SIAM J. COMPUT.
Vol. 38, No. 5, pp. 1970–1986

THE COMPLEXITY OF WEIGHTED BOOLEAN #CSP*

MARTIN DYER†, LESLIE ANN GOLDBERG‡, AND MARK JERRUM§

Abstract. This paper gives a dichotomy theorem for the complexity of computing the partition function of an instance of a weighted Boolean constraint satisfaction problem. The problem is parameterized by a finite set $\mathcal{F}$ of nonnegative functions that may be used to assign weights to the configurations (feasible solutions) of a problem instance. Classical constraint satisfaction problems correspond to the special case of 0,1-valued functions. We show that computing the partition function, i.e., the sum of the weights of all configurations, is $\mathsf{FP}^{\#\mathsf{P}}$-complete unless either (1) every function in $\mathcal{F}$ is of "product type," or (2) every function in $\mathcal{F}$ is "pure affine." In the remaining cases, computing the partition function is in P.

Key words. complexity theory, counting, #P, constraint satisfaction

AMS subject classifications. Primary, 68Q25; Secondary, 05C15, 68T27

DOI. 10.1137/070690201

1. Introduction.

This paper gives a dichotomy theorem for the complexity of the partition function of weighted Boolean constraint satisfaction problems. Such problems are parameterized by a set $\mathcal{F}$ of nonnegative functions that may be used to assign weights to configurations (solutions) of the instance. These functions take the place of the allowed constraint relations in classical constraint satisfaction problems (CSPs). Indeed, the classical setting may be recovered by restricting $\mathcal{F}$ to functions with range $\{0, 1\}$. The key problem associated with an instance of a weighted CSP is to compute its partition function, i.e., the sum of weights of all its configurations. Computing the partition function of a weighted CSP may be viewed as a generalization of counting the number of satisfying solutions of a classical CSP. Many partition functions from statistical physics may be expressed as weighted CSPs. For example, the *Potts model* [22] is naturally expressible as a weighted CSP, whereas in the classical framework only the "hard-core" versions may be directly expressed. (The hard-core version of the *antiferromagnetic* Potts model corresponds to graph coloring, and the hard-core version of the *ferromagnetic* Potts model is trivial—acceptable configurations color the entire graph with a single color.) A corresponding weighted version of the decision CSP was investigated by Cohen et al. [3]. This results in optimization problems.

We use $\#\mathsf{CSP}(\mathcal{F})$ to denote the problem of computing the partition function of weighted CSP instances that can be expressed using only functions from $\mathcal{F}$. We show in Theorem 4 below that if every function $f \in \mathcal{F}$ is "of product type," then computing the partition function $Z(I)$ of an instance I can be done in polynomial time. Formal definitions are given later, but the condition of being of product type is easily checked—it essentially means that the partition function factors. We show further in

*Received by the editors May 1, 2007; accepted for publication (in revised form) August 25, 2008; published electronically January 14, 2009. This work was partly funded by the EPSRC grant "The complexity of counting in constraint satisfaction problems."
http://www.siam.org/journals/sicomp/38-5/69020.html
†School of Computing, University of Leeds, Leeds LS2 9JT, UK (dyer@comp.leeds.ac.uk).
‡Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK (l.a.goldberg@liverpool.ac.uk).
§School of Mathematical Sciences, Queen Mary, University of London, Mile End Road, London E1 4NS, UK (m.jerrum@qmul.ac.uk).

Theorem 4 that if every function $f \in \mathcal{F}$ is "pure affine," then the partition function of $Z(I)$ can be computed in polynomial time. Once again, there is an algorithm to check whether $\mathcal{F}$ is pure affine. For each other set $\mathcal{F}$, we show in Theorem 4 that computing the partition function of a $\#\mathsf{CSP}(\mathcal{F})$ instance is complete for the class $\mathsf{FP}^{\#\mathsf{P}}$. The existence of algorithms for testing the properties of being purely affine or of product type means that the dichotomy is effectively decidable.

1.1. Constraint satisfaction. Constraint satisfaction, which originated in artificial intelligence, provides a general framework for modeling decision problems and has many practical applications. (See, for example, [17].) Decisions are modelled by *variables*, which are subject to *constraints*, modelling logical and resource restrictions. The paradigm is sufficiently broad that many interesting problems can be modelled, from satisfiability problems to scheduling problems and graph-theory problems. Understanding the complexity of CSPs has become a major and active area within computational complexity [7, 13].

