic way of presenting the argument.

Suppose for the sake of this proof that we allow a certain subset of the nodes of a matchgate to be "omittable with weight 1" in the sense that its signature will be defined by not just perfect matchings but also by all other matchings that saturate all the nonomittable nodes, but any omittable node may or may not be saturated. In other words we are using the polynomial MatchSum with $\lambda_i = 1$ for the omittable nodes, and $\lambda_i = 0$ for the unomittable nodes. Once we allow omittable nodes we have matchgates for any single basis elements such as p and n: Figure 3 shows a matchgate with omittable node 1 and output node 3. The standard signature is clearly (w_0, w_1) since if node 3 is not in Z then the only allowed matching is the edge (2,3) with weight w_0 and if node 3 is in Z then

the only allowed matching is edge (1,2) with weight w_1. Hence we get a matchgate with standard signature $p = (p_0, p_1)$ by fixing $w_0 = p_0$ and $w_1 = p_1$, and one with standard signature $n = (n_0, n_1)$ by fixing $w_0 = n_0$ and $w_1 = n_1$.

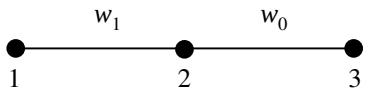

Figure 3: A generator matchgate having node 1 as an omittable node and node 3 as the output node. It has standard signature (w_0, w_1)

Suppose we pick a *fixed* element $x \in X$ from among the $|\mathbf{b}|^f$ that are potentially generated, and regard it as the tensor product $x_1 \otimes \ldots \otimes x_g$ where x_i corresponds to the basis elements that are involved in generator B_i, and equivalently as a tensor product $\bar{x}_1 \otimes \cdots \otimes \bar{x}_r$ where $\bar{x}_j$ corresponds to the basis elements that are involved in recognizer A_j. Then we can construct from Ω a matchgrid $G(x)$ that replaces each generator B_i having k outputs by k generators of the single basis elements specified by x_i for those k outputs. Further, for each such B_i the parameters in one of these single basis element generators will be set so as to multiply its value by $\mathrm{valG}(B_i, x_i)$ so that these generators for B_i generate x_i with that multiplier $\mathrm{valG}(B_i, x_i)$. Then it follows from the definitions of generators, recognizers, and the way they are assembled according to the definition of matchgrids that:

$$\mathrm{MatchSum}(G(x)) = \left[\prod_{1 \le i \le g} \mathrm{valG}(B_i, x_i)\right] \left[\prod_{1 \le i \le r} \mathrm{valR}(A_i, x)\right].$$

The reason for this equality is that a fixed vector $x \in X$ is being generated with weight $\Pi \mathrm{valG}(B_i, x)$, where the multiplication is over all the generators B_i. The inner product of this x with the standard signature u of each of the recognizers gives the contribution to MatchSum of the recognizers. But, by definition, the inner product ux equals $\mathrm{valR}(A_i, x)$ for each recognizer A_i,

Now partition X into equivalence classes of $|X_1|$ elements each so that all members of each equivalence class have identical $X_2, \ldots, X_g$ components. For each of these equivalence classes, say the one defined by $x_2 \in X_2, \ldots, x_g \in X_g$, define the matchgrid $G(x_2, \ldots, x_g)$ as follows: Set x to have components $x_2, \ldots, x_g$ and any $x_1 \in X_1$ and let $G(x_2, \ldots, x_g)$ be $G(x)$ but with the single element generators for x_1 replaced by the generator B_1, which generates the sum of all members of X_1, each with the appropriate weight. Then clearly, summing over all the values of x_1, gives

$$\mathrm{MatchSum}(G(x_2, \ldots, x_g)) = \sum_{x_1 \in X_1} \mathrm{valG}(B_1, x_1) \left[\prod_{2 \le i \le g} \mathrm{valG}(B_i, x_i)\right] \left[\prod_{1 \le i \le r} \mathrm{valR}(A_i, x)\right]$$

where x in the last term denotes $x_1 \otimes x_2 \otimes \cdots \otimes x_g$.

We iterate this process for $B_2, \ldots, B_g$, in turn. For example, for B_2 we partition $X_2 \otimes X_3 \otimes \cdots \otimes X_g$ into equivalence classes of $|X_2|$ elements each so that each class has identical $x_3, \cdots, x_g$ components. For each of these equivalence classes, say that defined by $x_3 \in X_3, \cdots, x_g \in X_g$, we define matchgrid $G(x_3, \cdots, x_g)$ to be $G(x_2, \cdots, x_g)$ for some x_2, but with the single element generators for x_2 replaced by the generator B_2. This will sum all the members of X_2 with the appropriate weights. It then follows that

$$\text{MatchSum}(G(x_3, \cdots, x_g)) = \sum_{x_1 \in X_1} \sum_{x_2 \in X_2} \text{valG}(B_1, x_1)\text{valG}(B_2, x_2) \left[\prod_{3 \le i \le g} \text{valG}(B_i, x_i) \right] \left[\prod_{1 \le i \le r} \text{valR}(A_i, x) \right]$$

After the last stage we have replaced all the generators of single basis elements and have just one matchgrid left, which is $G() = \Omega$. It follows then from the definition of the Holant that MatchSum(G) equals Holant(G). Note that at that point all the single element generators with omittable nodes have been replaced by the original generators with no omittable nodes, and hence the result also holds for PerfMatch(G) as claimed. ■

We use the Holant Theorem to express the intention of holographic reductions. A counting problem *#F has a simple holographic reduction* to planar PerfMatch if there is a transformation that (i) produces all edge weights from a fixed set Y in which each element can be computed to absolute error less than 2^{-n} in time polynomial in n, (ii) produces a weighted graph with the Holant and therefore also PerfMatch equal to $\#F$, and (iii) is computable in NC2.

Corollary 4.1 *If #F has a simple holographic reduction to planar PerfMatch then #F ∈ NC2.*

Proof. The instance of #F is first transformed to an instance of planar PerfMatch. By Theorem 3.1 the required solution is given by the square root of the determinant of a matrix that satisfies the conditions of Theorem 3.2. It then follows from Corollary 3.2.1 given in Section 12 that this determinant and the required solution can be computed in NC2. ■

All the reductions we exhibit in this paper are simple holographic reductions, in which every element of Y is either rational, or an algebraic number with an explicitly given polynomial equation. Hence, a solution can be found accurate to 2^{-n} in time polynomial in n, as required (e.g. Pan, 1997).

A direct application of the above result that uses the matchgates already described for the nonstandard basis **b1** is Theorem 8.1. The reader may choose to look at that Section 8 next before proceeding to other sections.

The previous theorem also supports the following generalization.

Corollary 4.2. *For any matchgrid Ω with omittable nodes and having weighted graph G, if in the definition of signature of a matchgate PerfMatch is replaced by MatchSum, and this is inherited in the definitions of valG and valR, then:*

$$\text{Holant}(\Omega) = \text{MatchSum}(G).$$

Note, however, that the only case we know in which this can be exploited for polynomial time algorithms is when all the omittable nodes are on the outer face and we can invoke Theorem 3.3, as we do in Theorem 9.6.

5 Signatures of Planar Matchgates

In this section we shall give a more systematic treatment of generators and recognizers.

Consider a graph G with three external nodes numbered 1, 2, 3. For each choice of $i, j, k \in \{0, 1\}$ let u_{ijk} equal the PerfMatch polynomial of G when nodes 1, 2, 3 are deleted, respectively, according

to whether i, j, k equal 1 or not. Thus u_{111} denotes PerfMatch(G') where G' is G with all three external nodes removed. Note that for a generator or recognizer the definition of the standard signature u given in Section 4 implies that u equals the 8-vector $(u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111})$.

For a given basis $\mathbf{b}$ we denote by $\{\mathbf{b}_{ijk} | i, j, k \in \{0,1\}\}$ the eight possible external 8-vectors $x_1 \otimes x_2 \otimes x_3$ where x_1, x_2, x_3 range over $\{n, p\}$, respectively. Thus $\mathbf{b}_{010} = n \otimes p \otimes n$ will denote the basis vector n at inputs 1 and 3, and the basis vector p at input 2. The (r, s, t)-element of the 8-vector $\mathbf{b}_{ijk}$ will be denoted by $(\mathbf{b}_{ijk})_{rst}$ and will represent in $x_1 \otimes x_2 \otimes x_3$ the product of the r-th component of x_1, the s-th component of x_2, and the t-th component of x_3, for $r, s, t \in \{0, 1\}$. Thus $(\mathbf{b}_{010})_{110}$ will equal $n_1 p_1 n_0$, for example.

For the special case of the standard basis $n = (1, 0), p = (0, 1)$ clearly the (r, s, t)-element of vector $\mathbf{b}_{ijk}$ will equal 0 unless $r = i$, $s = j$, and $t = k$, in which case it will equal 1.

