1
Counting Problems

This book is a study of the computational complexity of counting problems, especially those that can be expressed as a sum-of-product computation. Our aim is to give a systematic presentation to a body of work, mostly from the past two decades, that gives some comprehensive classifications to these sum-of-product computations from the perspective of computational complexity. All these sum-of-product problems are computable within the level of the complexity class #P. The classification theorems are stated in the following form, known as a dichotomy: For a class of problems expressible within a framework, every problem in the class is either computable in polynomial time, or it is #P-hard to compute, i.e., it is as hard as any other problem in the class #P.

1.1 Counting Problems and Models of Computation

We assume the readers have some basic and preliminary knowledge about computational complexity such as P and NP, for example, at a level provided by an undergraduate course on the subject of the Theory of Computing. This knowledge is not crucial, however, as we will not use many general results but rather present the framework and develop the necessary proof techniques gradually. Any reader wishing to be more thoroughly acquainted with the full scope of computational complexity theory can consult a standard textbook, e.g., [Sip96, Pap94, AB09]. On the other hand, a reader with no prior exposure to complexity theory but with a strong mathematical background, for example, at a level provided by a solid undergraduate mathematics education, should be able to follow all proofs in the book, provided he or she is willing to accept a handful of results without proof. These results can be found elsewhere, and the particular proofs of these do not impact an understanding of the material in this book. Such a reader, however, would benefit from a wider exposure to

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

complexity theory to gain some insight as to why certain problems are investigated and what questions are asked.

Briefly, in computational complexity theory a *problem* consists of an infinite set of problem instances, rather than one particular instance. For example, the problem DETERMINANT is to compute the determinant of an arbitrarily given matrix, not one particular matrix. Solving a problem is to say that there is an algorithm that solves all problem instances. The formal notion of an algorithm is a Turing machine, an idealized computer that carries out step-by-step operations according to a finitary program, valid for all problem instances. The efficiency of the algorithm is measured in terms of the maximum number of steps $T(n)$ the Turing machine may take for all problem instances of size n. For example, for integer matrices, the problem instance of DETERMINANT is an integer square matrix in which the size is the sum of the bit length of all the matrix entries. Formally the complexity classes P and NP are defined for decision problems, those problems for which the answer to each instance is either Yes or No. For example, the problem VERTEX COVER is the following decision problem: Given a problem instance consisting of a graph $G = (V, E)$ and an integer parameter k, whether there is a subset of vertices $S \subseteq V$ such that $|S| \leq k$ and every edge in E is incident to at least one vertex from S. A decision problem Π is in the complexity class P if there is an algorithm solving the problem, i.e., gives a correct Yes or No answer to every problem instance, such that $T(n)$ is bounded above by a fixed degree polynomial in n. The class NP consists of all decision problems Σ for which there exists some problem Π in P, and some polynomial $p(\cdot)$, such that for all problem instances x of Σ, x is a Yes instance of Σ iff there exists some y with size $|y| \leq p(|x|)$ and the pair $\langle x, y\rangle$ is a Yes instance of Π. Intuitively, y is a certificate of size polynomially bounded in x, such that it can be verified in polynomial time that y certifies that x is a Yes instance of Σ.

There are historical as well as internal logical reasons, mainly from computability theory, why the definitions of P and NP are formulated in terms of decision problems. Even though a problem may be more naturally stated as a search problem or a numerical problem, it can often be restated as a decision problem such that solving the decision problem is equivalent to solving the search or numerical problem within a polynomial factor in efficiency. For example, if there is an algorithm that solves the decision problem VERTEX COVER, then by binary search on the value k one can find the minimum size k_0 of any vertex cover of G. Furthermore, by considering a sequence of at most $O(k_0 n)$ graphs obtained by removing some vertices and their incident edges from G of n vertices, and repeatedly invoking the decision algorithm, one can compute a minimum size vertex cover of G. For the DETERMINANT problem one can

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

formulate a decision problem as whether the determinant of an integer matrix is greater than a threshold value. It is easy to compute an a priori bound for a given determinant; a binary search will then find the exact determinant value in polynomial time from the decision algorithm. Such reductions show that there is a theoretical equivalence between a decision problem and its search or numerical version; however, in reality, a polynomial-time algorithm usually directly computes the output value, without going through the decision version.

In this book we mostly study counting problems. Typically these include counting the number of specific combinatorial configurations, such as vertex covers, matchings, or proper colorings in a graph, or the evaluation of partition functions in statistical physics. For such problems, it is not expected that they are equivalent to their decision problem counterparts in polynomial time [Tod91]. Instead we define #P, a complexity class of functions for counting problems.

For every NP problem Σ one can formulate a corresponding counting problem using the problem Π in P that defines Σ in terms of certificates. For any problem instance x of Σ, we define the function $f(x)$ as the number of certificates y such that the pair $\langle x, y\rangle$ is a Yes instance of Π. The class of all such functions is denoted as #P. This complexity class, defined by Valiant [Val79a, Val79b] in the study of the complexity of the permanent function, will play a central role in this book. Natural counting problems corresponding to decision problems at the level of NP that have a nonnegative integer solution can all be formulated as a problem in #P. To include more problems at this level we also consider problems whose solutions are not necessarily nonnegative integers, such as those from statistical physics. To capture their computational complexity, we consider the closure of #P under polynomial-time Turing reductions, namely, those problems solvable by a polynomial-time algorithm given free access to a hypothetical algorithm to some problem in #P, where each query costs only the time it takes to write the query. This class is formally denoted as $\mathrm{FP}^{\#\mathrm{P}}$, where FP denotes the class of functions computable by a polynomial time Turing machine. Typical problems in #P include counting the number of satisfying assignments to a Boolean formula or the number of vertex covers in a graph. Weighted versions of these problems as well as sum-of-product computations such as partition functions from statistical physics can be easily formulated as problems that are polynomial-time reducible to #P.

The formal Turing machine model is naturally suited to the study of computation over discrete structures such as integers or graphs. However, in this book it is more natural to consider computation over the real or complex numbers. Doing so causes a technical issue of how one may represent exactly the individual real or complex numbers, and how to account for the complexity of various

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

operations on these numbers. This is a question relating to the model of computation. One can, for example, take the computational model on real numbers by Blum–Shub–Smale [BSS89, BCSS98]. This is more intuitive; however, in a strict logical sense this is not equivalent to the classical Turing machine model and the results we obtain would not have the same logical meaning. So, formally we still consider the classical Turing machine model, and restrict the objects of computation to algebraic (complex) numbers. Thus, technically, every number α is specified by a finite irreducible polynomial $f(X) \in \mathbb{Z}[X]$ with integer coefficients, and a small disk of rational radius containing a unique root α of $f(X)$. We discuss models of computation further in Section 1.5. But this is not a central issue for the type of sum-of-product computations we treat in this book; basically the theory can be developed in any reasonable model of computation in which sum and product can be efficiently computed.

Some basic notations. We denote by $\mathbb{N}$ the set of natural numbers $\{0, 1, 2, \ldots\}$, by $\mathbb{Z}$ the set of all integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$, and by $\mathbb{Z}_+$ the set of positive integers $\{1, 2, 3, \ldots\}$. We denote by $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ the sets of rational (algebraic) real, and complex numbers respectively. We denote $\mathrm{i} = \sqrt{-1}$. All graphs are finite and undirected, unless stated otherwise. Graphs may or may not have loops and parallel edges, which should be clear from the context. If α and β are finite bit strings, then α_i denotes its ith bit, and $\alpha \oplus \beta$ denotes its bitwise XOR string. For a binary string α, its Hamming weight is denoted by $\mathrm{wt}(\alpha)$, i.e., the number of 1's in α. We will generally treat a column vector $v \in \mathbb{C}^n$ and the row vector v^{T}, its transpose, interchangeably. For a matrix $M \in \mathbb{C}^{m \times n}$ and a $v \in \mathbb{C}^n$, we write Mv as its matrix vector product. Sometimes we may write Mv even though we explicitly listed the elements of v as a row vector. The formal meaning is just Mv^{T}, if v is a row vector. Similar comments apply for vM. If $A = (a_{ij}) \in \mathbb{C}^{m \times n}$ and $B = (b_{st}) \in \mathbb{C}^{k \times \ell}$, we denote $A \otimes B$ as the tensor product matrix in $\mathbb{C}^{mk \times n\ell}$, indexed by the pairs $(i, s) \in [m] \times [k]$ for rows and $(j, t) \in [n] \times [\ell]$ for columns, both in lexicographic order.

We use $\leq_{\mathrm{T}}^{\mathrm{P}}$ or simply $\leq_{\mathrm{T}}$ to denote polynomial time Turing reducibility, and $\equiv_{\mathrm{T}}^{\mathrm{P}}$ or simply $\equiv_{\mathrm{T}}$ to denote polynomial time Turing equivalence.

1.2 Three Classes of Counting Problems

We now formally define three frameworks for the types of sum-of-product computations in this book. Fix a finite domain $[q] = \{1, 2, \ldots, q\}$, for a positive integer q. If $q = 2$ it is called the Boolean domain. We consider any set of functions $\mathcal{F}$ on domain $[q]$, where each $f \in \mathcal{F}$ maps from $[q]^k$ to a commutative semiring $\mathbf{R}$, for some $k \geq 0$, called the arity of f. If $k = 0$, then f is a constant. If $k = 1$, 2, or 3, then f is called a unary, binary, or ternary function

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

respectively. In this book, by default we will take the semiring $\mathbf{R}$ to be the (algebraic) complex numbers, but still denote it as $\mathbb{C}$. Functions in $\mathcal{F}$ are also called *signatures* or local constraint functions. A signature f is symmetric if its value is invariant under permutation of its variables. For the Boolean domain this means that the value of f depends only on the Hamming weight of its input.

1.2.1 Spin Systems or Graph Homomorphism Problems

A spin system on a graph $G = (V, E)$ is the following model and it comes from statistical physics. Let $[q] = \{1, \ldots, q\}$ be a finite domain, where $q \geq 1$ is an integer. We consider all vertex assignments $\sigma : V \to [q]$. There is an edge function $f : [q]^2 \to \mathbb{C}$. For each assignment σ we have an evaluation $\prod_{(u,v)\in E} f(\sigma(u), \sigma(v))$, a product over every edge $(u, v) \in E$. Then we define the *partition function*

$$Z_f(G) = \sum_{\sigma:V\to[q]} \prod_{(u,v)\in E} f(\sigma(u), \sigma(v)). \tag{1.1}$$

In physics, the partition function represents the aggregate of thermodynamic variables or a normalizing factor of a system, as one sums over all possible configurations of the particles.

The value $f(\sigma(u), \sigma(v))$ is the local contribution, the product $\prod_{(u,v)\in E} f(\sigma(u), \sigma(v))$ is the weight of the assignment σ, and the partition function is the sum of weights over all assignments. If G is an undirected graph, as is typically the case, we require f to be a symmetric function, $f(i, j) = f(j, i)$, for all $i, j \in [q]$.

If $q = 2$ this is called a 2-spin system, and for general q it is called a q-spin system.