A CSP typically has a finite *domain*, which we will denote by $[q] = \{0, 1, \ldots, q-1\}$ for a positive integer q.[1] A *constraint language* Γ with domain $[q]$ is a set of relations on $[q]$. For example, take $q = 2$. The relation $R = \{(0,0,1), (0,1,0), (1,0,0), (1,1,1)\}$ is a 3-ary relation on the domain $\{0,1\}$, with four tuples.

Once we have fixed a constraint language Γ, an *instance* of the CSP is a set of *variables* $V = \{v_1, \ldots, v_n\}$ and a set of *constraints*. Each constraint has a *scope,* which is a tuple of variables (for example, (v_4, v_5, v_1)) and a relation from Γ of the same arity, which constrains the variables in the scope. A *configuration* σ is a function from V to $[q]$. The configuration σ is *satisfying* if the scope of every constraint is mapped to a tuple that is in the corresponding relation. In our example above, a configuration σ satisfies the constraint with scope (v_4, v_5, v_1) and relation R if and only if it maps an odd number of the variables in $\{v_1, v_4, v_5\}$ to the value 1. Given an instance of a CSP with constraint language Γ, the *decision problem* $\mathsf{CSP}(\Gamma)$ asks us to determine whether any configuration is satisfying. The *counting problem* $\#\mathsf{CSP}(\Gamma)$ asks us to determine the *number* of (distinct) satisfying configurations.

Varying the constraint language Γ defines the classes CSP and $\#\mathsf{CSP}$ of decision and counting problems. These contain problems of different computational complexities. For example, if $\Gamma = \{R_1, R_2, R_3\}$, where R_1, R_2, and R_3 are the three binary relations defined by $R_1 = \{(0,1),(1,0),(1,1)\}$, $R_2 = \{(0,0),(0,1),(1,1)\}$, and $R_3 = \{(0,0),(0,1),(1,0)\}$, then $\mathsf{CSP}(\Gamma)$ is the classical 2-satisfiability problem, which is in P. On the other hand, there is a similar constraint language Γ' with four relations of arity 3 such that 3-satisfiability (which is NP-complete) can be represented in $\mathsf{CSP}(\Gamma')$. It may happen that the counting problem is harder than the decision problem. If Γ is the constraint language of 2-satisfiability above, then $\#\mathsf{CSP}(\Gamma)$ contains the problem of counting independent sets in graph, and is $\#\mathsf{P}$-complete [21] even if restricted to 3-regular graphs [12].

Any decision problem $\mathsf{CSP}(\Gamma)$ is in NP, but not every problem in NP can be represented as a CSP. For example, the question "Is G Hamiltonian?" cannot naturally be expressed as a CSP, because the property of being Hamiltonian cannot be captured by relations of bounded size. This limitation of the class CSP has an important advantage. If $\mathsf{P} \neq \mathsf{NP}$, then there are problems which are neither in P nor NP-complete [15]. But, for well-behaved smaller classes of decision problems, the situation can be simpler. We may have a *dichotomy theorem*, partitioning all problems in the class into those

[1] Usually $[q]$ is defined to be $\{1, 2, \ldots, q\}$, but it is more convenient here to start the enumeration of domain elements at 0 rather than 1.

which are in P and those which are NP-complete. There are no "leftover" problems of intermediate complexity. It has been conjectured that there is a dichotomy theorem for CSP. The conjecture is that CSP(Γ) is in P for some constraint languages Γ, and CSP(Γ) is NP-complete for all other constraint languages Γ. This conjecture appeared in a seminal paper of Feder and Vardi [10] but has not yet been proved.

A similar dichotomy, between FP- and #P-complete, is conjectured for #CSP [2]. The complexity classes FP and #P are the analogues of P and NP for counting problems. FP is simply the class of functions computable in deterministic polynomial time. #P is the class of integer functions that can be expressed as the number of accepting computations of a polynomial-time nondeterministic Turing machine. Completeness in #P is defined with respect to polynomial-time Turing reducibility [16, Chap. 18]. Bulatov and Dalmau [2] have shown in one direction that, if #CSP(Γ) is solvable in polynomial time, then the constraints in Γ must have certain algebraic properties (assuming $\#\mathsf{P} \not\subseteq \mathsf{FP}$). In particular, they must have a so-called *Mal'tsev polymorphism*. The converse is known to be false, though it remains possible that the dichotomy (if it exists) does have an algebraic characterization.