Let us first consider generators. Suppose that G has standard signature u, and for all $\{i, j, k\} \in \{0, 1\}^3$

$$u_{ijk} = \sum q_{rst} (\mathbf{b}_{rst})_{ijk} \tag{5.1}$$

for some vector of numbers q where summation is over all $\{r, s, t\} \in \{0, 1\}^3$. Then we say that G *generates signature q with respect to basis* $\mathbf{b}$. Note that if G has no omittable nodes then it is either even or odd and hence either the even or the odd four elements of $\{0, 1\}^3$ have zero values for u_{ijk}.

Let us now consider recognizers. Suppose that G has standard signature $\hat{u}$, and that when the 8-vector $\mathbf{b}_{ijk}$ for some $\{i, j, k\} \in \{0, 1\}^3$ is input to G then G evaluates to $\hat{q}_{ijk}$. Then

$$\hat{q}_{ijk} = \sum \hat{u}_{rst} (\mathbf{b}_{ijk})_{rst} \tag{5.2}$$

must hold, where summation is over $\{r, s, t\} \in \{0, 1\}^3$. We then say that G *recognizes signature $\hat{q}$ over basis* $\mathbf{b}$. Again, if G has no omittable nodes then it is either even or odd and hence either the even or the odd four elements of $\{0, 1\}^3$ have zero values for $\hat{u}_{rst}$.

Proposition 5.1. *A gate G with standard signature equal to u will generate and recognize u with respect to the standard basis.*

Proof. This is immediate from the definition of generators and recognizers, and the fact observed above that for the standard basis $(\mathbf{b}_{ijk})_{rst} = \delta_{ir} \delta_{js} \delta_{kt}$ where δ is the Dirac delta function. ■

For any basis $\mathbf{b}$ and matchgate, whether a generator or recognizer, one can define the signature of the matchgate with respect to the basis to be the vector q that it generates according to relation (5.1), or the vector q that it recognizes according to relation (5.2) above. Thus if the matchgate has arity m then its signature with respect to $\mathbf{b}$ is a vector of length 2^m. We will denote it, for the $m = 3$ case, typically by $(q_{000}, q_{001}, \ldots, q_{111})$. The standard signature defined in Section 3 is just the signature with respect to the standard basis. When we discuss a basis we need to be clear about which basis is involved. However, signatures that differ from each other by a nonzero constant factor can be treated as equivalent since their contribution to the PerfMatch or MatchSum polynomials of any overall matchgrid differ by just that constant multiple.

If the arity m gate is symmetric in its inputs and outputs then we can define its *symmetric signature with respect to basis* $\mathbf{b}$ to be the vector $[S_0, S_1, \ldots, S_m]$ where S_i is equal to all the elements of the ordinary signature that are indexed by $\{0, 1\}^m$ patterns with i occurrences of 1. For example, the gate in Figure 1 is symmetric. With respect to basis **b1** it has ordinary signature

$(q_{00}, q_{01}, q_{10}, q_{11}) = (1, 1, 1, 0)$ and symmetric signature $[S_0, S_1, S_2] = [1, 1, 0]$. The gate in Figure 2 has symmetric instances, such as those where all the weights $w_i = 1$ and $m = 3$, say, in which case the symmetric signature is $[-3, 1, 0, 0]$ with respect to the same basis. We shall use round parentheses for signatures, and square parentheses for the abbreviated symmetric version.

6 Realizable Signatures for the Standard Basis

With respect to the standard basis we can characterize the standard signatures that are realizable with planar matchgates of arity up to four. We first note that in any such signature either the odd or the even components must be zero depending on the parity of the number of nodes in the matchgate. Propositions 6.1 and 6.2 therefore show that for arity 2 and 3 all signatures are realizable up to this basic constraint.

Proposition 6.1. *For all F and all $x, y \in F$ there exist matchgates with arity 2 and standard signatures $(u_{00}, u_{01}, u_{10}, u_{11}) = (x, 0, 0, y)$ and $(0, x, y, 0)$.*

Proof. The matchgates of Figure 4, with external nodes $\{1, 2\}$, suffice. ■

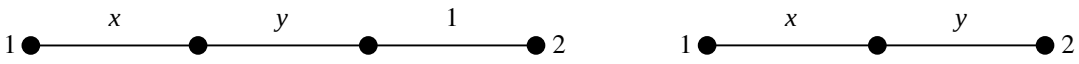

Figure 4: Two arity two gates with input/output gates $\{1, 2\}$.

Proposition 6.2. *For all F and all $x, y, z, t \in F$ there exist matchgates with arity 3 and standard signatures $(u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111}) = (t, 0, 0, z, 0, y, x, 0)$ and $(0, x, y, 0, z, 0, 0, t)$.*

Proof. For the odd case $(0, x, y, 0, z, 0, 0, t)$ consider Figure 5. Clearly if $t \neq 0$ the left-hand figure has standard signature $(0, x, y, 0, z, 0, 0, t)$ and solves the problem. If $t = 0$ we use the right-hand diagram.

For the even case $(t, 0, 0, z, 0, y, x, 0)$ the signature of the left part of Figure 6 is $(ax + by + cz, 0, 0, z, 0, y, x, 0)$ and therefore solves the problem for all values of x, y, z, t by appropriate choice of a, b, c, unless $x = y = z = 0$ and $t \neq 0$. In that exceptional case we use the right-hand diagram. ■

In general we shall refer to the elements of a signature being *even* or *odd* according to whether their index has an even or odd number of 1's. Thus, for example, u_{1010} and u_{0000} are even while u_{0100} and u_{0111} are odd.

Proposition 6.3. *Suppose the elements of the standard signature are represented by u_{ijkl} for $i, j, k, l \in \{0, 1\}$. For any F it is possible to realize by matchgates with arity 4 any standard signature such that*

(i) $u_{0000}u_{1111} - u_{0011}u_{1100} + u_{0101}u_{1010} - u_{0110}u_{1001} = 0$, $u_{1111} \neq 0$, *and all the odd elements are zero, or*

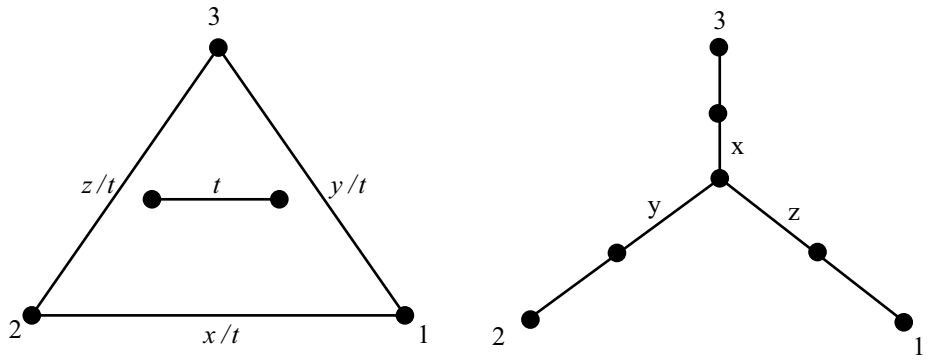

Figure 5: Arity three gates for nonzero odd components.

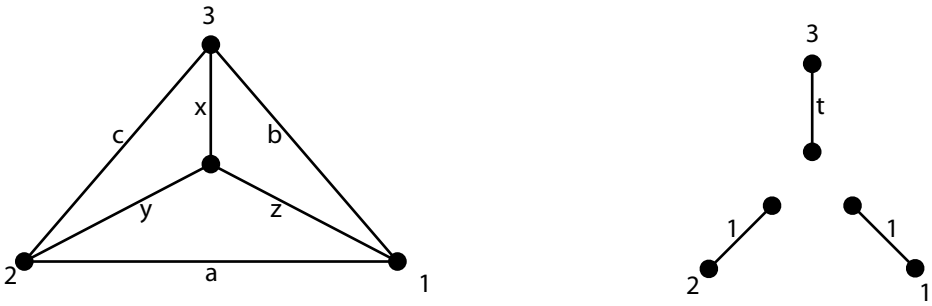

Figure 6: Arity three gates for nonzero even components.