Well-known examples of a 2-spin system include the Ising model [Isi25], where $f(0, 0) = f(1, 1) = a$ and $f(0, 1) = f(1, 0) = b$ for some two constants a and b. Sometimes there is also a vertex weight function, represented by a unary function normalized to $u(0) = 1$ and $u(1) = \lambda$. Then the partition function of the Ising model is

$$Z_{a,b,\lambda}(G) = \sum_{\sigma:V\to[2]} a^{|\{(u,v)\in E:\sigma(u)=\sigma(v)\}|} b^{|\{(u,v)\in E:\sigma(u)\neq\sigma(v)\}|} \lambda^{|\{v\in V:\sigma(v)=1\}|}. \tag{1.2}$$

Exercise: Fix any constant J. One can define the Hamiltonian function $H(\sigma)$ for any assignment $\sigma : V \to \{-1, +1\}$,

$$H(\sigma) = -J \sum_{(u,v)\in E} \sigma(u)\sigma(v).$$

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

(It is called ferromagnetic if $J > 0$ and antiferromagnetic if $J < 0$.) Then the partition function of the Ising model can be defined by $Z(G) = \sum_{\sigma: V \to \{-1,+1\}} e^{-\beta H(\sigma)}$, where $\beta \geq 0$ is called the inverse temperature. Show that this function $Z(G)$ can be realized by the sum-of-product expression $Z_{a,b,\lambda}(G)$.

A generalization of the Ising model is called the Potts model [Pot52], where for $q \geq 2$ and $i, j \in [q]$, $f(i, i) = a$ and $f(i, j) = f(j, i) = b$ for $i \neq j$. One can normalize $b = 1$ and write a as $1 + \gamma$ for a parameter γ. In this parameterization the partition function of the Potts model is

$$Z_{\text{Potts}}(G; q, \gamma) = \sum_{\sigma: V \to [q]} \prod_{(u,v) \in E} (1 + \gamma \delta(\sigma(u), \sigma(v))), \tag{1.3}$$

where $\delta(i, j) = 1$ if $i = j$, and 0 otherwise.

The partition function of the Potts model can be linked to the Tutte polynomial $\mathrm{T}(G; x, y)$ by setting $\gamma = y - 1$ and $q = (x - 1)(y - 1)$. Indeed, one way to define the Tutte polynomial in terms of q and γ is that $(x - 1)^{\kappa(V,E)}(y - 1)^{|V|}\, \mathrm{T}(G; x, y)$ is equal to (cf. Definition 6.27)

$$Z_{\text{Tutte}}(G; q, \gamma) = \sum_{F \subseteq E} q^{\kappa(V,F)} \gamma^{|F|},$$

where $\kappa(V, F)$ is the number of connected components in the spanning subgraph (V, F).

Although the Tutte polynomial is defined for any q, if we restrict to a positive integer q, then

$$Z_{\text{Tutte}}(G; q, \gamma) = Z_{\text{Potts}}(G; q, \gamma).$$

To prove this equality, we consider expanding the product $\prod_{(u,v) \in E}(1 + \gamma \delta(\sigma(u), \sigma(v)))$ in (1.3) as a sum indexed by $F \subseteq E$, which collects a factor $\gamma^{|F|}$, namely

$$\prod_{(u,v) \in E} (1 + \gamma \delta(\sigma(u), \sigma(v))) = \sum_{F \subseteq E} \gamma^{|F|} \prod_{(u,v) \in F} \delta(\sigma(u), \sigma(v)).$$

The product term indexed by F is 1 iff $\sigma(u) = \sigma(v)$ for all u and v that belong to the same connected component of the subgraph (V, F), and 0 otherwise. In (1.3) the sum $\sum_{\sigma: V \to [q]}$ has exactly $q^{\kappa(V,F)}$ such terms.

Spin systems are also called graph homomorphisms, and they come from a different source. Given two graphs G and H, a graph homomorphism from G to H is a map σ from the vertex set of G to the vertex set of H such that for every edge (u, v) in G, the image is also an edge in H. In general, multigraphs are allowed; thus, for example, an edge in G can be mapped to a loop in H. The

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press



(a) #VertexCover (b) #Antichain

Figure 1.1. Target graphs H and the combinatorial counting problems they define as #H-coloring problems.

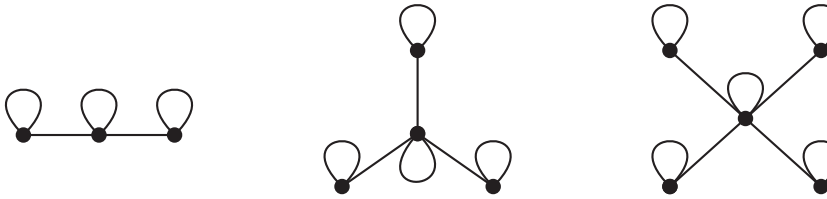
counting problem is to compute the number of graph homomorphisms from G to H.

We will fix an H, and consider the computational problem where G is the input. A special case is when $H = K_q$, the complete graph on q vertices, without self-loops. In this case, a graph homomorphism from G to H is a valid coloring of the vertices of G, using at most q distinct colors, as any adjacent pair of vertices of G must be mapped to distinct vertices of H. Owing partly to this special case, counting graph homomorphisms to H in general is also called #H-colorings, and H is called the target graph.

Graph homomorphisms can express a variety of combinatorial problems. In fact, their principal purpose is to express and then treat a wide variety of locally defined graph properties in a uniform way. For example, if H is a graph consisting of two vertices {T, F} and two edges {(T, T), (T, F)}, a loop and an edge between the two vertices (see Figure 1.1a), then the #H-coloring problem is the counting problem of Vertex Cover, i.e., to count the number of vertex covers in graph G, denoted as #VC. Indeed a homomorphism from G to this H is a two-coloring of vertices of G such that the subset of vertices of G mapped to T forms a vertex cover. By flipping the intended meaning of T and F, this also counts the number of independent sets in G. Of course this simply reflects the fact that a subset $S \subseteq V(G)$ is a vertex cover iff its complement $S^c \subseteq V(G)$ is an independent set. As another example, suppose H is the two-vertex directed graph connected by a single directed edge and both vertices have one directed self-loop (see Figure 1.1b). Then the #H-coloring problem takes directed graphs as input. If the input is a directed acyclic graph, then it defines a partial order, and the #H-coloring problem is to compute the number of antichains (or equivalently, the number of lower sets, i.e., downward closed sets in the partial order, or equivalently, the number of upper sets) in this partial order. If the input is not acyclic, then every strongly connected component must be mapped to one vertex in H, and the #H-coloring problem is to count the number of antichains in the induced partial order after collapsing each strongly connected component.

Exercise: Show that #VC can be expressed as #H-coloring by defining explicitly the binary constraint function.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press



(a) $q = 2$ (b) $q = 3$ (c) $q = 4$

Figure 1.2. Target graph H for counting q-particle Widom–Rowlinson configurations for $q \in \{2, 3, 4\}$ as an $\#H$-coloring problem.

Exercise: Show that there is a bijection between the set of antichains and the set of lower sets. Conclude that the number of antichains is the same as the number of lower sets. By symmetry the same is proved for upper sets.

More generally, we consider weighted graph homomorphisms . Let A be a q-by-q matrix over $\mathbb{C}$. Given a graph or a directed graph $G = (V, E)$, the (weighted) graph homomorphism problem is to compute

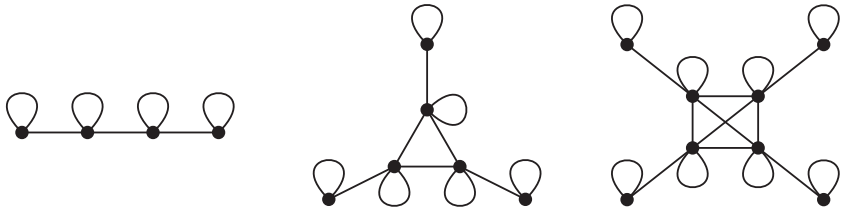
$$Z_A(G) = \sum_{\sigma: V \to [q]} \prod_{(u,v) \in E} A_{\sigma(u),\sigma(v)}. \tag{1.4}$$

The target graph H is now defined by the matrix A, which is the weighted adjacency matrix of H. When A is a 0-1 matrix, then this is the unweighted version of graph homomorphism. The matrix A can also be identified as a binary function. In the case of Vertex Cover the function is the binary Boolean Or function. For Independent Set this is the Nand function. For the problem of counting antichains, the function is the binary Implication function. For q-coloring, this is the binary Disequality function on domain $[q]$.

This is exactly the same notion of a partition function in statistical physics. The Ising model corresponds to the partition function with matrix $A = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$. The Potts model corresponds to the partition function with matrix $A = J_q + \gamma I_q$, were J_q is the $q \times q$ matrix of all 1's and I_q is the $q \times q$ identity matrix, and γ is a parameter.

The q-particle Widom–Rowlinson model [WR70] corresponds to the $\#H$-coloring problem in which the domain size is $q + 1$, H is the star graph on $q + 1$ vertices and all vertices have self-loops (see Figure 1.2). The q-type Beach model [BS94, BS95] corresponds to the $\#H$-coloring problem in which the domain size is $2q$, H is the complete graph on q vertices, each of these q vertices has a pendant vertex, and all $2q$ vertices have a self-loop (see Figure 1.3).

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press



(a) $q = 2$ (b) $q = 3$ (c) $q = 4$

Figure 1.3. Target graph H for counting q-type Beach model configurations for $q \in \{2, 3, 4\}$ as an #H-coloring problem.

For more on this connection with statistical physics, see [Wel93, Chapter 4] or [Thu09, Chapter 2] (as well as [Bax82]).

Graph homomorphism can be viewed as a special case of counting constraint satisfaction problems, where instead of one binary constraint function there can be a set of constraint functions.

1.2.2 Constraint Satisfaction Problems

A counting constraint satisfaction problem (#CSP) [CKS01] is parametrized by a set of local constraint functions $\mathcal{F}$. It is denoted by #CSP$_q(\mathcal{F})$ when the constraint functions in $\mathcal{F}$ are defined over a domain $[q]$ of size q. An instance of #CSP$_q(\mathcal{F})$ is a finite set of variables $x_1, x_2, \ldots, x_n$, and a finite set C of clauses. Each clause is a constraint $f \in \mathcal{F}$ of some arity k depending on f together with a sequence of k (not necessarily distinct) variables $x_{i_1}, \ldots, x_{i_k} \in \{x_1, x_2, \ldots, x_n\}$. The output is

$$\sum_{x_1, \ldots, x_n \in [q]} \prod_{(f, x_{i_1}, \ldots, x_{i_k}) \in C} f(x_{i_1}, \ldots, x_{i_k}). \tag{1.5}$$

In the study of #CSP$_q(\mathcal{F})$, the set $\mathcal{F}$ is usually finite and considered fixed. In particular, there is a maximum arity k among functions in $\mathcal{F}$. An input instance on n variables can be described by n^k bits for a fixed k. Hence we consider the input size is n. When it is the Boolean domain (i.e., $q = 2$), we denote it simply as #CSP$(\mathcal{F})$. If $\mathcal{F}$ consists of a single function f we write #CSP(f) for #CSP$(\{f\})$. We write similarly #CSP(f, g) if $\mathcal{F} = \{f, g\}$, and #CSP$(\mathcal{F}, g)$ for #CSP$(\mathcal{F} \cup \{g\})$, etc.

The canonical example of a #CSP is #Sat, or counting Boolean Satisfiability, the problem of counting the number of satisfying assignments to a given Boolean formula. As a constraint satisfaction problem, it is #CSP$(\mathcal{F})$, with

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

$\mathcal{F} = \{\text{OR}_k | k \geq 1\} \cup \{\neq_2\}$, where OR_k is the OR function of arity k and $(\neq_2)$ is the binary DISEQUALITY function. Here are several more well-known examples of #CSP($\mathcal{F}$)'s over the Boolean domain and their corresponding constraint sets:

SAT	has	$\mathcal{F} = \{\text{OR}_k \mid k \geq 1\} \cup \{\neq_2\}$
3SAT	has	$\mathcal{F} = \{\text{OR}_3, \neq_2\}$
1-IN-3SAT	has	$\mathcal{F} = \{\text{EXACT-ONE}_3, \neq_2\}$
NAE-3SAT	has	$\mathcal{F} = \{\text{NOT-ALL-EQUAL}_3, \neq_2\}$
MON-SAT	has	$\mathcal{F} = \{\text{OR}_k \mid k \geq 1\}$
MON-3SAT	has	$\mathcal{F} = \{\text{OR}_3\}$
MON-1-IN-3SAT	has	$\mathcal{F} = \{\text{EXACT-ONE}_3\}$
MON-NAE-3SAT	has	$\mathcal{F} = \{\text{NOT-ALL-EQUAL}_3\}$

By #CSP$^d(\mathcal{F})$, we denote the special case of #CSP($\mathcal{F}$) in which every variable appears a multiple of d times. Note that #CSP($\mathcal{F}$) is the same as #CSP$^1(\mathcal{F})$, and #CSP$^2(\mathcal{F})$ is the same as every variable appearing an even number of times.