The conjectured dichotomies for CSP and #CSP are major open problems for computational complexity theory. There have been many important results for subclasses of CSP and #CSP. We mention the most relevant to our paper here. The first decision dichotomy was that of Schaefer [18], for the Boolean domain $\{0, 1\}$. Schaefer's result is as follows.

THEOREM 1 (Schaefer [18]). *Let Γ be a constraint language with domain $\{0, 1\}$. The problem* CSP(Γ) *is in* P *if* Γ *satisfies one of the conditions below. Otherwise,* CSP(Γ) *is* NP*-complete.*

1. Γ *is* 0*-valid or* 1*-valid.*
2. Γ *is weakly positive or weakly negative.*
3. Γ *is affine.*
4. Γ *is bijunctive.*

We will not give detailed definitions of the conditions in Theorem 1, but the interested reader is referred to the paper [18] or to Theorem 6.2 of the textbook [7]. An interesting feature is that the conditions in [7, Thm. 6.2] are all checkable. That is, there is an algorithm to determine whether CSP(Γ) is in P or NP-complete, given a constraint language Γ with domain $\{0, 1\}$. Creignou and Hermann [6] adapted Schaefer's decision dichotomy to obtain a counting dichotomy for the Boolean domain. Their result is as follows.

THEOREM 2 (Creignou and Hermann [6]). *Let Γ be a constraint language with domain $\{0, 1\}$. The problem* #CSP(Γ) *is in* FP *if* Γ *is affine. Otherwise,* #CSP(Γ) *is* #P*-complete.*

A constraint language Γ with domain $\{0, 1\}$ is *affine* if every relation $R \in \Gamma$ is affine. A relation R is affine if the set of tuples $x \in R$ is the set of solutions to a system of linear equations over GF(2). These equations are of the form $v_1 \oplus \cdots \oplus v_n = 0$ and $v_1 \oplus \cdots \oplus v_n = 1$, where $\oplus$ is the *exclusive or* operator. It is well known (see, for example, Lemma 4.10 of [7]) that a relation R is affine if and only if $a, b, c \in R$ implies $d = a \oplus b \oplus c \in R$. (We will use this characterization below.) There is an algorithm for determining whether a Boolean constraint language Γ is affine, so there is an algorithm for determining whether #CSP(Γ) is in FP or #P-complete.

1.2. Weighted #CSP. The weighted graph homomorphism framework of [4] extends naturally to CSPs. Fix the domain $[q]$. Instead of constraining a length-k scope with an arity-k relation on $[q]$, we give a weight to the configuration on this scope by applying a function f from $[q]^k$ to the nonnegative rationals. Let $\mathcal{F}_q = \{f : [q]^k \to \mathbb{Q}^+ \mid k \in \mathbb{N}\}$ be the set of all such functions (of all arities).[2] Given a function $f \in \mathcal{F}_q$ of arity k, the *underlying relation* of f is given by $R_f = \{x \in [q]^k \mid f(x) \neq 0\}$. It is often helpful to think of R_f as a table, with k columns corresponding to the positions of a k-tuple. Each row corresponds to a tuple $x = (x_1, \dots, x_k) \in R_f$. The entry in row x and column j is x_j, which is a value in $[q]$.

A *weighted* #CSP problem is parameterized by a finite subset $\mathcal{F}$ of $\mathcal{F}_q$ and will be denoted by $\#\mathsf{CSP}(\mathcal{F})$. An instance I of $\#\mathsf{CSP}(\mathcal{F})$ consists of a set V of *variables* and a set $\mathcal{C}$ of *constraints*. Each constraint $C \in \mathcal{C}$ consists of a function $f_C \in \mathcal{F}$ (say of arity k_C) and a *scope*, which is a sequence $s_C = (v_{C,1}, \dots, v_{C,k_C})$ of variables from V. The variables $v_{C,1}, \dots, v_{C,k_C}$ need not be distinct. As in the unweighted case, a *configuration* σ for the instance I is a function from V to $[q]$. The *weight* of the configuration σ is given by

$$w(\sigma) = \prod_{C \in \mathcal{C}} f_C(\sigma(v_{C,1}), \dots, \sigma(v_{C,k_C})).$$

Finally, the *partition function* $Z(I)$ is given, for instance I, by

$$Z(I) = \sum_{\sigma : V \to [q]} w(\sigma). \tag{1}$$

In the computational problem $\#\mathsf{CSP}(\mathcal{F})$, the goal is to compute $Z(I)$, given an instance I.