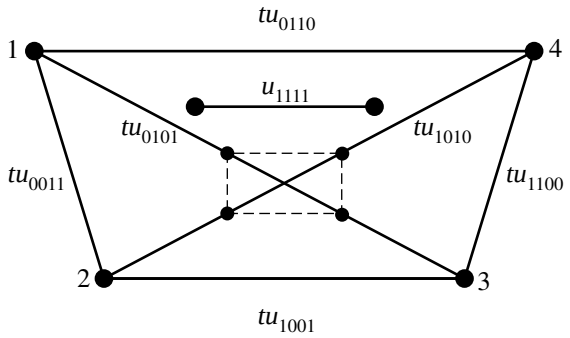

Figure 7: An arity four matchgate, where $t = 1/u_{1111}$

(ii) $u_{1000}u_{0111} - u_{1011}u_{0100} + u_{1101}u_{0010} - u_{1110}u_{0001} = 0$, $u_{0111} \neq 0$, *and all the even elements are zero.*

Proof. We consider (i) first. The algebraic relationship is the first matchgate identity from [Valiant, 2002a] and we follow the construction from there shown in Figure 7. First, ignoring the central square we have a nonplanar matchgate. It is easy to see that this matchgate does have the desired values for the seven components $u_{1111}, u_{0110}, u_{1001}, u_{0011}, u_{1100}, u_{0101}$, and u_{1010} of the signature. Now if we could somehow simulate the crossing edges $(1, 3)$ and $(2, 4)$ by a planar graph so as to create a change in sign the value of the eighth component u_{0000} for this matchgate would be the following:

$$tu_{0110}u_{1001} + tu_{0011}u_{1100} - tu_{0101}u_{1010}.$$

If we substitute $t = 1/u_{1111}$ then we would have the claimed relationship (i). Now to make the graph planar and to simulate the -1 factor, we replace the crossing edges by the planar graph shown in Figure 8. In Figure 8 if we substitute $a = 1$, $b = i$, $c = d = -1/2$, and $e = \sqrt{i}$, where $i^2 = -1$, then nonzero contributions from PerfMatch occur for just the four combinations of each of (1, 3) and (2, 4) being present or not in Figure 7. Each combination comes with a factor of $+1$, except the one that has both crossing edges present in Figure 7, which contributes a factor of -1 as required. This concludes the construction for the field of complex numbers. (It can also be verified that the same graph with appropriate $\pm$ 1 weights will have factors -1, 2, 2 and 4, which can be normalized to -1, 1, 1 and 1 respectively by appending appropriate graphs at the external nodes. Hence the construction applies for all fields F. Note that the signature of any planar matchgate has to satisfy algebraic identities similar to those of the character [Valiant 2002b].)

In order to obtain part (ii) we simply append an extra edge weighted 1 at input 1, and call the other endpoint of the new edge the new input node 1. This transformation leaves the elements of the signature unchanged, except that they are renamed by the process of flipping the first bit of the index in each term e.g., u_{0000} becomes u_{1000}. ■

We note that the constraints $u_{1111} \neq 0$ and $u_{0111} \neq 0$ can be eliminated in the following sense. If any of the sixteen components is nonzero then, by the method of the last paragraph one can flip bits so that the nonzero entry is moved to the 1111 or 0111 position, and the relation (i) or (ii) holds for the corresponding renaming of the elements.

Proposition 6.4. *For all F and any arity m and any $S_0 \in F$ there is a matchgate with standard*

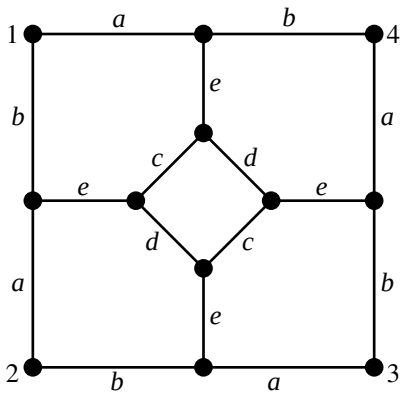

Figure 8: An arity four planar matchgate that is used to simulate the crossover in Figure 7. The substitution $a = 1$, $b = i$, $c = d = -1/2$, $e = \sqrt{i}$ suffices where $i^2 = -1$.

symmetric signature $[S_0, \ldots, S_m]$ *where* $S_1 = \ldots = S_m = 0$.

Proof. The gate consists of $2m$ nodes $v_1, \ldots, v_m$, $u_1, \ldots, u_m$ and m edges (v_i, u_i) where $u_1, \ldots, u_m$ are the output nodes. All the edges have weight one, except for one which has weight S_0. ■

7 Realizable Signatures for Arity Two Matchgates

Relations (5.1) and (5.2) in Section 5 relate the signatures realizable by an arbitrary basis to those realizable by the standard basis. In Section 6 we characterized the signatures that are realizable by the standard basis for gates of arity up to four. In this section we shall spell out some consequences for signatures with respect to arbitrary bases, that are realizable by gates of arity 2. These gates will be invoked in several places in the various algorithms described in later sections.

For ease of notation we shall consider the basis to be $\mathbf{b} = [(a, b), (c, d)]$ so that $\mathbf{b}_{00} = (a, b) \otimes (a, b)$, $\mathbf{b}_{01} = (a, b) \otimes (c, d)$, $\mathbf{b}_{10} = (c, d) \otimes (a, b)$, and $\mathbf{b}_{11} = (c, d) \otimes (c, d)$.

Relation (5.1) then describes the following requirements on a generator to have signature $(q_{00}, q_{01}, q_{10}, q_{11})$ with respect to $\mathbf{b}$:

$u_{00} = a^2 q_{00} + acq_{01} + acq_{10} + c^2 q_{11}$,
$u_{01} = abq_{00} + adq_{01} + bcq_{10} + cdq_{11}$,
$u_{10} = abq_{00} + bcq_{01} + adq_{10} + cdq_{11}$, and
$u_{11} = b^2 q_{00} + bdq_{01} + bdq_{10} + d^2 q_{11}$.

By Proposition 6.1 any standard signature is possible as long as either $u_{00} = u_{11} = 0$ or $u_{01} = u_{10} = 0$. Hence there exist generators with signature $(q_{00}, q_{01}, q_{10}, q_{11})$ with respect to basis $\mathbf{b}$ if either

$$a^2 q_{00} + acq_{01} + acq_{10} + c^2 q_{11} = 0 \text{ and } b^2 q_{00} + bdq_{01} + bdq_{10} + d^2 q_{11} = 0,$$

or

$$abq_{00} + adq_{01} + bcq_{10} + cdq_{11} = 0 \text{ and } abq_{00} + bcq_{01} + adq_{10} + cdq_{11} = 0.$$

Proposition 7.1. *For the basis* $\mathbf{b2} = [(1,1),(1,-1)]$ *for any* $x, y \in F$ *there is a generator for* $(x,y,y,x) = [x,y,x]$.

Proof. The second of the two cases above gives $q_{00} - q_{01} + q_{10} - q_{11} = 0$ and $q_{00} + q_{01} - q_{10} - q_{11} = 0$. Clearly these will be satisfied if $q_{00} = q_{11}$ and $q_{01} = q_{10}$ ■

Proposition 7.2. *For the basis* $\mathbf{b2} = [(1,1),(1,-1)]$ *for any* $x, y \in F$ *there is a generator for* $(x,y,-y,-x)$.

Proof. The first of the two cases above gives $q_{00} + q_{01} + q_{10} + q_{11} = 0$ and $q_{00} - q_{01} - q_{10} + q_{11} = 0$. Clearly these will be satisfied if $q_{00} + q_{11} = 0$, and $q_{01} + q_{10} = 0$. ■

Moving on to recognizers, we note that the requirements for a recognizer to have signature $(q_{00}, q_{01}, q_{10}, q_{11})$ are given by relation (5.2):

$q_{00} = a^2u_{00} + abu_{01} + abu_{10} + b^2u_{11}$,
$q_{01} = acu_{00} + adu_{01} + bcu_{10} + bdu_{11}$,
$q_{10} = acu_{00} + bcu_{01} + adu_{10} + bdu_{11}$, and
$q_{11} = c^2u_{00} + cdu_{01} + cdu_{10} + d^2u_{11}$.

By Proposition 6.1 any standard signature is possible as long as either $u_{00} = u_{11} = 0$ or $u_{01} = u_{10} = 0$. Hence there exist recognizers with signature $(q_{00}, q_{01}, q_{10}, q_{11})$ with respect to basis $\mathbf{b}$ of the two forms:

$(abu_{01} + abu_{10},\ adu_{01} + bcu_{10},\ bcu_{01} + adu_{10},\ cdu_{01} + cdu_{10})$,

and

$(a^2u_{00} + b^2u_{11},\ acu_{00} + bdu_{11},\ acu_{00} + bdu_{11},\ c^2u_{00} + d^2u_{11})$.

Proposition 7.3. *If* $\mathbf{b2} = [(1,1),(1,-1)]$ *is a basis for field* F *then for any* $x, y \in F$ *there is a recognizer for* $(x,y,y,x) = [x,y,x]$.

Proof. The second case above gives signature $(u_{00} + u_{11}, u_{00} - u_{11}, u_{00} - u_{11}, u_{00} + u_{11})$. ■

Proposition 7.4. *If* $[(a,b),(c,d)]$ *is a basis then there is a recognizer for* $(0, ad - bc, bc - ad, 0)$.

Proof. Follows from the first case above if $u_{01} = 1$ and $u_{10} = -1$. ■

Proposition 7.5. *If* $[(a,b),(c,d)]$ *is a basis then there is a recognizer for* $(a^2 + b^2, ac + bd, ac + bd, c^2 + d^2)$.

Proof. Follows from the second case with $u_{00} = 1$ and $u_{11} = 1$. ■

Proposition 7.6. *If* $[(a,b),(c,d)]$ *is a basis then there is a recognizer for* $(a^2 - b^2, ac - bd, ac - bd, c^2 - d^2)$.