1.2.3 Holant Problems

A Holant problem is parametrized by a set of local constraint functions $\mathcal{F}$, also called signatures . A *signature grid* $\Omega = (G, \pi)$ over $\mathcal{F}$ consists of a graph $G = (V, E)$ and a mapping π that assigns to each vertex $v \in V$ an $f_v \in \mathcal{F}$ and a linear order of the incident edges at v. The arity of f is equal to the degree at v, and the incident edges at v are associated with the input variables of f_v. If all signatures in $\mathcal{F}$ are symmetric then there is no need to assign an order for incident edges at any v.

Definition 1.1. For a set $\mathcal{F}$ of signatures over a domain $[q]$, we define Holant$_q(\mathcal{F})$ as

Input: A *signature grid* $\Omega = (G, \pi)$ over $\mathcal{F}$

Output:

$$\text{Holant}_q(\Omega; \mathcal{F}) = \sum_{\sigma: E \to [q]} \prod_{v \in V} f_v(\sigma \mid_{E(v)}),$$

where

- $G = (V, E)$, and $E(v)$ denotes the incident edges of v and
- $\sigma \mid_{E(v)}$ denotes the restriction of σ to $E(v)$, and $f_v(\sigma \mid_{E(v)})$ is the evaluation of f_v on the ordered input tuple $\sigma \mid_{E(v)}$.

In Volume I of this book we exclusively present the theory over the Boolean domain $q = 2$. We use Holant($\mathcal{F}$) to denote Holant$_2(\mathcal{F})$. We also denote

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Holant(Ω) or Holant$_\Omega$ for Holant(Ω; $\mathcal{F}$), omitting $\mathcal{F}$ in the expression when $\mathcal{F}$ is clear from the context. We use G in place of Ω when π is clear from context. We write Holant$_q(\mathcal{F}, g)$ for Holant$_q(\mathcal{F} \cup \{g\})$. When $\mathcal{F}$ is a finite set of signatures, we sometimes just list the signatures it contains. For example, if $\mathcal{F} = \{f, g\}$, then instead of writing Holant$_q(\{f, g\})$, we may also write Holant$_q(f, g)$. This is especially true when $\mathcal{F}$ is a singleton set.

A signature f of arity k over the domain $[q]$ can be denoted by $(f_0, \ldots, f_x, \ldots, f_{q^k-1})$, where f_x is the output of f on $x \in [q]^k$, ordered lexicographically as integers from 0 to $q^k - 1$. We also write it as a column vector. This listing of values of f in lexicographical order is the same as in a truth table. It is a vector in $\mathbb{C}^{q^k}$, or a tensor in $(\mathbb{C}^q)^{\otimes k}$ indexed by $[q]^k$. A symmetric signature f of arity k over the Boolean domain can be expressed as $[f_0, f_1, \ldots, f_k]$, where f_w is the value of f on inputs of Hamming weight w. An example is the Equality signature $(=_k) = [1, 0, \ldots, 0, 1]$ of arity k, where there are $k - 1$ entries of 0. Another example is the binary Disequality function $(\neq_2) = [0, 1, 0]$.

We give some examples of Holant problems using a symmetric signature f. If f is the only available signature and its arity is k, then the graph G in a Holant problem using $\{f\}$ must be k-regular, i.e., every vertex of the graph G has degree k. Here are four examples over the Boolean domain.

$$\text{Holant}(G; f) \text{ counts} \begin{cases} \text{matchings} & \text{in } G \text{ when } f = \text{At-Most-One}_k; \\ \text{perfect matchings} & \text{in } G \text{ when } f = \text{Exact-One}_k; \\ \text{cycle covers} & \text{in } G \text{ when } f = \text{Exact-Two}_k; \\ \text{edge covers} & \text{in } G \text{ when } f = \text{Or}_k. \end{cases}$$

An example problem over domain size q is q-Edge-Coloring, Holant$_q(G$; All-Distinct), which counts proper edge colorings in G using at most q colors; i.e., at every vertex of G the incident edges are colored with distinct colors from $[q]$. Each of these five examples is expressed as a Holant problem in a straightforward manner. A less obvious example is that Holant$(G; \frac{1}{2}[3, 0, 1, 0, 3])$ counts the number of Eulerian orientations of a 4-regular graph G; this will become clear when we develop the theory further.

As stated earlier, one can view a signature of arity k as a tensor along with a default choice of basis $\{e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_k} \mid i_1 i_2 \ldots i_k \in \{0, 1\}^k\}$, where $e_0 = (1,\ 0)$ and $e_1 = (0,\ 1)$. Readers who are familiar with quantum computation may want to write it as $e_0 = |0\rangle$, $e_1 = |1\rangle$, and $e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_k} = |i_1 i_2 \ldots i_k\rangle$. When doing so, a signature grid is equivalent to a tensor network, and the Holant of the signature grid is equal to the scalar that remains after contracting all edges in the corresponding tensor network. (However, in this book

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

we will not assume or use any knowledge of quantum computing, and will not use these quantum computing notations.)

A *planar signature grid* is a signature grid such that its underlying graph is planar and for some planar embedding, for every vertex v, the linear order of the incident edges at v agrees with the clockwise cyclic order of the incident edges at v in the embedding starting with some particular edge. We use Pl-Holant$_q(\mathcal{F})$ to denote the restriction of Holant$_q(\mathcal{F})$ to planar signature grids. For signature sets $\mathcal{F}$ and $\mathcal{G}$, a *bipartite signature grid* over $(\mathcal{F} \mid \mathcal{G})$ is a signature grid $\Omega = (H, \pi)$ over $\mathcal{F} \cup \mathcal{G}$, where $H = (V, E)$ is a bipartite graph with bipartition $V = (V_1, V_2)$ such that $\pi(V_1) \subseteq \mathcal{F}$ and $\pi(V_2) \subseteq \mathcal{G}$. Signatures in $\mathcal{F}$ are considered as row vectors (or covariant tensors); signatures in $\mathcal{G}$ are considered as column vectors (or contravariant tensors) [DP91] (see also chapter 2 of [Rom12]). We use Holant$_q(\mathcal{F} \mid \mathcal{G})$ to denote the restriction of Holant$_q(\mathcal{F} \cup \mathcal{G})$ to bipartite signature grids over $(\mathcal{F} \mid \mathcal{G})$, and Holant$_q(\Omega; \mathcal{F}|\mathcal{G})$ to denote the value on Ω. A *planar bipartite signature grid* is one that is both planar and bipartite. We use Pl-Holant$_q(\mathcal{F} \mid \mathcal{G})$ to denote the restriction to these signature grids, and Pl-Holant$_q(\Omega; \mathcal{F}|\mathcal{G})$ to denote the value on Ω. We use LHS and RHS to stand for the left-hand side and right-hand side respectively.

A signature f of arity k is *degenerate* if there exist unary signatures $u_j \in \mathbb{C}^q$ $(1 \le j \le k)$ such that $f = u_1 \otimes \cdots \otimes u_k$. Replacing such an f by the k unary signatures u_j, one for each associated edge, does not change the Holant value. A symmetric degenerate signature has the form $u^{\otimes k}$, for some unary $u \in \mathbb{C}^q$. Replacing a signature $f \in \mathcal{F}$ by a constant multiple cf, where $c \neq 0$, does not change the complexity of Holant$_q(\mathcal{F})$. It merely introduces a nonzero factor c^n to Holant$_q(\Omega; \mathcal{F})$, where f appears n times in Ω.

Exercise: Prove that the Holant value is unchanged when $f = u_1 \otimes \cdots \otimes u_k$ is replaced by unary signatures u_j, with u_j assigned to the jth incident edge of every vertex assigned f.

Exercise: Prove that if a degenerate signature $f = u_1 \otimes \cdots \otimes u_k$ is symmetric, then it has the form $u^{\otimes k}$, for some unary $u \in \mathbb{C}^q$.

We use the term tractable as a shorthand for polynomial time computable. We allow $\mathcal{F}$ to be an infinite set. For Holant$_q(\mathcal{F})$ to be tractable, the problem must be computable in polynomial time when the input graph G and the description of the signatures appearing in G are included in the input size. In contrast, we say Holant$_q(\mathcal{F})$ is #P-hard if there exists a finite subset of $\mathcal{F}$ for which the problem is #P-hard. We say a signature set $\mathcal{F}$ is tractable (resp. #P-hard) in the context of Holant problems if the corresponding counting problem Holant$_q(\mathcal{F})$ is tractable (resp. #P-hard). Similarly for a signature f, we say f is tractable (resp. #P-hard) if $\{f\}$ is. We also speak of a signature or signature set

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

as being tractable or #P-hard for various restricted classes of Holant problems, such as those defined over planar, bipartite, or planar and bipartite graphs. The meaning of the restriction should be clear from context.

We can express $\#\mathrm{CSP}^d_q(\mathcal{F})$ as a Holant problem. An instance of $\#\mathrm{CSP}^d_q(\mathcal{F})$ has the following bipartite view. Create a node for each variable and each clause. Connect a variable node to a clause node if the variable appears in the clause. This bipartite graph is also known as the *constraint graph* . To each variable vertex, we assign the EQUALITY signature of the appropriate arity. To each clause vertex, we assign the constraint used in that clause. Under this view, we see that

$$\#\mathrm{CSP}^d_q(\mathcal{F}) \equiv_{\mathrm{T}} \mathrm{Holant}_q(\mathcal{EQ}_d \mid \mathcal{F}), \tag{1.6}$$

where $\mathcal{EQ}_d = \{=_{dk} | k \geq 1\}$ is the set of EQUALITY signatures of arities equal to a multiple of d. We denote by $\text{Pl-}\#\mathrm{CSP}^d_q(\mathcal{F})$ the restriction of $\#\mathrm{CSP}^d_q(\mathcal{F})$ to inputs with a planar constraint graph. Again we drop q and write $\#\mathrm{CSP}^d$ or $\text{Pl-}\#\mathrm{CSP}^d$ if $q = 2$, and drop d if $d = 1$. The preceding construction also shows that

$$\text{Pl-}\#\mathrm{CSP}^d_q(\mathcal{F}) \equiv_{\mathrm{T}} \text{Pl-}\mathrm{Holant}_q(\mathcal{EQ}_d \mid \mathcal{F}). \tag{1.7}$$

If $d \in \{1, 2\}$, then more is true.