Note that an (unweighted) CSP counting problem $\#\mathsf{CSP}(\Gamma)$ can be represented naturally as a weighted CSP counting problem. For each relation $R \in \Gamma$, let f^R be the indicator function for membership in R. That is, if $x \in R$, we set $f^R(x) = 1$. Otherwise we set $f^R(x) = 0$. Let $\mathcal{F} = \{f^R \mid R \in \Gamma\}$. Then for any instance I of $\#\mathsf{CSP}(\Gamma)$ the number of satisfying configurations for I is given by the (weighted) partition function $Z(I)$ from (1).

This framework has been employed previously in connection with *graph homomorphisms* [1]. Suppose $H = (H_{ij})$ is any symmetric $q \times q$ matrix H of rational numbers. We view H as being an edge-weighting of an undirected graph $\mathcal{H}$, where a zero weight in H means that the corresponding edge is absent from $\mathcal{H}$. Given a (simple) graph $G = (V, E)$, we consider computing the partition function

$$Z_H(G) = \sum_{\sigma : V \to [q]} w(\sigma), \quad \text{where} \quad w(\sigma) = \prod_{\{u,v\} \in E} H_{\sigma(u)\sigma(v)}.$$

Within our framework above, we view H as the binary function $h : [q]^2 \to \mathbb{R}$, and the problem is then computing the partition function of $\#CSP(\{h\})$.

Bulatov and Grohe [4] call H *connected* if $\mathcal{H}$ is connected and *bipartite* if $\mathcal{H}$ is bipartite. They give the following dichotomy theorem for nonnegative H.[3]

THEOREM 3 (Bulatov and Grohe [4]). *Let H be a symmetric matrix with nonnegative rational entries. Then we have the following:*

[2] We assume $0 \in \mathbb{N}$, so we allow nonnegative constants.

[3] This is not quite the original statement of the theorem. We have chosen here to restrict all inputs to be rational, in order to avoid issues of how to represent, and compute with, arbitrary real numbers.

1. *If H is connected and not bipartite, then computing Z_H is in* FP *if the rank of H is at most 1; otherwise computing Z_H is* $\#\mathsf{P}$*-hard.*
2. *If H is connected and bipartite, then computing Z_H is in* FP *if the rank of H is at most 2; otherwise computing Z_H is* $\#\mathsf{P}$*-hard.*
3. *If H is not connected, then computing Z_H is in* FP *if each of its connected components satisfies the corresponding conditions stated in 1 or 2; otherwise computing Z_H is* $\#\mathsf{P}$*-hard.*

Many partition functions arising in statistical physics may be viewed as weighted $\#\mathsf{CSP}$ problems. An example is the q-state Potts model (which is, in fact, a weighted graph homomorphism problem). In general, weighted $\#\mathsf{CSP}$ is very closely related to the problem of computing the partition function of a Gibbs measure in the framework of Dobrushin, Lanford, and Ruelle (see [1]). See also the framework of Scott and Sorkin [19].

1.3. Some notation. We will call the class of (rational) weighted #CSP problems *weighted* $\#\mathsf{CSP}$. The subclass having domain size $q = 2$ will be called weighted *Boolean* $\#\mathsf{CSP}$ and will be the main focus of this paper. We will give a dichotomy theorem for weighted Boolean $\#\mathsf{CSP}$.