Proof. Follows from the second case with $u_{00} = 1$ and $u_{11} = -1$. ■

8 The Basis b1 = $[(1,-1),(1,0)]$

We shall now apply our method to the problem of matchings - not necessarily perfect - in planar graphs. This is also known as the monomer-dimer problem. Considerable efforts had been expended in attempts to reduce it to the planar perfect matching problem. The lack of success achieved was explained by the work of Jerrum [1987] who showed that this counting problem was #P-complete. Subsequently Vadhan [2001] showed that it remained #P-complete even when the planar graph was bipartite, and its degree was restricted to 6. If the degree is restricted to 2 then the graph consists of a set of cycles and the problem is easily solvable. Any class that allows higher degrees is a natural candidate for #P-compleness. However, we can show that the following such problem is computable in polynomial time.

Theorem 8.1. *There is a polynomial time algorithm for #X-MATCHINGS.*

Proof. Consider a given planar weighted graph $H = (V, E, W)$ where V has bipartition $V1$, $V2$, where every node $v \in V1$ has degree 2 and every node $v \in V2$ has some arbitrary degree $\deg(v)$. We construct a matchgrid Ω_H over **b1** by replacing each $V1$ node with the generator matchgate of Proposition 4.1, replacing each $V2$ node with the recognizer matchgate of Proposition 4.2, and, for each edge (u, v) in H by having a connecting edge joining an output of the generator for u to an input of the recognizer for v so as to preserve planarity. The edge in the recognizer that is adjacent to this connecting edge will have the same weight w_i as the edge (u, v) has in H.

Now the Holant was defined as

$$\text{Holant}(\Omega_H) = \sum_{x \in X} \left[\prod_{1 \le j \le g} \text{valG}(B_j, x) \right] \left[\prod_{1 \le i \le r} \text{valR}(A_i, x) \right].$$

where j ranges over all the generators, i over all the recognizers, and x over all possible tensor products of the basis elements. But each generator has arity two and generates $n \otimes n$, $n \otimes p$, $p \otimes n$, and $p \otimes p$ with weights 1, 1, 1, and 0, respectively. Hence the nonzero contributions to the Holant will come from edge sets of H such that at most one edge from the set is adjacent to each $V1$ node. But the matchgates at the $V2$ nodes are defined so that $\text{valR}(A_i, x)$ is

(i) 0 if there is more than 1 edge incident,

(ii) w_i if there is exactly one, and its weight is w_i, and

(iii) $-(w_1 + \ldots + w_k)$ if there are no incident edges.

Hence the value of the Holant is the sum over all matchings E' of H of the mass of E' defined as follows. The mass of E' is the product of the weights of all the edges that are present in it and also of the value of $-(w_1 + \ldots + w_k)$ for every $V2$ node that is not saturated by the matching. Hence, by virtue of Theorem 4.1 and Corollary 4.1, this mass can be computed in polynomial time, and, in fact, in NC2. ■

9 The Basis b2 = $[(1,1),(1,-1)]$

In this section we study the basis **b2** $= [(1,1),(1,-1)]$, which has a remarkable range of capabilities. We shall assume that field F does not have characteristic two, since then **b2** would have just one

distinct element. We first note that by Propositions 7.1 and 7.3, the arity 2 symmetric signature $[x, y, x]$ can be realized for any x, y, both as a generator and as a recognizer. Thus equality has weight x and inequality has weight y. The case $x = 0$ gives inequality gates and the case $y = 0$ equality gates. The arity one signature $[x, x]$ is also realizable, by 2-node matchgates, but the arity one constants [1, 0] and [0, 1] are not — they would require omittable nodes.

If we have a generator over this basis and join to its outputs equality recognizers then we get a recognizer gate with the same signature as the original generator. Similarly we can convert arbitrary recognizers to generators with the same signature by appending generator equality gates. *Hence for this basis* **b2** *the signatures that can be realized by generators are exactly the same as those that can be realized by recognizers.*

For arity three we shall now enumerate the eight possible combinations of basis elements for the inputs, namely $\mathbf{b2}_{000}, \ldots, \mathbf{b2}_{111}$, rewrite each as an 8-vector of coefficients with respect to the standard basis, and group them according to some semantics:

THREE POSITIVES								
$(1,-1) \otimes (1,-1) \otimes (1,-1)$:	(1	–1	–1	1	–1	1	1	–1)
ZERO POSITIVES								
$(1,1) \otimes (1,1) \otimes (1,1)$:	(1	1	1	1	1	1	1	1)
ONE POSITIVE								
$(1,-1) \otimes (1,1) \otimes (1,1)$:	(1	1	1	1	–1	–1	–1	–1)
$(1,1) \otimes (1,-1) \otimes 1,1)$:	(1	1	–1	–1	1	1	–1	–1)
$(1,1) \otimes (1,1) \otimes (1,-1)$:	(1	-1	1	–1	1	–1	1	–1)
TWO POSITIVE								
$(1,-1) \otimes (1,1) \otimes (1,-1)$:	(1	–1	1	–1	–1	1	–1	1)
$(1,1) \otimes (1,-1) \otimes (1,-1)$:	(1	–1	–1	1	1	–1	–1	1)
$(1,-1) \otimes (1,-1) \otimes (1,1)$:	(1	1	–1	–1	–1	–1	1	1)
SUMS:								
0 OR 3 POSITIVES:	(2	0	0	2	0	2	2	0)
1 OR 2 POSITIVES:	(6	0	0	–2	0	–2	–2	0)

By taking linear combinations of the rows as specified by relation (5.1) we can determine which combinations are realizable standard signatures. By Proposition 6.2 it is sufficient in the arity three case for a standard signature that either all the odd elements, or all the even elements, be zero.

It is clear that if we add the THREE POSITIVES and the ZERO POSITIVES vectors we get an all-even signature (2, 0, 0, 2, 0, 2, 2, 0). It follows that the symmetric signature [1, 0, 0, 1] is realizable for the basis **b2**. Similarly adding the remaining six vectors also gives an all-even vector and hence the symmetric signature [0, 1, 1, 0] is also realizable. Further, if we add x times the first two vectors to y times the last six we still get an all-even vector. Hence for all $x, y \in F$, the symmetric signature $[x, y, y, x]$ is realizable.

Theorem 9.1. *There is a polynomial time algorithm for #PL-3-NAE-ICE.*

Proof. We represent each degree three node of the given graph G by a recognizer matchgate with symmetric signature [0, 1, 1, 0] over **b2**, i.e., the NOT-ALL-EQUAL or NAE gate. For degree two

nodes we have a recognizer for [0, 1, 0] from Proposition 7.3. For each edge we will have a generator matchgate with symmetric signature [0, 1, 0] from Proposition 7.1. We will have connecting edges between the outputs of the generators and inputs of the recognizers as specified by G. If p on a connecting edge of a recognizer gate represents the orientation towards that gate, and an n an orientation away from it then clearly each edge of G will be given a consistent orientation by virtue of the binary inequality generator gate [0, 1, 0], which ensures that its two outputs carry opposite basis elements. Further the recognizer gates will ensure that either one or two of the edges are directed towards it. It follows that the holant of the given matchgrid will equal the desired value of #PL-3-NAE-ICE. ■

Theorem 9.2. *There is a polynomial time algorithm for #PL-3-(1,1)-CYCLECHAIN.*

Proof. Suppose we are given graph G as input to the (1,1) cycle-chain problem. We shall represent each node by a recognizer for $[0, 1, 1, 0]$. We represent each edge of G by a generator for $[1, 0, 1]$. Now if a p generated by a generator signifies that the corresponding edge of G is in the cycle-chain cover then clearly the edges of G will have a consistent such association by virtue of the [1, 0, 1] generators. But the recognizers will ensure that either one or two edges of G incident to any one vertex are labeled p. It follows that there is a one-to-one correspondence between labellings of the edges of G by $\{n, p\}$ such that the edges labeled by p form a cycle-chain cover, and contributions of 1 to the Holant of the constructed matchgrid. The result follows. ■

Theorem 9.3. *There is a polynomial time algorithm for PL-NODE- BIPARTITION.*

Proof. Suppose we are given graph G as input to the PL-NODE-BIPARTITION problem. We shall represent each node by a recognizer for $[x, y, y, x]$ or $[x, y, x]$, depending on whether the degree is three or two, where x, y are variables to be given various values. (Any node of degree one can be simply deleted.) Each edge we represent by a generator for $[0, 1, 0]$. Then as in the proof of Theorem 9.1 we can interpret nonzero contributions to the Holant as orientations of G. Nodes that have all edges directed towards them (sinks) or all edges directed away from them (sources) will give a contribution of x to the Holant, and those that are neither sources or sinks will have a contribution of y. Now if we fix $y = 1$ then the Holant will be a polynomial $SS(x)$ where the coefficient of x^i will be the number of orientations of the edges of G that have exactly i nodes as either sources or sinks.

Now it is easy to verify that the largest i for which the coefficient of x^i in $SS(x)$ is nonzero is the maximum number of nodes that a bipartite graph can have that is obtained by deleting nodes and incident edges from G. In one direction, if there is an orientation with i sources and sinks then the graph induced by the nodes that are sources and sinks in G must be bipartite. In the reverse direction, if we have a bipartite subgraph in G where the nodes have bipartition $V1'$ and $V2'$ then we can define an orientation of G where all the nodes $V1'$ are sources and all the nodes $V2'$ sinks, and the orientation of any edge not incident to $V1'$ or $V2'$ can be arbitrary.