Lemma 1.2. *Let $\mathcal{F}$ be a set of signatures over a domain of size q. If $d \in \{1, 2\}$, then*

$$\#\mathrm{CSP}^d_q(\mathcal{F}) \equiv_{\mathrm{T}} \mathrm{Holant}_q(\mathcal{EQ}_d \cup \mathcal{F}) \quad \text{and}$$
$$\text{Pl-}\#\mathrm{CSP}^d_q(\mathcal{F}) \equiv_{\mathrm{T}} \text{Pl-}\mathrm{Holant}_q(\mathcal{EQ}_d \cup \mathcal{F}).$$

Proof. By (1.6) and (1.7), it suffices to show

$$\mathrm{Holant}_q(\mathcal{EQ}_d \mid \mathcal{F}) \equiv_{\mathrm{T}} \mathrm{Holant}_q(\mathcal{EQ}_d \cup \mathcal{F})$$

and correspondingly for $\text{Pl-}\mathrm{Holant}_q(\mathcal{EQ}_d \mid \mathcal{F})$. It both cases, the reduction from left to right in the equivalence is trivial; just ignore the bipartite restriction. For the other direction, we take a signature grid for the problem on the right and create a bipartite signature grid for the problem on the left such that both signature grids have the same Holant value up to an easily computable factor. If the initial graph is planar, then the final graph will also be planar, so this will prove both equivalences.

If two signatures in $\mathcal{F}$ are assigned to adjacent vertices, then we subdivide all edges between them and assign the binary EQUALITY signature $(=_2) \in \mathcal{EQ}_d$ to all new vertices. Suppose EQUALITY signatures $(=_n), (=_m) \in \mathcal{EQ}_d$ are assigned to adjacent vertices connected by ℓ edges. If $n = m = \ell$, then these two vertices are disjoint from the rest, and we simply remove these two vertices. The Holant

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

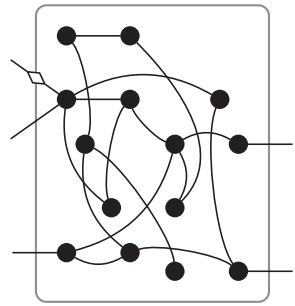


Figure 1.4. An $\mathcal{F}$-gate with five dangling edges.

of the resulting signature grid differs from the original by a factor of q. Otherwise, we contract all ℓ edges, merge the two vertices into one, and assign $(=_{n+m-2\ell}) \in \mathcal{EQ}_d$ to the new vertex. □

1.3 Reductions

We now introduce several types of reductions we will use in this book, including gadget constructions, holographic transformations, and interpolations.

1.3.1 Gadget Construction

Local Gadget Constructions via $\mathcal{F}$-gates

A basic type of reduction is generally known as a gadget construction. In the context of Holant problems, we construct a graph in order to realize a signature. Fix a set $\mathcal{F}$ of signatures over a domain $[q]$ of size q. We say a signature f is *realizable* from $\mathcal{F}$ if there is a graph with some dangling edges where each vertex is assigned a signature from $\mathcal{F}$, and the resulting signature with inputs on the dangling edges is exactly f.

Formally, this is defined in terms of an $\mathcal{F}$-gate. An $\mathcal{F}$-gate F is similar to a signature grid (G, π) for Holant($\mathcal{F}$) except that $G = (V, E, E')$ is a graph with regular edges E and some dangling edges E', and $E \cap E' = \emptyset$. Each $e' \in E'$ has only one end incident to a vertex in V, and the other end dangling. The dangling edges define external variables for the $\mathcal{F}$-gate. They are ordered, and usually presented pictorially by starting at the edge marked with a diamond and proceeding cyclically (see Figure 1.4 for an example.)

An $\mathcal{F}$-gate F with k dangling edges defines the function

$$\Gamma(y_1, \ldots, y_k) = \sum_{\sigma: E \to [q]} \prod_{v \in V} f_v(\widehat{\sigma}\ |_{E(v)}),$$

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

where $(y_1, \ldots, y_k) \in [q]^k$ is an assignment on the dangling edges, $\widehat{\sigma}$ is the extension of the assignment σ on the internal edges E by the assignment $(y_1, \ldots, y_k)$ on the dangling edges E', and f_v is the signature assigned by π at v. We call this function Γ the signature of the $\mathcal{F}$-gate. We also call an $\mathcal{F}$-gate a gadget. If the signature of an $\mathcal{F}$-gate is invariant under cyclic permutations of inputs, then we omit the diamond because it is unnecessary. We say that such signatures are *rotationally symmetric*. If the signature is invariant under any permutation of inputs, then we say it is *symmetric*. These notions are defined in terms of the signature function Γ. The signature of an $\mathcal{F}$-gate may be rotationally symmetric or symmetric as a function, even though the graph defining it may not be so.

An $\mathcal{F}$-gate is *planar* if the underlying graph can be embedded in the plane without edge crossings and the dangling edges are in the outer face in the cyclic order. Now suppose we have two signature sets $\mathcal{F}$ and $\mathcal{G}$ in the context of a bipartite Holant problem $\text{Holant}_q(\mathcal{F} \mid \mathcal{G})$. Then an $(\mathcal{F} \mid \mathcal{G})$-gate is an $(\mathcal{F} \cup \mathcal{G})$-gate such that the underlying graph is bipartite, the vertices in one part are assigned signatures from $\mathcal{F}$, and the vertices in the other part are assigned signatures from $\mathcal{G}$. Furthermore, we say that an $(\mathcal{F} \mid \mathcal{G})$-gate is *on the left* (resp. *on the right*) if each vertex incident to a dangling edge is assigned a signature from $\mathcal{F}$ (resp. $\mathcal{G}$).

Using $\mathcal{F}$-gates, we can reduce one Holant problem to another.

Lemma 1.3. *Let $\mathcal{F}$ be a set of signatures over a domain of size q. If there exists an $\mathcal{F}$-gate with signature f, then*

$$\text{Holant}_q(\mathcal{F}, f) \leq_{\text{T}} \text{Holant}_q(\mathcal{F}).$$

Similar statements hold for gadgets that are planar, bipartite, or both for Holant problems defined over the same class of graphs.

Proof. Let F be an $\mathcal{F}$-gate with signature f. Given an instance Ω of $\text{Holant}_q(\mathcal{F}, f)$, we replace every occurrence of f by the $\mathcal{F}$-gate F to obtain an instance Ω' of $\text{Holant}_q(\mathcal{F})$. Since f is the signature of the $\mathcal{F}$-gate F, the Holant values for these two signature grids are identical. Furthermore, the size of F is a constant with respect to Ω, so the size of Ω' is only a constant factor larger than that of Ω. □

Even for a very simple signature set $\mathcal{F}$, the signatures for all $\mathcal{F}$-gates can be quite complicated and expressive. In Chapter 4 we will see that for the simple class of weighted Exact-One functions, which define weighted Perfect Matchings, the resulting signature set can be quite expressive. These are called *matchgates*.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Signature Matrix

It is convenient to write a signature as a matrix. An immediate advantage is that a matrix is more of a pictorial representation than a vector, which aids understanding. However, the more important reason is to simplify the computation of the signature of a gadget.

Definition 1.4. Let f be a signature of arity n over a domain of size q. The *signature matrix of f with parameter ℓ* is an $q^{\ell} \times q^{n-\ell}$ matrix for some integer $0 \leq \ell \leq n$ in which the first ℓ inputs (in order) are the row index and the remaining $n - \ell$ inputs (in reverse order) are the column index.

If the arity of f is even, then the *signature matrix of f*, without specifying a parameter, is the signature matrix of f with parameter $\ell = \frac{n}{2}$ and is denoted by M_f.

For example, if $f = (f_{00}, f_{01}, f_{10}, f_{11})$ has arity 2 over the Boolean domain, then $M_f = \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix}$. If g is a signature of arity 4 over the Boolean domain with $g(i, j, k, \ell) = g_{ijk\ell}$, then

$$M_g = \begin{bmatrix} g_{0000} & g_{0010} & g_{0001} & g_{0011} \\ g_{0100} & g_{0110} & g_{0101} & g_{0111} \\ g_{1000} & g_{1010} & g_{1001} & g_{1011} \\ g_{1100} & g_{1110} & g_{1101} & g_{1111} \end{bmatrix}.$$

Notice the reversal of order in the column index (k, ℓ), i.e., the columns are listed in the order $k\ell = 00, 10, 01, 11$. The purpose of reversing the order of the column index is so that we can use matrix product in gadget computations. Let F be an $\mathcal{F}$-gate with signature f of arity n. We often depict F with ℓ dangling edges protruding to the left and $n - \ell$ dangling edges protruding to the right to aid in the mapping from F to the signature matrix of f with parameter ℓ. If F and F' are two $\mathcal{F}$-gates of arity 4 with signature matrices M and M', then the matrix product MM' is the signature matrix of the $\mathcal{F}$-gate linking the two $\mathcal{F}$-gates, namely by merging the fourth and third edges (in cyclic order) of F with the first and second edges (also in cyclic order) of F' respectively. In particular, if F and F' are planar $\mathcal{F}$-gates, then this linking operation produces a planar $\mathcal{F}$-gate.

1.3.2 Holographic Transformation

Sometimes two counting problems may appear different, but are really the same problem under different guises. Holographic transformation is a primary tool to establish this quantitative connection.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

It is a simple fact that in a graph, the number of vertex covers is equal to the number of independent sets. The reason is that the complement of one type is the other. At the vertex assignment level, this is accomplished by an exchange of the assignments 0 and 1. This is a type of combinatorial mapping.

In a holographic transformation the 0-1 assignments are no longer considered purely as discrete objects. Rather, they are embedded in a vector space as basis elements, and then we consider linear transformations of the ambient space. The mapping between vertex covers and independent sets can be described by a linear transformation that is simply a basis exchange $e_0 = (1,\ 0) \leftrightarrow e_1 = (0,\ 1)$. In general, a holographic transformation will be much more algebraically "mixing." Such a transformation puts the assignments of 0 and 1 into a superposition much like the states of a qubit in quantum computing. However, quantum computation is not required – or even any computation at all. Just like the example with counting vertex covers and independent sets, we are merely describing a mathematical proof that two different looking problems are actually the same. By undergoing a change of basis we can connect different problems from this perspective, and prove that they have the same (or closely related) answers.

To formally introduce the idea of holographic transformations, it is convenient to consider bipartite graphs. For a general graph, we can always transform it into a bipartite graph while preserving the Holant value as follows. For each edge in the graph, we replace it by a path of length two. (This operation is called a *2-stretch* of the graph and yields the edge–vertex incidence graph.) Each new vertex is assigned the binary EQUALITY signature ($=_2$).

For a $q \times q$ matrix T and a signature f of arity n, written as a column vector in $\mathbb{C}^{q^n}$, we write $Tf = T^{\otimes n} f$ as the transformed signature. For a signature set $\mathcal{F}$, define $T\mathcal{F} = \{Tf \mid f \in \mathcal{F}\}$. For a signature f of arity n written as a row vector, we similarly define $fT = fT^{\otimes n}$, and $\mathcal{F}T$. Here the tensor product denoted by $\otimes$ is the Kronecker product, namely, if $X = X_{a\times b}$ and $Y = Y_{c\times d}$ are two matrices, then $X \otimes Y$ is an $ac \times bd$ matrix, with entry $X_{ij}Y_{k\ell}$ at row indexed by $(i, k) \in [a] \times [c]$ and column indexed by $(j, \ell) \in [b] \times [d]$, both ordered lexicographically. Tensor power is defined inductively $X^{\otimes n} = X^{\otimes(n-1)} \otimes X$. Whenever we write $T^{\otimes n} f$ or $T\mathcal{F}$, we view the signatures as column vectors; similarly for $fT^{\otimes n}$ or $\mathcal{F}T$ as row vectors.

Let T be an invertible $q \times q$ matrix. The holographic transformation defined by T is the following operation: given a bipartite signature grid $\Omega = (H, \pi)$ of Holant$(\mathcal{F} \mid \mathcal{G})$, for the same bipartite graph H, we get a new signature grid $\Omega' = (H, \pi')$ of Holant$(\mathcal{F}T \mid T^{-1}\mathcal{G})$ by replacing each signature in $\mathcal{F}$ or $\mathcal{G}$ with the corresponding signature in $\mathcal{F}T$ or $T^{-1}\mathcal{G}$. For this reason, signatures

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

in $\mathcal{F}$ are called convariant and those in $\mathcal{G}$ contravariant. Valiant's Holant Theorem [Val08] , Theorem 1.5, states that the Holant value is unchanged, under a holographic transformation (see also [CC07]).