Since weights can be arbitrary nonnegative rational numbers, the solution to these problems is not an integer in general. Therefore $\#\mathsf{CSP}(\mathcal{F})$ is not necessarily in the class $\#\mathsf{P}$. However, Goldberg and Jerrum [11] have observed that $Z(I) = \tilde{Z}(I)/K(I)$, where $\tilde{Z}$ is a function in $\#\mathsf{P}$ and $K(I)$ is a positive integer computable in FP. This follows because, for all $f \in \mathcal{F}$, we can ensure that $f(\cdot) = \tilde{f}(\cdot)/K(I)$, where $\tilde{f}(\cdot) \in \mathbb{N}$, by "clearing denominators." The denominator $K(I)$ can obviously be computed in polynomial time, and it is straightforward to show that computing $\tilde{Z}(I)$ is in $\#\mathsf{P}$, so the characterization of [11] follows. The resulting complexity class, comprising functions which are a function in $\#\mathsf{P}$ divided by a function in FP, is named $\#\mathsf{P}_{\mathbb{Q}}$ in [11], where it is used in the context of approximate counting. Clearly we have

$$\mathsf{weighted}\,\#\mathsf{CSP} \;\subseteq\; \#\mathsf{P}_{\mathbb{Q}} \;\subseteq\; \mathsf{FP}^{\#\mathsf{P}}.$$

On the other hand, if $Z(I) \in \mathsf{weighted}\,\#\mathsf{CSP}$ is $\#\mathsf{P}$-hard, then, using an oracle for computing $Z(I)$, we can construct a $\#\mathsf{P}$ oracle $\tilde{Z}(I)$ as outlined above. (Note that $Z(I) \notin \#\mathsf{P}$ in general.) Using this oracle, we can compute any function in $\mathsf{FP}^{\#\mathsf{P}}$ with a polynomial time-bounded oracle Turing machine. Thus any $\#\mathsf{P}$-hard function in $\mathsf{weighted}\,\#\mathsf{CSP}$ is complete for $\mathsf{FP}^{\#\mathsf{P}}$. We will use this observation to state our main result in terms of completeness for the class $\mathsf{FP}^{\#\mathsf{P}}$.

We make the following definition, which relates to the discussion above. We will say that $\mathcal{F} \subseteq \mathcal{F}_q$ *simulates* $f \in \mathcal{F}_q$ if, for each instance I of $\#\mathsf{CSP}(\mathcal{F} \cup \{f\})$, there is a polynomial time computable instance I' of $\#\mathsf{CSP}(\mathcal{F})$ such that $Z(I) = \varphi(I)Z(I')$ for some $\varphi(I) \in \mathbb{Q}$ which is FP-computable. This generalizes the notion of *parsimonious reduction* [16] among problems in $\#\mathsf{P}$. We will use $\leq_{\mathrm{T}}$ to denote the relation "is polynomial-time Turing-reducible to" between computational problems. Clearly, if $\mathcal{F}$ simulates f, we have $\#\mathsf{CSP}(\mathcal{F} \cup \{f\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\mathcal{F})$. Note also that, if $\tilde{f} = Kf$, for some constant $K > 0$, then $\{f\}$ simulates $\tilde{f}$. Thus there is no need to distinguish between "proportional" functions.

We use the following terminology for certain functions. Let $\chi_{=}$ be the binary *equality* function defined on $[q]$ as follows. For any element $c \in [q]$, $\chi_{=}(c, c) = 1$, and for any pair (c, d) of distinct elements of $[q]$, $\chi_{=}(c, d) = 0$. Let $\chi_{\neq}$ be the binary *disequality* function given by $\chi_{\neq}(c,d) = 1 - \chi_{=}(c,d)$ for all $c,d \in [q]$.[4] We say that a function f is of *product type* if f can be expressed as a product of unary functions and binary functions of the form $\chi_{=}$ and $\chi_{\neq}$.

We focus attention in this paper on the Boolean case, $q = 2$. In this case, we say that a function $f \in \mathcal{F}_2$ has *affine support* if its underlying relation R_f, defined earlier, is affine. We say that f is *pure affine* if it has affine support and range $\{0, w\}$ for some $w > 0$. Thus a function is pure affine if and only if it is a positive real multiple of some (0,1-valued) function which is affine over GF(2).