Now by giving x any fixed value we can compute the Holant for that value and hence obtain the value of $SS(x)$. By doing this for $|V| + 1$ distinct values of x and performing polynomial interpolation on the $|V| + 1$ values obtained we can compute all the coefficients of $SS(x)$. The largest i such that the coefficient of x^i in $SS(x)$ is nonzero will give the minimum number $|V| - i$ of nodes whose removal leads to a bipartite graph. ■

The following folds in the results for gates with 1, 2 and 3 inputs described above, with some

result for gates with 4 inputs detailed below, and the equality gate for any number of inputs.

Theorem 9.4. *For matchgrids where each matchgate is one of* $[x, x]$, $[x, y, x]$, $[x, y, y, x]$, $[1, 0, 0, 0, 1]$, $[1, 0, -1, 0, 1]$, $[0, 1, 0, -1, 0]$, $[0, 1, \pm\sqrt{2}, 1, 0]$ *, in the case* $2x^2 = yw+y^2$ $[w, x, y, x, w]$, *and* $[1, 0, \ldots 0, 1]$ *for any arity, the Holant can be computed in polynomial time. Here different matchgates may have different values of* $x, y, w \in F$.

Proof. For arities one, two and three we have already established that $[x, x]$, $[x, y, x]$ and $[x, y, y, x]$ are realizable.

For arity four we shall enumerate the sixteen possible combinations of basis elements for the inputs, namely $\mathbf{b2}_{0000}, \ldots, \mathbf{b2}_{1111}$, rewrite each as a 16-vector of coefficients with respect to the standard basis, and group them according to some semantics, as we did for arity 3. By taking linear combinations of the rows as specified by relation (5.1) we can again determine which combinations are realizable standard signatures. By Proposition 6.3 it is sufficient in the arity four case for a standard signature that all the odd elements, or all the even elements, be zero, provided in addition that the polynomial relation stated there holds among the eight remaining elements.

ZERO POSITIVES																
(1,1)⊗(1,1)⊗(1,1)⊗(1,1):	(1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1)
FOUR POSITIVES																
(1,-1)⊗(1,-1)⊗(1,-1)⊗(1,-1):	(1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1)
CROSSINGS																
(1,-1)⊗(1,1)⊗(1,-1)⊗(1,1):	(1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1)
(1,1)⊗(1,-1)⊗(1,1)⊗(1,-1):	(1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1)
THE OTHER FOUR																
TWO POSITIVES CASES																
(1,1)⊗(1,1)⊗(1,-1)⊗(1,-1):	(1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1)
(1,-1)⊗(1,-1)⊗(1,1)⊗(1,1):	(1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1)
(1,1)⊗(1,-1)⊗(1,-1)⊗(1, 1):	(1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1)
(1,-1)⊗(1,1)⊗(1,1)⊗(1,-1):	(1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1)
ONE POSITIVE																
(1,1)⊗(1,1)⊗(1,-1)⊗(1,1):	(1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1)
(1,-1)⊗(1,1)⊗(1,1)⊗(1,1):	(1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1)
(1,1)⊗(1,1)⊗(1,1)⊗(1,-1):	(1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1)
(1,1)⊗(1,-1)⊗(1,1)⊗(1,1):	(1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1)
THREE POSITIVES																
(1,1)⊗(1,-1)⊗(1,-1)⊗(1,-1):	(1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1)
(1,-1)⊗(1,-1)⊗(1,1)⊗(1,-1):	(1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1)
(1,-1)⊗(1,-1)⊗(1,-1)⊗(1,1):	(1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1)
(1,-1)⊗(1,1)⊗(1,-1)⊗(1,-1):	(1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1)
SUMS:																
0 OR 4 POSITIVES	(2	0	0	2	0	2	2	0	0	2	2	0	2	0	0	2)
TWO POSITIVES	(6	0	0	-2	0	-2	-2	0	0	-2	-2	0	-2	0	0	6)
ONE POSITIVE	(4	2	2	0	2	0	0	-2	2	0	0	-2	0	-2	-2	-4)
THREE POSITIVES	(4	-2	-2	0	-2	0	0	2	-2	0	0	2	0	2	2	-4)

Each 4-output matchgate will have standard signature $(u_{0000}, \ldots, u_{1111})$. Each gate will be either even or odd and will have at most eight of the elements of their signature nonzero. For convenience we shall here represent the signature of an even gate by the 8-vector $(u_{0000}, u_{0011}, u_{0101}, u_{0110}, u_{1001}, u_{1010}, u_{1100}, u_{1111})$ and the signature of an odd gate by the 8-vector $(u_{0001}, u_{0010}, u_{0100}, u_{0111}, u_{1000}, u_{1011}, u_{1101}, u_{1110})$.

(i) Signature $[1,0,0,0,1]$: Adding the signatures for the two cases **(0-positives) + (4-positives)** gives for the even case the 8-vector $(2,2,2,2,2,2,2,2)$ which is feasible by Proposition 6.3(i).

(ii) Signature $[1,0,-1,0,1]$: Forming the linear combination for the eight cases **z(2-positives) + (0 or 4 positives)** gives for the even case the 8-vector $(6z+2, 2-2z, 2-2z, 2-2z, 2-2z, 2-2z, 2-2z, 6z+2)$. By Proposition 6.3(i) this is realizable if $(6z+2)*(6z+2) = (2-2z)(2-2z)$, or $36z^2+24z+4 = 4z^2-8z+4$, or $32z^2+32z = 0$, or $z = -1$.

(iii) Signature $[0,1,0,-1,0]$: Forming the linear combination for the eight cases **(1-positive) - (3 positives)** gives for the odd case the 8-vector $(4,4,4,-4,\ 4,-4,-4,-4)$. By Proposition 6.3(ii) this is realizable since $-16+16+16-16 = 0$.

(iv) Signature $[0,1,\pm\sqrt{2},1,0]$: Forming the linear combination for the fourteen cases **(1-positive) + (3 positives) + y(2 positives)** gives for the even case the 8-vector $(8+6y, -2y, -2y, -2y, -2y, -2y, -2y, -8+6y)$. Therefore it is sufficient that $(8+6y)(-8+6y) = 4y^2$, or $-64+36y^2 = 4y^2$, or $y = \sqrt{2}$ or $y = -\sqrt{2}$.

(v) Signature $[w,x,y,x,w]$: Forming the linear combination for the sixteen cases **w(0-positives) + w(4-positives) + x(1-positive) + x(3 positives) + y(2 positives)** gives for the even case the 8-vector $(2w+6y+8x, 2w-2y, 2w-2y, 2w-2y, 2w-2y, 2w-2y, 2w-2y, 2w+6y-8x)$, for which any y, x, and w with $2x^2 = yw+y^2$ will suffice.

Note that relation (v) generalizes relations (i), (ii) and (iv).

Finally we note that equality gates of any arity m can be obtained by chaining together $m-2$ ternary equality gates $[1,0,0,1]$ using the equality recognizers of Propositon 7.3 . ■

Theorem 9.5. *There is a polynomial time algorithm for#PL-3-NAE-SAT.*

Proof. The construction follows that for Theorem 9.1 except that for NAE nodes we have recognizers with symmetric signatures $[0,1,1,0]$, and for variable nodes we have recognizers for $[1,0,\ldots,0,1]$ gates of the same arity as the number of clauses in which the variable appears. Further, if a variable occurrence is negated we have a $[0,1,0]$ generator along the edge that joins the variable recognizer and the NAE recognizer, and if the variable occurrence is not negated then we have $[1,0,1]$. ■

Theorem 9.6. *There is a polynomial time algorithm for $\oplus$PL-EVEN-LIN2.*

Proof. The construction follows that for Theorem 9.1 with some exceptions. First we note that any equation of even length more than four can be reduced to a set of equations all of length four by the introduction of new variables. For example $z_1+\ldots+z_6 = 1$ becomes the two equations $z_1+z_2+z_3+y = 0$ and $y+z_4+z_5+z_6 = 1$. Now each equation of length four is simulated by a $[1,0,-1,0,1]$, or a $[0,1,0,-1,0]$ gate depending on whether the constant term of the equation being simulated is 0 or 1. For length 2 we use $[1,0,1]$ and $[0,1,0]$, respectively. The boundary conditions that fix the values of variables can be realized by using 2-node matchgates with one omittable node as shown in Figure 3 of Theorem 4.1. We then invoke Corollary 4.2 and Theorem 3.3.