Theorem 1.5. *Let $\mathcal{F}$ and $\mathcal{G}$ be sets of complex-valued signatures over a domain of size q. Suppose Ω is a bipartite signature grid over $(\mathcal{F} \mid \mathcal{G})$. If $T \in \mathbf{GL}_q(\mathbb{C})$, then*

$$\text{Holant}_q(\Omega; \mathcal{F} \mid \mathcal{G}) = \text{Holant}_q(\Omega'; \mathcal{F}T \mid T^{-1}\mathcal{G}),$$

where Ω' is the corresponding signature grid over $(\mathcal{F}T \mid T^{-1}\mathcal{G})$.

Proof. We transform Ω to Ω' in several steps, preserving the Holant value in each step. The essential idea is that operator application is associative. We illustrate this in Figure 1.5.

Let $\Omega_0 = \Omega$. Let $G = (U, V, E)$ be the graph underlying Ω_0. Vertices in U are assigned signatures in $\mathcal{F}$ by π while vertices in V are assigned signatures in $\mathcal{G}$ by π. Let $e = (u, v) \in E$ be an edge with endpoints $u \in U$ and $v \in V$, assigned signatures $f \in \mathcal{F}$ and $g \in \mathcal{G}$ respectively. Initially, Figure 1.5a depicts the neighborhood of u and v in $\Omega = \Omega_0$. In this example, both u and v are incident to three other edges but the vertices incident to the other ends of these edges are not shown. We do the following operations for each edge $e = (u, v) \in E$: Subdivide e and assign $(=_2)$ to the new vertex w. Let the resulting signature grid be Ω_1 (see Figure 1.5b). It is clear that

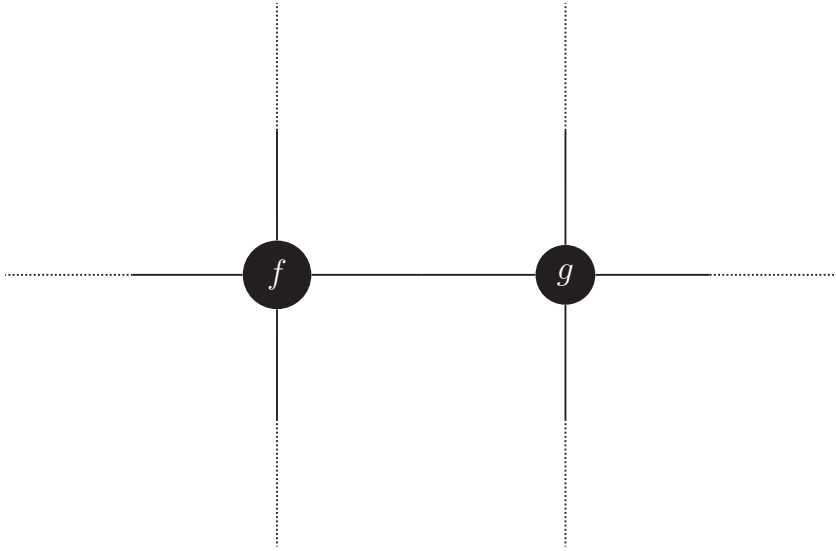
$$\text{Holant}_q(\Omega_0; \mathcal{F} \mid \mathcal{G}) = \text{Holant}_q(\Omega_1; \mathcal{F} \cup \mathcal{G} \cup \{=_2\}).$$

Then we subdivide w to get two adjacent vertices u' and v' so that we now have a path (u, u', v', v) for every edge $e = (u, v) \in E$ in G. Assign to u' and v' the binary signature (call them h_u and h_v respectively) whose signature matrix is T and T^{-1} respectively. If T is not a symmetric matrix, then these signatures are not symmetric and it matters which edge corresponds to which input. The first input for h_u, represented by the row index in T, corresponds to the edge $\{u, u'\}$. The first input for h_v, represented by the row index in T^{-1}, corresponds to the edge $\{u', v'\}$, so that it is merged with the second input for h_u, represented by the column index in T. Let the resulting signature grid be Ω_2. (See Figure 1.5c, which indicates the first inputs of h_u and h_v by the rotated placement of T and T^{-1}, with the first input to their left.) If we contract the edge $\{u', v'\}$ within the dashed box, then we get back Ω_1. Thus, the Holant value is unchanged,

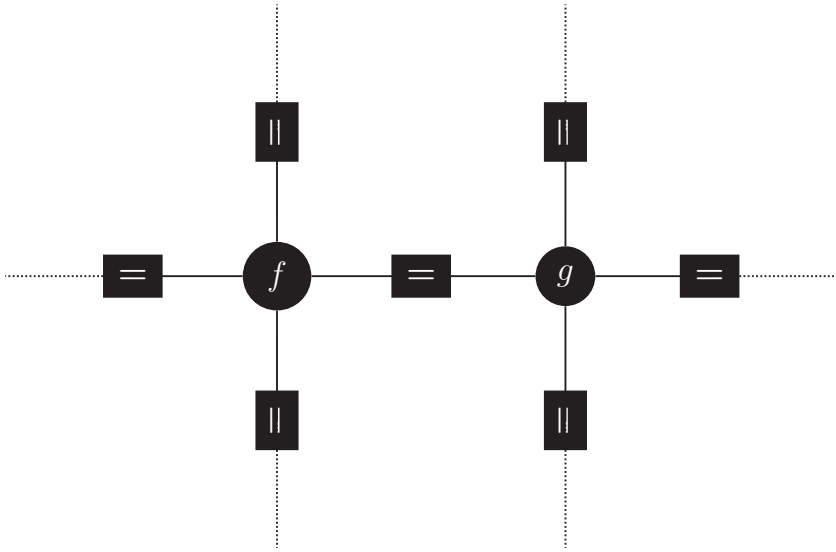
$$\text{Holant}_q(\Omega_1; \mathcal{F} \cup \mathcal{G} \cup \{=_2\}) = \text{Holant}_q(\Omega_2; \mathcal{F} \cup \mathcal{G} \cup \{T, T^{-1}\}).$$

Now Ω_3 in Figure 1.5d is really the same as Ω_2, except we think of the order of contraction operations differently by associating the binary T and T^{-1} to the

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press



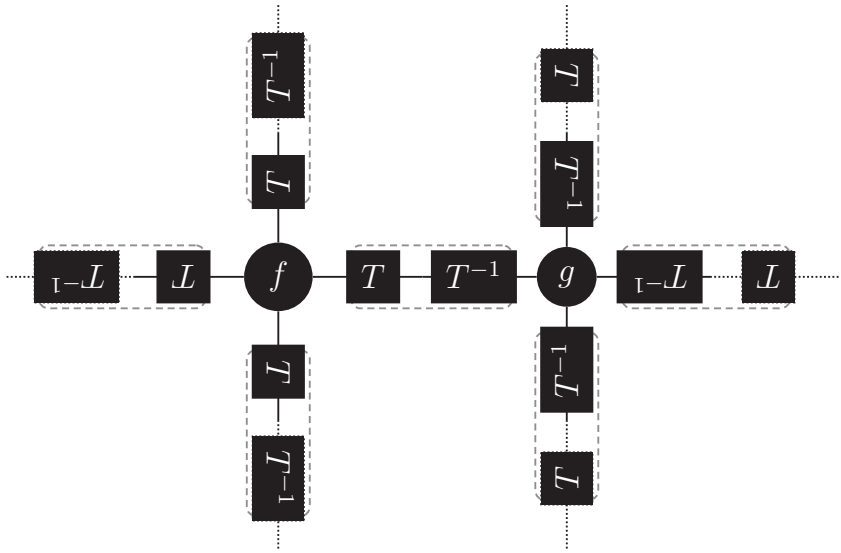
(a) $\Omega = \Omega_0$



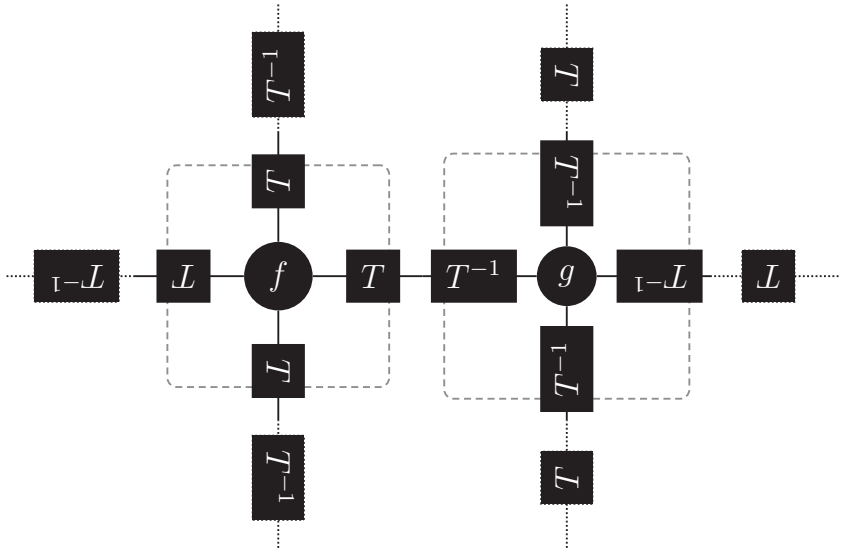
(b) Ω_1

Figure 1.5. Neighborhood around two adjacent vertices.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press



(c) Ω_2



(d) Ω_3

Figure 1.5. (*cont.*)

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

original signatures f and g first.

$$\text{Holant}_q(\Omega_2; \mathcal{F} \cup \mathcal{G} \cup \{T, T^{-1}\}) = \text{Holant}_q(\Omega_3; \mathcal{F} \cup \mathcal{G} \cup \{T, T^{-1}\})$$

Finally we contract $\{u, u'\}$ and $\{v', v\}$. This defines $\Omega' = \Omega_4$. After doing so, we once again have the bipartite graph G. What has changed is the assignment to each vertex. Any vertex $u \in U$ of degree $\deg(u)$ assigned $f \in \mathcal{F}$ in Ω is now assigned $fT^{\otimes \deg(u)}$. Similarly any vertex $v \in V$ of degree $\deg(v)$ assigned $g \in \mathcal{G}$ in Ω is now assigned $(T^{-1})^{\otimes \deg(v)}g$. In general, the new assignment to each vertex is the transformed signature in $\mathcal{F}T$ or $T^{-1}\mathcal{G}$ respectively, as claimed. Hence,

$$\text{Holant}_q(\Omega_3; \mathcal{F} \cup \mathcal{G} \cup \{T, T^{-1}\}) = \text{Holant}_q(\Omega_4; \mathcal{F}T \mid T^{-1}\mathcal{G}). \qquad \square$$

Therefore, an invertible holographic transformation does not change the complexity of the Holant problem in the bipartite setting. Furthermore, there is a special kind of holographic transformation, the orthogonal transformation, that preserves the binary equality and thus can be used freely in the standard (non-bipartite) setting. We denote by $\mathbf{O}_q(\mathbb{C})$ the group of $q \times q$ orthogonal matrices over $\mathbb{C}$. Note that $T \in \mathbf{O}_q(\mathbb{C})$ iff $T^{-1}T = I_q$ iff the holographic transformation $(=_2) \mapsto (=_2)T^{\otimes 2} = (=_2)$. The latter identity can be checked by its matrix form, because for a binary signature f, if M is its matrix form, then $T^{\mathrm{T}}MT$ is the matrix form of the binary signature $fT^{\otimes 2}$.