1.4. Our result. Our main result is the following.

THEOREM 4. *Suppose $\mathcal{F} \subseteq \mathcal{F}_2 = \{f : \{0,1\}^k \to \mathbb{Q}^+ \mid k \in \mathbb{N}\}$. If every function in $\mathcal{F}$ is of product type, then $\#\mathsf{CSP}(\mathcal{F})$ is in FP. If every function in $\mathcal{F}$ is pure affine, then $\#\mathsf{CSP}(\mathcal{F})$ is in FP. Otherwise, $\#\mathsf{CSP}(\mathcal{F})$ is $\mathsf{FP}^{\#\mathsf{P}}$-complete.*

Proof. Suppose first that $\mathcal{F}$ is of product type. In this case the partition function $Z(I)$ of an instance I with variable set V is easy to evaluate because it can be factored into easy-to-evaluate pieces: Partition the variables in V into equivalence classes according to whether or not they are related by an equality or disequality function. (The equivalence relation on variables here is "depends linearly on.") An equivalence class consists of two (possibly empty) sets of variables U_1 and U_2. All of the variables in U_1 must be assigned the same value by a configuration σ of nonzero weight, and all variables in U_2 must be assigned the other value. Variables in $U_1 \cup U_2$ are not related by equality or disequality to variables in $V \setminus (U_1 \cup U_2)$. The equivalence class contributes one weight, say α, to the partition function if variables in U_1 are given value "0" by σ, and it contributes another weight, say β, to the partition function if variables in U_1 are given value "1" by σ. Thus, $Z(I) = (\alpha + \beta)Z(I')$, where I' is the instance formed from I by removing this equivalence class. Therefore, suppose we choose any equivalence class and remove its variables. Since $\mathcal{F}$ contains only unary, equality, or binary disequality constraints, we can also remove all functions involving variables in $U_1 \cup U_2$ to give $\mathcal{F}'$. Then I' is of product type with fewer variables, so we may compute $Z(I')$ recursively.

Suppose second that $\mathcal{F}$ if pure affine. Then $Z(I) = \prod_{f \in \mathcal{F}} w_f^{k_f} Z(I')$, where $\{0, w_f\}$ is the range of f, k_f is the number of constraints involving f in I, and I' is the instance obtained from I by replacing every function f by its underlying relation R_f (viewed as a function with range $\{0,1\}$). $Z(I')$ is easy to evaluate, because this is just counting solutions to a linear system over GF(2), as Creignou and Hermann have observed [6].

Finally, the $\#\mathsf{P}$-hardness in Theorem 4 follows from Lemma 5 below. □

LEMMA 5. *If $f \in \mathcal{F}_2$ is not of product type and $g \in \mathcal{F}_2$ is not pure affine, then $\#\mathsf{CSP}(\{f, g\})$ is $\#\mathsf{P}$-hard.*

Note that the functions f and g in Lemma 5 may be one and the same function. So $\#\mathsf{CSP}(\{f\})$ is $\#\mathsf{P}$-hard when f is not of product type nor pure affine. The rest of this article gives the proof of Lemma 5.

2. Useful tools for proving hardness of #CSP.

2.1. Notation. For any sequence $u_1, \ldots, u_k$ of variables of I and any sequence $c_1, \ldots, c_k$ of elements of the domain $[q]$, we will let $Z(I \mid \sigma(u_1) = c_1, \ldots, \sigma(u_k) = c_k)$ denote the contribution to $Z(I)$ from assignments σ with $\sigma(u_1) = c_1, \ldots, \sigma(u_k) = c_k$.

2.2. Projection. The first tool that we study is projection, which is referred to as "integrating out" in the statistical physics literature.

[4] A more general disequality function is defined in the appendix.

Let f be a function of arity k, and let $J = \{j_1, \ldots, j_r\}$ be a size-r subset of $\{1, \ldots, k\}$, where $j_1 < \cdots < j_r$.[5] We say that a k-tuple $x' \in [q]^k$ *extends* an r-tuple $x \in [q]^r$ on J (written $x' \sqsupseteq_J x$) if x' agrees with x on indices in J; that is to say, $x'_{j_i} = x_i$ for all $1 \leq i \leq r$. The *projection* g of f onto J is defined as follows. For every $x \in [q]^r$, $g(x) = \sum_{x' \sqsupseteq_J x} f(x')$.

The following lemma may be viewed as a weighted version of Proposition 2 of [2], where it is proved for the unweighted case. It is expressed somewhat differently in [2], in terms of counting the number of solutions to an existential formula.