For each original noncompulsory equation we pick an arbitrary variable occurrence in it and simulate it "possibly being faulty" by having as the corresponding link between its variable and equation recognizers a generator for $[1,x,1]$ if the variable occurs positively, and $[x,1,x]$ if it occurs negated. For all other occurrences of variables the corresponding link is a generator for $[1,0,1]$ or $[0,1,0]$ as appropriate. The Holant will then be a polynomial in x. The coefficient of x^i will arise

from "solutions" of the equations where exactly i variable occurrences, all in distinct noncompulsory equations, have their bits inverted. In other words they arise from solutions that satisfy all but exactly i of the noncompulsory equations. Since each solution contributes ± 1 to the Holant the result follows. ■

We note that the generating function of the Ising problem, or #PL-CUT, is nothing other than the Holant when edges are replaced by [1, y, 1] gates and nodes by $[1, 0, \cdots, 0, 1]$ gates over **b2**. Hence this offers yet another treatment of the Ising problem for planar structures. In the reverse direction this implies that algorithms based on just these gates can be derived also from #PL-CUT through classical reductions, essentially following our proofs here. The reader can verify that Theorems 9.1, 9.2, 9.3 and 9.5 are all in this category since [1, y, y, 1] can be simulated by three [1, y, 1] gates. However, if the problems solved in these Theorems are generalized to allow the degree four gates permitted by Theorem 9.4 then polynomial time algorithms follow for some, perhaps less natural, problems for which no classical reduction to #PL-CUT is apparent.

To conclude this section we now summarize more explicitly what the holographic treatment of an instance G of #PL-CUT involves. The basis used will be **b2**. Each node of G of degree d will be replaced by a recognizer for [1, 0, $\cdots$, 0, 1] which enforces equality on its d inputs. Clearly (Theorem 9.4) such a recognizer can be constructed by chaining together d-2 recognizers for [1, 0, 0, 1] with generators for [1, 0, 1]. Each edge of G will be simulated by a generator for [1 ,y, 1]. The simulation consists of replacing the nodes and edges by these gates and joining up the corresponding pairs of input/output nodes of these gates by single edges. The coefficient of y^k in PerfMatch of the resulting weighted graph will give the answer to the #PL-CUT problem by virtue of the Holant Theorem. The only technical facts that need to be verified are that for the basis **b2** generators for [1, y, 1] and [1, 0, 1], and recognizers for [1, 0, 0, 1] exist. Proposition 7.1 guarantees the former. Proposition 6.2 guarantees the latter in conjunction with the observation made at the start of the current section that the sum of "0 or 3 positives" over **b2** corresponds to an all even standard signature.

10 The Basis b3 = [(1, 1), (1, -1), (1, 0)]

We now consider the problem PL-FO-2-COLOR and shall employ this basis **b3**.

Theorem 10.1. *There is a polynomial time algorithm for PL-FO-2-COLOR.*

Proof. Given a graph G we assume that all its nodes have degree two or three. At nodes of degree three we place matchgates that generate:

$(1,0) \otimes (1,1) \otimes (1,1) + (1,1) \otimes (1,0) \otimes (1,1) + (1,1) \otimes (1,1) \otimes (1,0) +$
$(1,0) \otimes (1,-1) \otimes (1,-1) + (1,-1) \otimes (1,0) \otimes (1,-1) + (1,-1) \otimes (1,-1) \otimes (1,0),$

and at nodes of degree two those that generate

$(1,0) \otimes (1,1) + (1,1) \otimes (1,0) +$
$(1,0) \otimes (1,-1) + (1,-1) \otimes (1,0)$.

We note that all the nonzero terms are even and hence by Proposition 6.2 there are matchgates to generate them. In place of the edges of G we place recognizers that on input (a, b), (c, d) at their respective inputs have value $ac - bd$. The recognizer of Proposition 7.6 suffices.

We say that a node of G represents **0** if it generates $(1,1)$ in some direction, and a **1** if it generates $(1,-1)$ in some direction. In either case it directs its arrow along the edge on which it sends $(1,0)$ and away from itself.

Clearly the recognizer between two nodes that both represent **0** or both **1** will have value zero unless at least one of the nodes sends an arrow, i.e., $(1,0)$, to the recognizer, in which case its value will be 1. A recognizer between two nodes that represent **0** and **1** respectively will have value 2 if there are no arrows towards the recognizer, and 1 otherwise.

The Holant of this matchgrid will be nonzero if and only if PL-FO-2-COLOR has a solution. Each solution will be counted 2^k times where k is the number of edges in G that have no arrows and whose endpoints represent opposite values. ■

11 General Complexity-Theoretic Questions

We regard the most important among the currently widely held conjectures of complexity theory to be: (1) P $\neq$ NP, (2) P $\neq$ P$^{\#\mathrm{P}}$, (3) P $\neq$ BPP, (4) P $\neq$ QBP, (5) P $\not\subset$ PolyLogSPACE, (6) P $\neq$ NC, (7) P $\neq$ PSPACE. We observe that a positive solution to the question $P^{\#P} =?NC$ would resolve all the above seven questions (the first six would be contradicted). (N.B. Regarding question (3) P = BPP is also widely conjectured.)

Now consider polynomial systems of the form

$$S(x) = \{E_1(x), E_2(x), \cdots, E_m(x)\},$$

where x stands for a set of variables $\{x_1, \cdots, x_n\}$ and $E_i(x)$ stands for a polynomial equation with coefficients from the integers. We shall say that such a system is *solvable* if it is satisfied by a set of complex numbers. In order that we may invoke Theorem 3.2, which is needed to ensure that the linear algebra computations be polynomial time, we need that such a system be efficiently solvable. We say that a system is *efficiently solvable* if for that *one fixed* system there exists an algorithm that for some polynomial $p(n)$ and any $n > 1$ computes some solution to the system to n decimal places of accuracy within $p(n)$ Boolean operations. This is a weak requirement in that the size of the polynomial system can be regarded as a fixed constant. The requirement is only that the cost of computing the solutions to higher and higher accuracy is polynomial time bounded in the number of digits of the accuracy for that one fixed system. (It need not be polynomial time in the size of the system.)

In fact it can be seen that solvable systems are always also efficiently solvable in our sense: Systems with finite numbers of solutions are efficiently solvable since they can be reduced by elimination to univariate polynomial solving. Further, systems with infinitely many solutions are also efficiently solvable by means of univariate representations [Bürgisser 2000, Theorem 4.12; Basu *et. al.* 2003, Algorithm 11.60]. It is well known that univariate polynomials are efficiently solvable [e.g. Pan, 1997].

We now observe that holographic methods can be viewed as providing constructions of natural systems S such that

$$S(x) \text{ is solvable } \Rightarrow \mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}. \tag{11.1}$$

More particularly, for any one *formulation* F (specifiable by appropriate local constraints) of a combinatorial problem for which the counting problem $\#F$ is #P-complete, and any basis size k, and gate size g, we shall construct a polynomial system $S_{F,k,g}$ such that $S_{F,k,g}$ is solvable if and

only if there is a simple holographic reduction from $\#F$ to planar PerfMatch using basis size k and gate size g.

We can then define S_F to be the family of polynomial systems $\{S_{F,k,g} \mid k = 1, 2, \cdots ; g = 1, 2, \cdots\}$ and make the final claim:

$$\text{Some member of } S_F \text{ is solvable } \Rightarrow \mathrm{P}^{\#\mathrm{P}} = \text{NC2}. \tag{11.2}$$

We shall explain the above claims, in the first instance, in the context of gates of arity up to three and basis size 1, as developed earlier in the paper. In Section 5 we have already stated the polynomial constraints (5.1) on generators

$$u_{ijk} = \sum q_{rst}(\mathbf{b}_{rst})_{ijk} \tag{11.3}$$

and (5.2) on recognizers

$$\hat{q}_{ijk} = \sum \hat{u}_{rst}(\mathbf{b}_{ijk})_{rst} \tag{11.4}$$

where summation is over $\{r, s, t\} \in \{0, 1\}^3$. Note that here q and $\hat{q}$ describe the formulation of the combinatorial problem, $\mathbf{b}$ is the basis, and u and $\hat{u}$ are the standard signatures. Also, from the definition of standard signatures in Section 4, the various components of $u, \hat{u}$ equal PerfMatch$(G - Z)$, PerfMatch$(\hat{G} - Z)$, for various choices of Z, assuming for simplicity, that there is just one kind of generator and one kind of recognizer. Hence the components u_{ijk} of u each equal a polynomial expression, say $U^g_{ijk}(W)$, over the weights W of the generic gate G of size g and similarly for $\hat{u}_{ijk}$. Hence the third set of constraints we need is:

$$u_{ijk} = U^g_{ijk}(W),\ \hat{u}_{ijk} = \hat{U}^{\hat{g}}_{ijk}(\hat{W}). \tag{11.5}$$

It follows from what we have said that the equations (11.3) - (11.5) are solvable if and only if there is a holographic reduction from the given formulation of $\#F$ to planar PerfMatch using basis size 1 and gate size g. We note that in Section 6 we characterized the polynomial equations that can be realized for arities up to four for gates of any size, not just for fixed values of g.