Exercise: Verify that for a binary covariant signature (row vector) f with signature matrix M, the signature matrix of $fT^{\otimes 2}$ is $T^{\mathrm{T}}MT$. If f is a binary contravariant signature (column vector) with signature matrix M, then the signature matrix of $(T^{-1})^{\otimes 2}f$ is $T^{-1}M(T^{-1})^{\mathrm{T}}$. Generalize this to signatures of higher arity.

Theorem 1.6. *Let $\mathcal{F}$ be a set of complex-valued signatures over a domain of size q. Suppose Ω is a signature grid over $\mathcal{F}$. If $H \in \mathbf{O}_q(\mathbb{C})$, then*

$$\text{Holant}_q(\Omega; \mathcal{F}) = \text{Holant}_q(\Omega'; H\mathcal{F}),$$

where Ω' is the corresponding signature grid over $H\mathcal{F}$.

Proof. $\text{Holant}_q(\Omega; \mathcal{F}) = \text{Holant}_q(\Omega; (=_2) \mid \mathcal{F})$. $\square$

We use Theorems 1.5 and 1.6 to reduce both between tractable problems and between #P-hard problems. Some of our reductions between tractable problems use the following definition, where $\mathscr{C}$ is usually some known tractable class of signatures.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Definition 1.7. Let $\mathcal{F}$ be any set of complex-valued signatures over a domain of size q. We say $\mathcal{F}$ is $\mathscr{C}$-transformable if there exists a $T \in \mathbf{GL}_q(\mathbb{C})$ such that $(=_2)T^{\otimes 2} \in \mathscr{C}$ and $\mathcal{F} \subseteq T\mathscr{C}$.

Note that if $\text{Holant}_q(\mathscr{C})$ is tractable over any set of graphs, then $\text{Holant}_q(\mathcal{F})$ is tractable over the same set of graphs for any $\mathscr{C}$-transformable set $\mathcal{F}$.

Lemma 1.8. *Let $\mathcal{F}$ be any set of complex-valued signatures over a domain of size q. If $\mathcal{F}$ is $\mathscr{C}$-transformable, then*

$$\text{Holant}_q(\mathcal{F}) \leq_{\text{T}} \text{Holant}_q(\mathscr{C}),$$

where both problems are defined over the same set of graphs.

Another Identity

Holographic transformations between two signature grids give the same Holant value. The following lemma shows a different kind of reason why two Holant problems must always have the same Holant value.

Lemma 1.9. *Let x and y be any numbers in $\mathbb{C}$. Then for every signature grid Ω having a $(2, 4)$-regular bipartite underlying graph,*

$$\text{Holant}(\Omega; [0, 1, 0] \mid [x, y, 1, 0, 0]) = \text{Holant}(\Omega; [0, 1, 0] \mid [0, 0, 1, 0, 0]).$$

Proof. In any signature grid Ω having a $(2, 4)$-regular bipartite underlying graph, because $[0, 1, 0]$ is the only signature on the left, any nonzero term in the Holant sum must assign 1 to exactly half of the edges in G. On the right side, if some copy of $[x, y, 1, 0, 0]$ contributes an x or y in some assignment, then fewer than half of its incident edges are assigned 1. To compensate, some other copy of $[x, y, 1, 0, 0]$ must have more than half of its incident edges assigned 0, so it contributes a factor 0. □

Exercise: Prove that for all $x, y \in \mathbb{C}$ not both 0, there is no holographic transformation between the two Holant problems in Lemma 1.9. Thus the converse of Theorem 1.6 does not hold.

1.3.3 Polynomial Interpolation

Polynomial interpolation is a powerful technique to prove #P-hardness for counting problems. We can introduce this technique with an example from [Val79a] by Valiant.

Lemma 1.10. #PerfectMatching $\leq_{\text{T}}$ #Matching

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

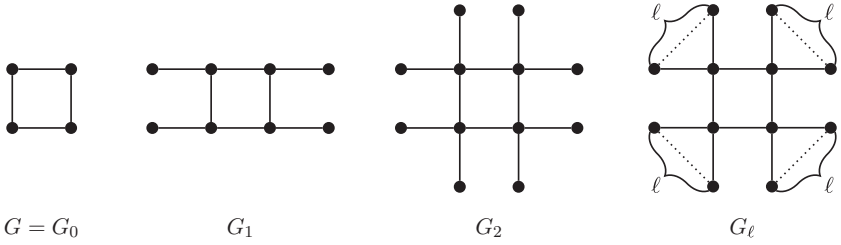


Figure 1.6. Some graphs obtained from an initial graph G in the proof of Lemma 1.10.

Proof. Let $G = (V, E)$ be a graph. We want to determine the number of perfect matchings in G assuming that we have an oracle to count (all, not necessarily perfect) matchings.

For every integer $0 \le \ell \le n$, we construct a graph G_ℓ from G in the following way. For each vertex $v \in V$ we will "grow a thistle," namely, we add ℓ new vertices v_k $(1 \le k \le \ell)$, each connected to v by one edge. See Figure 1.6 for some examples of these graphs beginning with a specific graph G. In G_ℓ each v_k has degree 1 and clearly it can only be matched to v. If v is matched to some v_k $(1 \le k \le \ell)$, then v cannot be matched to any other v_ℓ for $\ell \ne k$ nor can it be matched to any other original vertex of G.

Let m_k be the number of matchings in G that omit k vertices. Then we can express the number of matchings in G_ℓ as

$$\sum_{k=0}^{n} (\ell + 1)^k m_k = \#\textsc{Matching}(G_\ell). \tag{1.8}$$

This is because a matching of G that omits exactly k vertices can be extended to exactly $(\ell + 1)^k$ matchings in G_ℓ, and every matching in G_ℓ is obtained uniquely this way from a matching of G. To see this, let M be a matching of G that omits k vertices. Then each of the k vertices in G that is not matched by M has $\ell + 1$ possibilities with respect to a matching M' in G_ℓ extending M, because it can can be matched with one of the ℓ new neighbors in G_ℓ or remain unmatched. Conversely, for every matching M' of G_ℓ, we can consider its subset consisting of edges in G. This defines a matching M in G and M' is one of the extensions from M, and such an M is unique.

We collect these equations to form the linear system

$$\begin{bmatrix} (0+1)^0 & (0+1)^1 & \cdots & (0+1)^n \\ (1+1)^0 & (1+1)^1 & \cdots & (1+1)^n \\ \vdots & \vdots & \ddots & \vdots \\ (n+1)^0 & (n+1)^1 & \cdots & (n+1)^n \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} \#\textsc{Matching}(G_0) \\ \#\textsc{Matching}(G_1) \\ \vdots \\ \#\textsc{Matching}(G_n) \end{bmatrix}. \tag{1.9}$$

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

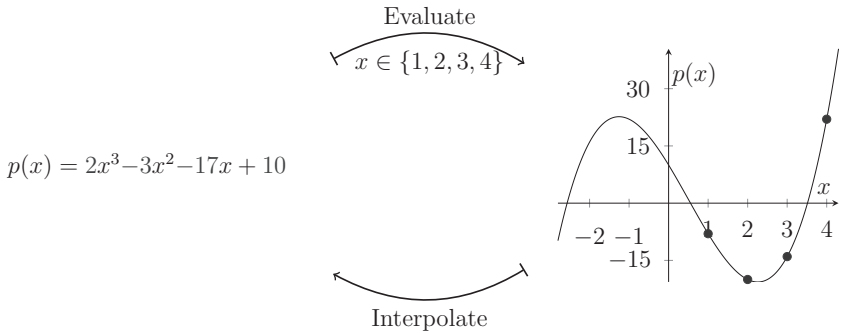


Figure 1.7. Interpolation is the inverse of evaluation.

Using our oracle, we know the right side. On the left, the coefficient matrix is Vandermonde . It is invertible because the values $\ell + 1$, for $0 \leq \ell \leq n$, are all distinct. Therefore, we can invert this matrix and solve for the unknown m_k's in polynomial time. Then m_0, the number of matchings in G that omit no vertices, is the number of perfect matchings G as desired. □

The word "polynomial" did not appear in this proof, so what makes it an example of polynomial interpolation? The polynomial is implicit; it is $p(x) = \sum_{k=0}^{n} m_k x^k$. Asking our oracle for the number of matchings in G_ℓ is equivalent to evaluating $p(x)$ at $\ell + 1$. Of course this is made possible by the design of the gadget of the "thistle," a weed-like structure, which produced such an expression for #MATCHING(G_ℓ). Such designs are where the inventiveness is in an interpolation proof. Polynomial interpolation is the process of recovering the coefficients of a polynomial of known degree from its evaluations at various points (see Figure 1.7), which is what this proof did.

Since our $n + 1$ evaluation points are distinct, we can recover the coefficients of $p(x)$. In this simple interpolation proof, the coefficient matrix is clearly invertible. For more complicated proofs this can be more demanding mathematically, and of course this works hand-in-hand with the inventive process of designing the right gadget. Given these coefficients, our reduction can proceed by computing any polynomial-time computable function from these coefficients. It is also often the case that we are interested in some evaluation of the interpolated polynomial. In the proof of Lemma 1.10, the answer we seek is the evaluation of the polynomial at a particular point 0, obtaining $p(0) = m_0$.

In fact both problems #MATCHING and #PERFECTMATCHING are #P-complete. The reverse reduction, from the problem of counting all matchings #MATCHING to the problem of counting perfect matchings #PERFECTMATCHING, is given in Lemma 1.11. In contrast, this proof is not by interpolation.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Lemma 1.11. #MATCHING $\leq_T$ #PERFECTMATCHING

Proof. Let $G = (V, E)$ be a graph. We want to determine the number of all, not necessarily perfect, matchings in G assuming that we have an oracle to count perfect matchings. Suppose G has n vertices $V = \{v_i \mid 1 \leq i \leq n\}$. For every k such that $0 \leq k \leq n/2$, consider the graph $G_k = (V', E')$, where $V' = V \cup \{z_j \mid 1 \leq j \leq n - 2k\}$, a disjoint union of V and $n - 2k$ new vertices, and E' is the disjoint union of E and $n(n - 2k)$ new edges consisting of all pairs $\{(v_i, z_j) \mid 1 \leq i \leq n, 1 \leq j \leq n - 2k\}$, i.e., a copy of the complete bipartite graph $K_{n,n-2k}$. Now it is easy to see that every matching in G of size k has exactly $(n - 2k)!$ extensions to a perfect matching in G_k, matching the remaining $n - 2k$ vertices of V to the new vertices in V'. Conversely, every perfect matching in G_k is obtained uniquely in this way. Thus by computing #PERFECTMATCHING on G_k we obtain the number of matchings in G of size k, and summing over all $0 \leq k \leq n/2$ gives #MATCHING on G. □

We note that the proof in Lemma 1.10 is planar in the sense that if the given G is a planar graph then all graphs G_ℓ constructed from G are also planar graphs. By contrast, the proof in Lemma 1.11 is not planar in the sense that starting from a planar graph G the graphs G_k constructed from G in the proof are in general not planar. There is an intrinsic reason for this discrepancy in the proof. It turns out that #MATCHING is #P-complete even when restricted on planar graphs, while #PERFECTMATCHING is only #P-complete for general graphs; we will see in Chapter 4 that #PERFECTMATCHING can be computed in polynomial time over planar graphs. Consequently, any proof of Lemma 1.11 that is planar in the aforementioned sense would imply a collapse of #P to P. Since collapsing #P to P trivially implies P = NP, this is a more severe collapse than P = NP. This eventuality is considered most unlikely, and ultimately the dichotomy theorems in this book are meaningful only assuming #P does not collapse to P, although the statements and proofs of these theorems do not make this assumption. One of the themes in this book is also to obtain complexity classifications of counting problems, which further delineates the boundary between planar and nonplanar graphs, and the algorithm for #PERFECTMATCHING over planar graphs plays an important role.