LEMMA 6. *Suppose $\mathcal{F} \subseteq \mathcal{F}_q$. Let g be a projection of a function $f \in \mathcal{F}$ onto a subset of its indices. Then $\#\mathsf{CSP}(\mathcal{F} \cup \{g\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\mathcal{F})$.*

Proof. Let k be the arity of f, and let g be the projection of f onto the subset J of its indices. Let I be an instance of $\#\mathsf{CSP}(\mathcal{F} \cup \{g\})$. We will construct an instance I' of $\#\mathsf{CSP}(\mathcal{F})$ such that $Z(I) = Z(I')$. The instance I' is identical to I except that every constraint C of I involving g is replaced with a new constraint C' of I' involving f. The corresponding scope $(v_{C',1}, \ldots, v_{C',k})$ is constructed as follows. If j_ℓ is the ℓth element of J, then $v'_{C',j_\ell} = v_{C,\ell}$. The other variables, $v_{C',j}$ $(j \notin J)$, are distinct new variables. We have shown that $\mathcal{F}$ simulates g with $\phi(I) = 1$. ∎

2.3. Pinning. For $c \in [q]$, δ_c denotes the unary function with $\delta_c(c) = 1$ and $\delta_c(d) = 0$ for $d \neq c$. The following lemma, which allows "pinning" CSP variables to specific values in hardness proofs, generalizes Theorem 8 of [2], which does the unweighted case. Again [2] employs different terminology, and its theorem is a statement about the full idempotent reduct of a finite algebra. The idea of pinning was used previously by Bulatov and Grohe of [4] in the context of counting weighted graph homomorphisms (see Lemma 32 of [4]). A similar idea was used by Dyer and Greenhill in the context of counting *unweighted* graph homomorphisms—in that context, Theorem 4.1 of [8] allows pinning all variables to a particular *component* of the target graph H.

LEMMA 7. *For every $\mathcal{F} \subseteq \mathcal{F}_q$, $\#\mathsf{CSP}(\mathcal{F} \cup \bigcup_{c \in [q]} \delta_c) \leq_{\mathrm{T}} \#\mathsf{CSP}(\mathcal{F})$.*

The proof of Lemma 7 is deferred to the appendix. Since we use only the case $q = 2$ in this paper, we provide the (simpler) proof for the Boolean case here.

LEMMA 8. *For every $\mathcal{F} \subseteq \mathcal{F}_2$, $\#\mathsf{CSP}(\mathcal{F} \cup \{\delta_0, \delta_1\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\mathcal{F})$.*

Proof. For $x \in [2]^k$, let $\overline{x}$ be the k-tuple whose ith component, $\overline{x}_i$, is $x_i \oplus 1$ for all i. Say that $\mathcal{F}$ is *symmetric* if it is the case that for every arity-k function $f \in \mathcal{F}$ and every $x \in [2]^k$, $f(\overline{x}) = f(x)$.

Given an instance I of $\#\mathsf{CSP}(\mathcal{F} \cup \{\delta_0, \delta_1\})$ with variable set V, we consider two instances I' and I'' of $\#\mathsf{CSP}(\mathcal{F})$. Let V_0 be the set of variables v of I to which the constraint $\delta_0(v)$ is applied. Let V_1 be the set of variables v of I to which the constraint $\delta_1(v)$ is applied. We can assume without loss of generality that V_0 and V_1 do not intersect. (Otherwise, $Z(I) = 0$ and we can determine this without using an oracle for $\#\mathsf{CSP}(\mathcal{F})$.) Let $V_2 = V \setminus (V_0 \cup V_1)$. The instance I' has variables $V_2 \cup \{t_0, t_1\}$, where t_0 and t_1 are distinct new variables that are not in V. Every constraint C of I involving a function $f \in \mathcal{F}$ corresponds to a constraint C' of I'. C' is the same as C except that variables in V_0 are replaced with t_0 and variables in V_1 are replaced with t_1. Similarly, the instance I'' has variables $V_2 \cup \{t\}$, where t is a new variable that is not in V. Every constraint C of I involving a function $f \in \mathcal{F}$ corresponds to a constraint C'' of I''. The constraint C'' is the same as C except that variables in $V_0 \cup V_1$ are replaced with t.

[5]It is not necessary to choose this particular ordering for J, but it is convenient to do so.