For example, in Section 4 we found a solution in the case that $q = (1, 1, 1, 0)$ and $\hat{q} = (-3, 1, 1, 0, 1, 0, 0, 0)$. (N.B. We had an arity 2 gate for the generators, but arbitrary arity for the recognizers, the $\hat{q}$ here being the instance for arity three.) That gave us a polynomial time algorithm for #X-MATCHINGS. Planar matchings are known to be #P-complete [Jerrum, 1987, 1990], even in the planar bipartite case of maximum degree six [Vadhan, 2001]. Hence the solvability of such a #P-complete case would imply $\mathrm{P}^{\#\mathrm{P}} = \text{NC2}$. We note that the nonsolvability of such systems can be proved mechanically, in principle, using computer algebra systems. With such a system we have verified, for example, that the basis given for #X-MATCHINGS is essentially unique among those of size one.

Now equations (11.3) - (11.5) as stated are limited to bases that have size 1 and two components, and gates of arity up to three. To allow for h rather than two components the only change needed in (11.3) - (11.5) is that $\{r, s, t\}$ should be summed over $\{1, 2, \cdots, h\}$. It is easy to see that these polynomials can be generalized also to allow for arbitrary basis size and arbitrary arity.

By a formulation of a problem we mean a mapping of "the parts" of the problem to generator and recognizer gates in the manner of the reductions we have given for our various specific problems. Given a #P-complete problem such as planar matchings there are many possible formulations and it is not clear which, if any, are the most useful for searching for positive solutions of $\mathrm{P}^{\#\mathrm{P}}$. The

formulation given in Section 8 mapped the nodes to one of two kinds of matchgates. Another would be to map a group of nodes to one kind of matchgate, and the edges to another. Also, as illustrated in some reductions in Section 9, the original problem may be mapped to a number of matchgrids and the final answer recovered by polynomial interpolation. Clearly there are numerous such formulations that one might try. Thus for any combinatorial problem such as those we have described one can ask whether some formulation of some variant is both #P-complete and has a solvable equation system.

Our treatment here emphasizes solutions from $\mathbb{C}$ only because these seem the easiest to find mechanically. Clearly, solutions over finite fields would be even better for computational purposes if these can be found, though the positive consequences would be only for the corresponding fields in the first instance [Valiant, 1979a].

Also, we have defined signatures as matrices whose rows correspond to input configurations of matchgates, and columns to output configurations. In this paper the matchgates we used were all generators or recognizers, corresponding to column and row vectors respectively. The treatment can be adapted, clearly, to matchgates that have both inputs and outputs.

Throughout this paper we have emphasized planar structures. However, within the same framework we can deal with nonplanar structures as long as in their formulation we also allow for "crossover" nodes (and simulate them effectively with matchgates.)

An entirely orthogonal issue is that in this paper we have used the PerfMatch polynomial at the matchgate level, and the FKT method for planar graphs as the combining mechanism. An alternative approach for the whole development is to use the Pfaffian at the matchgate level, and the Pfaffian combining approach described in [Valiant 2002a, Valiant 2005a] instead.

We have considered only matchgrids that use the same basis throughout. We could equally use a different basis for each connection in the matchgrid.

In conclusion we observe that if *any* polynomial system generated in the manner described above for a #P-complete problem is solvable, then it would follow that $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}$ and that the seven conjectures enumerated at the beginning of this Section would be resolved. In the apparent absence of alternative general approaches to these complexity issues, we suggest that as long as the solvability of even one such polynomial system remains unresolved, it is rational to regard these complexity questions as being truly open.

12 Numerical Considerations

Proof of Theorem 3.2. We shall use Berkowitz' algorithm for computing the determinant [Berkowitz 1984] and exploit the fact that, unlike Gaussian elimination, it uses no division. Inspection of Berkowitz' algorithm shows that it uses $3\log_2 n + O(1)$ levels of multiplications of pairs of matrices of sizes at most $n \times n$, where the matrix entries initially are either -1 or members of Y, and at subsequent steps are the entries, sometimes multiplied by -1, of matrix products previously obtained.

For $x \in \mathbb{C}$ let $|x|$ be the modulus of x. Let $D = \max\{1, \max\{|x| : x \in Y\}\}$. Our algorithm will depend on Y only through the value of D. For all matrices it will execute the same sequence of arithmetic operations defined by Berkowitz algorithm except that the arithmetic will be performed in arithmetic with $g = g(n, Y) = O(n^3)(\log_2 D + \log_2 n)$ decimal places of accuracy in fixed precision arithmetic both to the left and to the right of the decimal point. The roundoff error introduced in each operation is at most 2^{-g} in absolute value.

We want F_i to be an upper bound on the modulus of any value computed at the i-th level of the exact algorithm. Clearly $F_0 = D$ and $F_i > (F_{i-1})^2 n^k$ suffice if each level is a matrix multiplication

of matrices of size at most $n^k \times n^k$. It follows that if $\exp(i) = 2^i$ then $F_i = (n^k D)^{\exp(i)}$ suffices.

We now want ϵ_i to be an upper bound on the maximum absolute error on an output of level i that can occur through the accumulation of roundoff errors. We take $\epsilon_0 = 2^{-g}$ and will maintain $2^{-g} \leq \epsilon_i \leq 1/2$ and $F_i \geq 1$ by induction. Now the maximum value that can be taken by a product of true absolute values U and V is

$$\begin{aligned}(U + \epsilon_{i-1})(V + \epsilon_{i-1}) + 2^{-g} &= UV + (U + V)\epsilon_{i-1} + \epsilon_{i-1}^2 + 2^{-g} \\ &\leq UV + 2\epsilon_{i-1}F_{i-1} + \epsilon_{i-1}^2 + 2^{-g} = UV + \epsilon_i'\end{aligned}$$

say. The maximum error of a subsequent n^k-fold sum, as required by a matrix multiplication, performed as $n^k - 1$ pairwise operations is $(n^k(\epsilon_i' + 2^{-g})$. Combining these gives that

$$\epsilon_i \leq n^k(2\epsilon_{i-1}F_{i-1} + \epsilon_{i-i}^2 + 2.2^{-g}) \leq 6n^k F_{i-1}\epsilon_{i-1} \leq (6n^k)^i F_{i-1}F_{i-2}\ldots F_0\epsilon_0$$

where for the second inequality we have used $\epsilon_{i-1}^2 \leq 2F_{i-1}\epsilon_{i-1}$ (since $F_{i-1} > 1$ and $\epsilon_{i-1} \leq \epsilon$), and also $2^{-g} \leq F_{i-1}\epsilon_{i-1}$. Since $F_i = (n^k D)^{\exp(i)}$ we deduce that

$$\epsilon_i \leq (6n^k)^i (n^k D)^{\exp(i+1)} 2^{-g}. \qquad (*)$$

Now if we want to ensure that the integer value of the determinant is computed correctly for $i = 3\log_2 n + O(1)$ and $k = 1$ then $\epsilon_i < 1/2$ is needed for these parameters. From inequality (*) it follows that $g = O(n^3)(\log_2 D + \log_2 n) + O(\log n)^2$ decimal places of accuracy to the right of the decimal point are enough. Since no term is larger than $F_i = (n^k D)^{\exp(i)}$ it follows that $O(n^3)(\log_2 D + \log_2 n)$ decimal places to the left of the decimal point are sufficient, and hence $O(n^3)(\log_2 D + \log_2 n)$ bit arithmetic overall will suffice. ■

Corollary 3.2.1. *The algorithm in the above theorem can be implemented in NC2.*

Proof. Each of the $O(\log n)$ stages of the algorithm can be implemented by Boolean circuits of polynomial size and $O(\log n)$ depth, since it requires multiplications and n-fold additions [Beame, Cook and Hoover 1986]. ■

Corollary 3.2.2. *Theorem 3.2 and Corollary 3.2.1 also hold if Y is infinite, M_n contains elements from some $Y_n \subseteq Y$, there is a polynomial $p(n)$ such that $2^{p(n)}$ upper bounds the absolute value of the elements of Y_n, and there is an algorithm that given n and the index of an element in Y_n computes that element to absolute error less than 2^{-n} in time polynomial in n.*

Proof. The proofs above support this stronger statement. ■

Proof of Theorem 3.3. Theorem 6.2 implies that there is a 3-input gate for the even parity standard signature [1, 0, 1, 0]. By chaining $n-2$ of these together we get a gate for the even parity n input signature [1, 0, 1, 0, $\cdots$]. By deleting one of the external nodes of such a chain we obtain an $n-1$ input odd parity signature [0, 1, 0, 1, $\cdots$].