The heart of polynomial interpolation as a reduction technique is finding an equation system like (1.9). For the ℓth equation, on the left we have a combination involving some partial sums of the original problem on G. On the right, we have the evaluation of the Holant problem on the constructed graph G_ℓ. The art is to pick G_ℓ so that a relation like (1.8) holds. It is important that the original sum on G is partitioned into at most a polynomial number of partial sums. In this example (1.8) all exponentially many configurations (up to $\binom{n}{k}$)

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

of matchings in G that omit exactly k vertices are combined, with the common property that each of them has exactly $(\ell+1)^k$ extensions to a matching in G_ℓ. The construction defines an infinite family $\{G_\ell\}$, and the size of graphs in this family should not grow faster than a polynomial in G and ℓ, so that we can hope to construct polynomially many members in $\{G_\ell\}$ in polynomial time. Finally, they should indeed be constructible individually in polynomial time and some effective relationship such as (1.8) can be established.

Now we consider an example in which the polynomial is quite explicit. For any natural number $k \in \mathbb{N}$ let $\chi(G; k)$ be the number of proper vertex colorings of G using at most k colors, i.e., a labeling of the vertices of G by $[k]$ such that no two neighboring vertices receive the same label. If x, y are two vertices of G, then $G\backslash xy$ (deletion) is the graph obtained from G by deleting all edges between x and y, and G/xy (contraction) is the graph obtained from G by contracting x and y, namely, x and y are merged to form a single vertex and all edges between them are removed. Then

$$\chi(G; k) = \chi(G\backslash xy; k) - \chi(G/xy; k).$$

Hence, by induction, $\chi(G, k)$ is a polynomial in k of degree n, the number of vertices of G. The chromatic polynomial, denoted by $\chi(G; \lambda)$, is the unique polynomial in λ, such that for natural numbers $k \in \mathbb{N}$, $\chi(G; k)$ is the number of proper vertex colorings of G.

The following example of polynomial interpolation is a dichotomy theorem for the chromatic polynomial. The reduction we use comes from Linial [Lin86] and the dichotomy was first explicitly stated in [JVW90]. Let $\chi(\lambda)$ be the problem of evaluating $\chi(G; \lambda)$ on an input graph G.

Lemma 1.12. *Let $\lambda \in \mathbb{C}$. Then $\chi(\lambda)$ is #P-hard if $\lambda \notin \{0, 1, 2\}$. The problem is computable in polynomial time if $\lambda \in \{0, 1, 2\}$.*

Proof. If $\lambda = 0$, then $\chi(G, \lambda) = 0^0 = 1$ if G has no vertices and is 0 otherwise. If $\lambda = 1$, then $\chi(G, \lambda) = 1$ if G has no edges and is 0 otherwise. If $\lambda = 2$, then $\chi(G, \lambda) = 2^k$ if G is bipartite with k connected components and is 0 otherwise.

Now suppose $\lambda \notin \{0, 1, 2\}$. We reduce from $\chi(3)$, which is known to be #P-hard (see, e.g., [Lin86, Main Theorem, Case (6)] or [Bar04, Proposition 5]). Let G be a graph with n vertices, and let K_t be the complete graph on t vertices. We use $G + K_t$ to denote the graph obtained from G by adding K_t and all possible edges between the vertices of G and the vertices of K_t. Then clearly

$$\chi(G + K_t; \lambda) = \lambda(\lambda - 1) \cdots (\lambda - t + 1)\chi(G; \lambda - t).$$

when $\lambda \in \mathbb{N}$. Thus, it must also hold as an equation of polynomials when considering λ as an indeterminate. Thus for λ not an integer between 0 and $t - 1$,

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

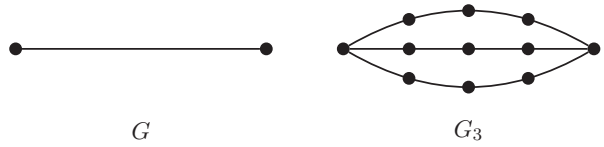


Figure 1.8. Example construction from the proof of Lemma 1.13 with $\ell = 3$.

we have

$$\chi(G;\lambda - t) = \frac{1}{\lambda(\lambda-1)\cdots(\lambda-t+1)}\chi(G + K_t;\lambda). \tag{1.10}$$

If $\lambda \geq 3$ is an integer, then by setting $t = \lambda - 3$, we can directly solve for $\chi(G;3)$ via (1.10). Otherwise, $\lambda \notin \mathbb{N}$, since $\lambda \neq 0, 1, 2$. Then choosing $0 \leq t \leq n$, using (1.10), we can compute $\chi(G;\lambda - t)$ by evaluating $\chi(G + K_t;\lambda)$ at the given λ for a sequence of graphs $G + K_t$. From these evaluations, we can interpolate the coefficients of $\chi(G;\lambda)$ and evaluate it at $\lambda = 3$. □

The proofs of both Lemmas 1.10 and 1.12 involve interpolation of a single variable polynomial. Some reductions between counting problems are accomplished via interpolation of multivariate polynomials. An early example of this occurs in [PB83, Main Theorem, Case 1], which interpolates a homogeneous polynomial in three variables.

Lemma 1.13. #VertexCover $\leq_T$ #BipartiteVertexCover

Proof. Given a graph G with n vertices and m edges, we create a graph G_ℓ for every $1 \leq \ell \leq N = \binom{m+2}{2}$ in two steps as follows. First we replace every edge of G with ℓ parallel edges to obtain a graph G'_ℓ; this operation is called an ℓ-thickening on G. Then we replace every edge of G'_ℓ with a path of length 4; this operation is called a 4-stretching on G'_ℓ. Since this stretch is by an even length 4, the resulting graph is bipartite. See Figure 1.8 for an example with $\ell = 3$.

We stratify all subsets S of V according to a detailed account on how many edges in G are not covered by S, how many edges are covered by exactly one vertex in S, and how many are covered by two vertices in S. Specifically, let c_{ijk}, where $i, j, k \geq 0$ and $i + j + k = m$, be the number of $S \subseteq V$ such that exactly

- i edges in G have neither endpoint in S,
- j edges in G have exactly one endpoint in S, and
- k edges in G have both endpoints in S.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Now define

$$p(x, y, z) = \sum_{\substack{i+j+k=m \\ i,j,k\geq 0}} x^i y^j z^k c_{ijk}.$$

Then it can be seen that #VertexCover$(G) = p(0, 1, 1)$.

Let v_1, v_2, v_3, v_4 and v_5 be the five vertices along a path of length 4. It is a direct verification that there are the following possibilities for vertex covers according to whether exactly zero, one, or two endpoints $\{v_1, v_5\}$ are included in a vertex cover of the path. Indeed

1. There are exactly *two* vertex covers when neither endpoint is in the vertex cover, because in this case both v_2 and v_4 must be in the vertex cover, and v_3 can be arbitrarily placed.
2. There are exactly *three* vertex covers when exactly one endpoint, say v_5, is in the vertex cover, because in this case v_2 and at least one of v_3 and v_4 must be in the vertex cover.
3. There are exactly *five* vertex covers when both endpoints are in the vertex cover, because in this case either v_3 is in the vertex cover and then v_2 and v_4 can be arbitrarily placed, or v_3 is not in the vertex cover and then both v_2 and v_4 must be in the vertex cover.

Thus,

$$\#\text{VertexCover}(G_\ell) = p(2^\ell, 3^\ell, 5^\ell) = \sum_{\substack{i+j+k=m \\ i,j,k\geq 0}} (2^i 3^j 5^k)^\ell c_{ijk}. \quad (1.11)$$

This defines a Vandermonde system that has full rank iff $2^i 3^j 5^k \neq 2^{i'} 3^{j'} 5^{k'}$ for all distinct pairs $(i, j, k) \neq (i', j', k')$. This is clearly valid by unique prime factorization. Therefore, we can interpolate $p(x, y, z)$ in polynomial time, and then evaluate it at $p(0, 1, 1)$ to obtain the number of vertex covers of G. □

In most interpolation proofs a crucial point is the following. There is a suitable stratification of the Holant sum on the original instance with at most polynomially many different types (in Lemma 1.13 it is $N = \binom{m+2}{2}$). This is usually defined by specifying the number of local configurations induced by a global object (in Lemma 1.13 it is specified by the tuple $\langle i, j, k\rangle$.) Given a type of local configuration, we can find the factor associated with it in the Holant sum on the constructed instance, and this is independent of the exact distribution of these local configurations. Here the fact that we are dealing with sum-of-product computations is the salient reason.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

Remark: Let P_n denote the path graph of length n, with vertices $v_0, v_1, \ldots, v_n$. We showed in the proof that the vertex covers of P_4 can be partitioned into three sets represented by the triple $(2, 3, 5)$. A more general relation for P_n is the following recurrence relation

$$\#\textsc{VertexCover}(P_n) = \#\textsc{VertexCover}(P_{n-1}) + \#\textsc{VertexCover}(P_{n-2}).$$

This is the same recurrence relation as the Fibonacci numbers. The recurrence is to partition the vertex covers of P_n according to whether it contains the endpoint v_n or not. The derivation in the proof of Lemma 1.13 corresponds to the following partition: (1) Both v_0 and v_n are not in the vertex cover, in which case v_1 and v_{n-1} must be chosen and then it is the same as vertex covers for P_{n-4}. (2) Exactly one of v_0 and v_n is in the vertex cover, say v_n, in which case v_1 must be chosen and then it is the same as vertex covers for P_{n-3}. However, in the account for $\#\textsc{VertexCover}(P_n)$ this case should be counted twice. (3) Both v_0 and v_n are in the vertex cover in which case it is the same as vertex covers for P_{n-2}. Thus this recurrence is $x_n = x_{n-4} + 2x_{n-3} + x_{n-2}$ which is consistent with the more succinct Fibonacci recurrence relation. However, in the proof of Lemma 1.13 we need to stratify it this way because that is how the common factors $(2^i 3^j 5^k)^\ell$ can be extracted in the expression $p(2^\ell, 3^\ell, 5^\ell)$.

For homogeneous polynomials of degree d in n variables, the number of monomials is $\binom{d+n-1}{n-1}$ and thus the minimum number of points that can interpolate such polynomials is $\binom{d+n-1}{n-1}$, which is polynomial in n for a fixed d. However, often the matrix may not be of full rank. In those case we will have to work harder to prove that interpolation succeeds.

1.4 Further Discussion on Models of Computation

In fact, for the computational problems we will consider in this book, every single problem involves only a fixed number of complex numbers as given parameters $\alpha_1, \alpha_2, \ldots, \alpha_k$, and all other complex numbers involved in the problem belong to the algebraic extension field $\mathbf{F} = \mathbb{Q}(\alpha_1, \alpha_2, \ldots, \alpha_k)$. Thus these parameters $\alpha_1, \alpha_2, \ldots, \alpha_k$ are part of the specification of the problem, and should not be confused with the computational complexity, which is measured by the size of the problem instance.

Because these parameters are part of the problem specification, one may take the following formalistic view. It is known that every finite extension field $\mathbf{F}$ over the rational numbers $\mathbb{Q}$ has the following structural form [Jac85b (Theorem 8.35 on p. 512)]: $\mathbf{F}$ is a finite algebraic extension $\mathbf{E}(\beta_1, \beta_2, \ldots, \beta_\ell)$ of a certain purely transcendental extension field $\mathbf{E}$ over $\mathbb{Q}$, having the form

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

$\mathbf{E} = \mathbb{Q}(X_1, \ldots, X_m)$ where $X_1, \ldots, X_m$ are algebraically independent. $\mathbf{F}$ is said to have a finite transcendence degree m over $\mathbb{Q}$. It is known that $\mathbf{F}$ has a unique and well-defined transcendence degree over $\mathbb{Q}$. Then the formalistic view demands that $\mathbf{F}$ be specified as such (as part of the task of the one who specifies the problem). The β_j's are specified by irreducible polynomials over $\mathbf{E}$ as usual.

The advantage of this totally general view is that we can then consider the theorems proved in this book logically cover all $\mathbb{C}$, and not restricted to algebraic numbers. Thus, numbers such as e or π need not be excluded on the account of models of computation. However, in realistic terms, this approach is not advisable. There is too much unknown about transcendental numbers; e.g., it is still unknown whether $e + \pi$ or $e\pi$ are rational, algebraic irrational, or transcendental, and it is open whether $\mathbb{Q}(e, \pi)$ has transcendence degree 2 (or 1) over $\mathbb{Q}$, i.e., whether e and π are algebraically independent (although Nesterenko proved that π and e^{π} are algebraically independent [Nes96a, Nes96b]). By placing the foundation of this theory concerning models of computation on the existence of transcendence degree (which is proved nonconstructively), while it may appear to be elegant and sweeping, will obscure the real issue of computational complexity to be addressed in this book. Thus we choose to formally restrict ourselves to algebraic numbers.

However, owing to the mathematical inner connections and the proof techniques to be developed in this book, it is *inadvisable* to restrict ourselves to, say, integers, or $\{0, 1\}$ entries. By allowing complex numbers we discover structures that would be invisible if we were to artificially restrict ourselves to integers, or even real numbers. Moreover, the scope of algebraic numbers is adequate to develop this theory. This is the reason we choose algebraic numbers for our models of computation.

Because our computations mostly concern algebraic operations such as sum and product, the consideration of models of computation appears to be not crucial. For brevity and convenience, we will simply use $\mathbb{C}$ to refer to complex algebraic numbers in $\mathbb{C}$, and simply call them complex numbers.

1.5 An Outline of This Book

Here we give an outline of this book.

The book is divided into two volumes. Volume I of this book is about the dichotomy theory on Boolean domain problems. A characteristic feature for the theory on Boolean domain problems is that we can prove quite explicit and highly effective complexity dichotomies, in all three frameworks of spin systems, #CSP, and Holant problems. Another important feature is that we can

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

give quite a complete classification on problems which are #P-hard in general but polynomial time tractable over planar instances. Volume II of the book will deal with counting problems over general domains. Two main theorems that will be presented are the complexity dichotomy for graph homomorphisms over $\mathbb{C}$ and for #CSP for arbitrary constraint functions taking values in $\mathbb{C}$.

For Volume I, after this introductory Chapter 1, we commence the development of our theory in Chapter 2, where we introduce Holant* problems, a restricted class of counting problems in the Holant framework where all unary (or equivalently all degenerate) functions are assumed to be freely available. We will encounter Fibonacci gates, and show that they constitute a main part of the tractable signatures in Holant* problems. In later chapters it will be seen that Fibonacci gates mainly correspond to what is transformable to the so-called product type constraint functions. The main theorem in Chapter 2 is a complexity dichotomy for Holant* problems for any set of symmetric signatures. One reason we start the presentation of this theory with Holant* problems is that here the theory is relatively simple, very concrete, and yet nontrivial. Also we can illustrate many proof ideas, such as holographic transformations in a simple setting, which will be further developed in later chapters. Another reason is that to be able to obtain all unary functions, either by direct construction or by simulation, is a main theme of proving #P-hardness in later chapters.

In Chapter 3, we consider a second framework, #CSP over Boolean domains, and prove a complexity dichotomy for #CSP for any set of constraint functions, not necessarily symmetric. Here we introduce affine signatures. Together with signatures of product type, these constitute the tractable constraints for Boolean #CSP. The important proof technique – interpolation – is employed here. The dichotomy of Chapter 3 is the special case of a much more general dichotomy of #CSP (in Volume II) restricted to the Boolean domain. Some preparatory theorems can also be derived in a more general setting. However, we choose to give a proof in the Boolean domain independently from the general theory because this proof is elementary, and the resulting dichotomy is explicit. The idea of simulating unary signatures in #P-hardness proofs also appears here, which utilizes the dichotomy from Chapter 2 both explicitly as well as in spirit. It is proved that in Boolean #CSP, one can always obtain two special unary functions, which will define a weaker type of Holant* problems, and will be called Holant^c problems.

Chapter 4 starts with the theory of Pfaffian orientations and Kasteleyn's algorithm, also known as the Fisher–Kasteleyn–Temperley algorithm, or the FKT algorithm, which can count perfect matchings on planar graphs. Then the chapter gives a thorough treatment of the theory of matchgates. It is with matchgates that holographic reductions were first introduced, although in this book

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

we use it already to prove dichotomy theorems in Chapter 2. These matchgates, and together with holographic transformations, are important ingredients in those counting problems that are #P-hard in general but computable in polynomial time for planar graphs. The theory for general matchgates depends on matchgate identities. But for most of Volume I, dichotomy theorems concerning matchgates only use symmetric signatures; for these it is logically possible to skip matchgate identities and only use Theorem 4.11. One can derive the form of symmetric matchgate signatures as given in Theorem 4.11, by the necessary parity constraints and the explicit construction in Section 4.3.3.

Chapter 5 proves a dichotomy for spin systems on regular graphs. A number of proof techniques are introduced in this chapter, which are summarized at the beginning of the chapter. This chapter proves the dichotomy for the case of k-regular graphs, for $k = 3$ and $k = 4$. The more general case for all k is also true. However, for the purpose of the main dichotomy for Holant problems, we will only need the case $k = 3$ and 4. This chapter consciously develops the material in a parallel fashion, by introducing technical tools gradually throughout while moving toward the proof of the desired dichotomy theorem. The statement of this dichotomy for 3-regular and 4-regular graphs will be a key result for later dichotomies.

Chapter 6 is a transitional chapter, serving many purposes. It connects Holant problems and #CSP. It also introduces the transformational perspective of looking at dichotomy theorems so that now not only are holographic transformations used as proof techniques, but also they become the language in which the statement of a dichotomy theorem is expressed. In doing so, naturally we introduce group actions. This chapter also proves a dichotomy for Holant^c problems and for CSP^d; it also proves Eulerian Orientation is #P-hard. It concludes with a section on Kirchhoff's Matrix Tree theorem and some related results.

Chapter 7 proves a main dichotomy theorem for Volume I, toward which much of the material in previous chapters are preparatory. The dichotomy is proved for any set of symmetric signatures on the Boolean domain without assuming any auxiliary functions. An important new ingredient is the isolation of the vanishing signatures.

Chapter 8 proves a dichotomy for planar #CSP for any set of symmetric signatures on the Boolean domain. It shows that for the #CSP framework, a holographic reduction to Kasteleyn's algorithm, implemented by matchgates, is a universal strategy for those problems that are #P-hard but tractable over planar graphs.

Chapter 9 presents a classification theorem for planar Holant problems for any set of symmetric signatures on the Boolean domain. It turns out that for planar Holant problems there are additional problems that are #P-hard but

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

tractable over planar graphs. These cannot be captured by holographic reductions to Kasteleyn's algorithm. However, together these two strategies are universal.

Chapter 10 presents additional dichotomy theorems for sets of asymmetric signatures. Most of the dichotomy theorems presented so far are for symmetric constraint function sets, except for #CSP in Chapter 3. In this final chapter of Volume I, we first present a classification for planar #CSP problems for all constraint functions that are not necessarily symmetric, over the Boolean domain. Then we revisit Holant* problems of Chapter 2, and prove a Holant* dichotomy for not necessarily symmetric signatures, over the Boolean domain. In doing so, we also gain some new perspective on the nature of the tractable signatures sets.

Volume II presents the theory on the general domain.

1.5.1 Logical Dependence and Omitted Proofs

The logical dependence of the results in Volume I of this book basically follow the order of the chapters. After the preliminary Chapter 1, we present the first nontrivial complexity dichotomy theorem in Chapter 2 for Holant* problems with symmetric constraint functions. Chapter 3 has minimal dependence of Chapter 2 for the proof of the dichotomy theorem for Boolean #CSP. This dependence can be removed with some alternative proof, but we feel that it is a good idea to highlight the general principle early on that the ability to interpolate all unary signatures is a powerful step toward a dichotomy. Then Chapter 4 introduces a new theme, that of counting problems over planar structures. This is the setting where holographic algorithms were originally discovered by Valiant, even though in our presentation we introduce holographic reductions first with Fibonacci gates, not with matchgates. All results in subsequent chapters that concern planar cases depend on the theory presented in Chapter 4. Theorems or parts of theorems that do not have planarity restrictions are independent of Chapter 4. Furthermore, for planar counting problems specified by symmetric local constraint functions, the logical dependence on Chapter 4 can be considerably reduced as indicated above. However, a full appreciation of the theory of what can and cannot be computed efficiently on planar structures cannot be gained with this abridged version.

Chapter 5 presents dichotomy theorems for spin systems on regular graphs. In Chapter 5 we present the proof for k-regular graphs only for $k = 3$ and $k = 4$. The reason is two folded: (1) We have organized the subsequent chapters so that logically the theorem with $k = 3$ and 4 is sufficient for subsequent theorems. (2) The technical details of this chapter are already quite demanding. By restricting to the cases $k = 3$ and $k = 4$ we can avoid some technical complications, while

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press

highlighting the essential techniques of this chapter in a more transparent setting. Chapter 6 is a transitional chapter that serves many purposes. In terms of logical dependence, it mainly introduces the transformational perspective, the Eulerian Orientation problem, and the Tutte polynomial, and proves the CSP^d dichotomy for symmetric constraints. After Chapters 5 and 6 that prepared the ground, Chapter 7 presents a dichotomy for Holant problems for symmetric constraints. This proof is quite long; and we have decided to omit the proof of Theorem 7.53, on the complexity of deciding whether a signature set satisfies the dichotomy criteria. This theorem is not needed logically for subsequent chapters. Chapter 8 presents the classification of planar Boolean #CSP for symmetric constraints. The full proof of this classification is included. Chapter 9 presents the corresponding classification of planar Holant problems over the Boolean domain for symmetric constraints. The surprise in Chapter 9 is that unlike planar #CSP problems, planar Holant problems have additional tractable problems besides those that those that can be transformed to Kasteleyn's algorithm. Here, because of the length of the proof of this theorem (Theorem 9.29) we omit most of the proof, but focus on the new tractable problems.

Finally, Chapter 10 presents the few results we currently know about asymmetric constraint functions. For planar Boolean #CSP problems, we present Theorem 10.1. This theorem generalizes the classification in Chapter 8 to all, not necessarily symmetric, constraints. This is also a generalization of the Boolean #CSP dichotomy in Chapter 3. But we omit most of its proof due to its length, merely stating some sketch of proof ideas. The second part of Chapter 10 presents in full proof the Holant* dichotomy that generalizes the main theorem of Chapter 2 to all, not necessarily symmetric constraints. For both theorems, a main proof technique is to reduce to the symmetric case and apply the dichotomy theorem for symmetric constraints. This last chapter serves as a retrospective on the journey we embarked starting from Chapter 2 for Holant* problems with symmetric constraints. Currently our knowledge about the classification program with asymmetric local constraint functions is still very limited. This knowledge is mainly restricted to Holant* problems and planar and nonplanar Boolean #CSP, as presented in Chapters 3 and 10.

https://doi.org/10.1017/9781107477063.002 Published online by Cambridge University Press