Given a matchgrid Γ with m non-omittable nodes and r omittable nodes on the boundary we shall create a new matchgrid by adding an r-input parity gate in the outside face of Γ with its external nodes identified with the omittable nodes of Γ. The parity of this parity gate will be chosen odd or even according to whether m is odd or even. Clearly PerfMatch for the augmented matchgrid will equal MatchSum for the original matchgrid, as required. ■

13 Note Added in July 2007.

Since the first appearance of this paper several results have been obtained that shed further light on holographic algorithms. Cai and Choudhary [2006a] gave a tensor based treatment and an alternate proof of the Holant Theorem. Cai and Choudhary [2006b] also showed that any standard signature of arity n can be realized by a planar matchgate with $O(n^4)$ nodes thus, generalizing the corresponding result for $n \leq 4$ of our Section 6. Further, Cai and Choudhary [2006c] showed that the planar matchgrid approach taken here is essentially equivalent to the Pfaffian matchcircuit approach of [Valiant, 2005a]. In Valiant [2006] it was shown that the Cai-Choudhary [2006b] result could be used to show that a certain elementary class of holographic algorithms is insufficent to compute Boolean satisfiabiltiy or the permanent. It leaves open whether more general holographic algorithms can compute these functions, or whether this elementary class in sufficient for other #P-complete problems. Cai and Lu [2007a] gave explicit characterizations of certain classes of signatures that are realizable. Cai and Lu [2007b] further showed that any matchgrid that is realizable in some basis is also realizable in a basis of size one. Holographic reductions have been also used to prove some new $\oplus$P-completeness [Valiant 2006] and #P-completeness [Xia, Zhang and Zhao, 2007] results.

Acknowledgements: I am grateful to Matthew Cook, Oded Goldreich, and Mark Jerrum for their helpful comments on an earlier draft of this paper.

References

Barbanchon, R., On unique graph 3-colorability and parsimonious reductions in the plane. *Theoretical Computer Science* 319, 455-482.

Basu, S., Pollack, R., and Roy, M.-F. 2003. *Algorithms in Real Algebraic Geometry*, Spring-Verlag, Berlin.

Baxter, R. J. 1982. *Exactly Solved Models in Statistical Physics,* Academic Press, London.

Beame, P., Cook, S. A., and Hoover, H. J. 1986. Log depth circuitsfor division and related problems, *SIAM J. Comput.* 15(4): 994–1003.

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

Bernstein, E. and Vazirani, U. V. 1997. Quantum complexity theory, *SIAM J. Comput.* 26: 1411–1473.

Bürgisser, P., Clausen, M., and Shokrollahi, M. A. 1996. *Algebraic Complexity Theory*, Springer-Verlag, Berlin.

Bürgisser, P., 2000. *Completeness and Reduction in Algebraic Complexity*, Springer-Verlag, Berlin.

J.-Y. Cai and V. Choudhary 2006a. Valiant's holant theorem and matchgate tensors. *Lecture Notes in Comptuer Science*, 3959 (2006) 248-261.

J.-Y. Cai and V. Choudhary 2006b. On the theory of matchgate computations, *ECC-018*, 2006).

J.-Y. Cai and V. Choudhary 2006c. Some results on matchgates and holographic algorithms. *Lecture Notes in Computer Science* 4051, (2006) 703-714.

J.-Y. Cai and P. Lu 2007a. Holographic algorithms: from art to science. *Proc. 39th ACM Symp. on Theory of Computing* (2007) 401-410.

J.-Y. Cai and P. Lu 2007b. Holographic algoirthms: the power of dimensionality resolved. *ICALP*, (2007).

Cowen, L. J., Goddard, W., and Jesurum, C. E. 1997. Defective coloring revisited, *J. Graph Theory*: 205–219.

Cook, S. A. 1971. The complexity of theorem proving procedures. *Proc. 3rd ACM Symp. on Theory of Computing*: 151–158.

Deutsch, D. 1985. Quantum theory, the Church-Turing principle, and the universal quantum computer, *Proc. Roy. Soc. London Ser. A*, 400: 97–117.

Fisher, M. E. 1961. Statistical mechanics of dimers on a plane lattice, *Phys. Rev.* 124: 1664–1672.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. 1976. Some simplified NP-complete graph problems, *Theoretical Computer Science* 1: 237–267.

Garey, M. R. Johnson, D. S., and Tarjan, R. E. 1976. The planar Hamiltonian circuit problem is NP-complete, *SIAM J. Comput.* 5, 4: 704–714.

Garey, M. R. and Johnson, D. S. 1977. The rectilinear Steiner tree problem is NP-complete, *SIAM J. Appl. Math.* 32: 826–834.

Hadlock, F. 1975. Finding a maximum cut of a planar graph in polynomial time, *SIAM J. Comput.* 4, 3: 221–225.

Hastad, J. 2001. Some optimal inapproximability results, *Journal of ACM* 48: 798–859.

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

Jaeger, F., Vertigan, D. L., and Welsh, D. J. A. 1990. On the computational complexity of the Jones and Tutte polynomials, *Math. Proc. Camb. Phil. Soc.* 108: 35–53.

Jerrum, M. R. 2003. *Counting, Sampling and Integrating: Algorithms and Complexity*, Birkhauser, Basel, Switzerland.

Jerrum, M. R. and Snir, M. 1982. Some exact complexity results for straight-line computations over semirings, *J. ACM* 29(3): 874–897.

Jerrum, M. R. 1987. Two-dimensional monomer-dimer systems are computationally intractable, *J. Statistical Physics* 48, 1/2: 121–134. (Also 1990, 59,3/4: 1087–1088.)

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, pp. 85–103.

Kasteleyn, P. W. 1961. The statistics of dimers on a lattice, *Physica* 27: 1209–1225.

Kasteleyn, P. W. 1967. Graph theory and crystal physics. In *Graph Theory and Theoretical Physics* (F. Harary, ed.), Academic Press, London, 43–110.

Krishnamoorthy, M. S. and Deo, N. 1979. Node-deletion NP-complete problems, *SIAM J. Comput.* 8, 4: 619–625.

Lewis, J. M. and Yannakakis, M. 1980. The node deletion problem for hereditary properties is NP-complete, *J. Comp. and Syst. Sciences* 20: 219–230.

Lichtenstein, D. 1982. Planar formulae and their uses, *SIAM J. Comput.* 11,2: 329–343.

Lieb, E. H. 1967a. Exact solution of the problem of the entropy of two-dimensional ice, *Phys. Rev. Lett.* 18: 692–694.

Lieb, E. H. 1967b. Residual entropy of square ice, *Phys. Rev.* 162: 162–172.

Lieb, E. H. 1967c. Exact solution of the F model of an antiferroelectric, *Phys. Rev. Lett.* 18: 1046–1048.

Lieb, E. H. 1967d. Exact solution of the two-dimensional Slater KDP model of a ferroelectric, *Phys. Rev. Lett.* 19: 108–110.

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

Liskiewicz, M., Ogihara, M. and Toda, S. 2003. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes, *Theoretical Computer Science* 304(1–3): 129–156.

Lovász. L., and Plummer, M. D. 1986. *Matching Theory*, North-Holland, Amsterdam.

Mahajan, M. and Vinay, V. 1999. Determinant: Old algorithms, new insights, *SIAM Journal on Discrete Mathematics* 12(4): 474–490.

Orlova, G. I. and Dorfman, Y. G. 1972. Finding the maximum cut in a graph, *Engineering Cybernetics* 10: 502–506.

Pan, V.V. 1997. Solving polynomial equations: some history and recent papers. *SIAM Review*, 39:2, 187-220.

Papadimitriou, C. H. 1994. *Computational Complexity*, Addison-Wesley, Reading, MA.

Pauling, L. 1935. *J. Am. Chem. Soc.* 57, 2680.

Strassen, V. 1969. Gaussian elimination is not optimal, *Numerische Mathematik* 14(3): 354–356.

Tardos, E. 1987. The gap between monotone and nonmonotone circuit complexity is exponential, *Combinatorica* 7: 141–142.

Temperley, H. N. V. and Fisher, 1961. M. E. Dimer problems in statistical mechanics — An exact result, *Philosophical Magazine* 6: 1061–1063.

Vadhan, S. P. 2001. The complexity of counting in sparse, regular and planar graphs, *SIAM J. Comput.* 31(2): 398–427

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

Valiant, L. G. 1979b. The complexity of enumeration and reliability problems, *SIAM J. on Comput.* 8, 3: 410–421.

Valiant, L. G. 1980. Negation can be exponentially powerful, *Theoretical Computer Science* 12: 303–314.

Valiant, L. G. 1992. Why is Boolean complexity theory difficult? In *Boolean Function Complexity* (M. S. Paterson, ed.), London Mathematical Society Lecture Note Series, Cambridge University Press 169: 84-94.

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

Valiant, L. G. 2002b. Expressiveness of matchgates, *Theoretical Computer Science* 289, 1: 457–471.

Valiant, L. G. 2005a. Holographic circuits, Proc. *ICALP32*, *LNCS*, Vol. 3980, Springer, 1-15.

Valiant, L. G. 2005b. Completeness for parity problems, Proc. 11th COCOON, *LNCS*, Vol. 3959, Springer, 1-9.

Valiant, L.G. 2006. Accidental algorithms *Proc. 47th IEEE Symposium on Foundations of Computer Science* (2006) 509-517.

Vertigan, D. L. and Welsh, D. J. A. 1992. The computational complexity of the Tutte plane: The bipartite case, *Comb. Probab. Comput.* 1(2): 181–187.

Welsh, D. J. A. 1993. *Complexity: Knots, Colourings and Counting*, Cambridge University Press.

Xia, M., Zhang, P. and Zhao. W. 2007. The complexity of counting on 3-regular planar graphs. *Theoretical Comptuer Science*, To appear.