Case 1. $\mathcal{F}$ is symmetric: By construction,

$$Z(I') - Z(I'') = Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1) + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0).$$

By symmetry, the summands are the same, so

$$Z(I') - Z(I'') = 2Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1) = 2Z(I).$$

Case 2. $\mathcal{F}$ is not symmetric: Let f be an arity-k function in $\mathcal{F}$, and let $x \in [2]^k$ so that $f(x) > f(\overline{x}) \geq 0$. Let $s = (t_{x_1}, \ldots, t_{x_k})$, and let I'_x be the instance derived from I' by adding a new constraint with function f and scope s. Similarly, let I''_x be the instance derived from I'' by adding a new constraint with function f and scope $(t, \ldots, t)$. Now

$$\begin{aligned} Z(I'_x) &= Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}) \\ &\quad + Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 0)f(0, \ldots, 0) + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 1)f(1, \ldots, 1) \\ &= Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}) + Z(I''_x). \end{aligned}$$

Thus we have two independent equations,

$$\begin{aligned} Z(I'_x) - Z(I''_x) &= Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}), \\ Z(I') - Z(I'') &= Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1) \qquad + Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0) \qquad , \end{aligned}$$

in the unknowns $Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1)$ and $Z(I' \mid \sigma(t_0) = 1, \sigma(t_1) = 0)$. Solving these, we obtain the value of $Z(I' \mid \sigma(t_0) = 0, \sigma(t_1) = 1) = Z(I)$. ∎

2.4. #P-hard problems. To prove Lemma 5, we will give reductions from some known #P-hard problems. The first of these is the problem of counting homomorphisms from simple graphs to 2-vertex multigraphs. We use the following special case of Bulatov and Grohe's Theorem 3.

COROLLARY 9 (Bulatov and Grohe [4]). *Let H be a symmetric 2×2 matrix with nonnegative real entries. If H has rank 2 and at most one entry of H is 0, then* EVAL(H) *is #P-hard.*

We will also use the problem of computing the *weight enumerator* of a linear code. Given a *generating matrix* $A \in \{0,1\}^{r \times C}$ of rank r, a *code word* c is any vector in the linear subspace Υ generated by the rows of A over GF(2). For any real number λ, the *weight enumerator* of the code is given by $W_A(\lambda) = \sum_{c \in \Upsilon} \lambda^{\|c\|}$, where $\|c\|$ is the number of 1's in c. The problem of computing the weight enumerator of a linear code is in FP for $\lambda \in \{-1, 0, 1\}$ and is known to be #P-hard for every other fixed $\lambda \in \mathbb{Q}$ (see [22]). We could not find a proof, so we provide one here. We restrict our attention to positive λ, since that is adequate for our purposes.

LEMMA 10. *Computing the weight enumerator of a linear code is #P-hard for any fixed positive rational number $\lambda \neq 1$.*

Proof. We will prove hardness by reduction from a problem EVAL(H), for some appropriate H, using Corollary 9. Let the input to EVAL(H) be a connected graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Let B be the $n \times m$ incidence matrix of G, with $b_{ij} = 1$ if $v_i \in e_j$ and $b_{ij} = 0$ otherwise. Let A be the $(n-1) \times m$ matrix which is B with the row for v_n deleted. A will be the generating matrix of the weight enumerator instance, with $r = n - 1$ and $C = m$. It has rank $(n-1)$ since G contains a spanning tree. A code word c has $c_j = \bigoplus_{i \in U} b_{ij}$, where $U \subseteq V \setminus \{v_n\}$. Thus $c_j = 1$ if and only if e_j has exactly one endpoint in U, and the weight of c is λ^k, where k is the number of edges in the cut $U, V \setminus U$. Thus $W_A(\lambda) = \frac{1}{2} Z_H(G)$, where

H is the symmetric weight matrix with $H_{11} = H_{22} = 1$ and $H_{12} = H_{21} = \lambda$. The $\frac{1}{2}$ arises because we fixed which side of the cut contains v_n. Now H has rank 2 unless $\lambda = 1$, so this problem is #P-hard by Corollary 9. Note, by the way, that $Z_H(G)$ is the partition function of the Ising model in statistical physics [5]. □

3. The proof of Lemma 5. Throughout this section, we assume $q = 2$. The following lemma is a generalization of a result of Creignou and Hermann [6], which deals with the case in which f is a relation (or, in our setting, a function with range $\{0, 1\}$). The inductive technique used in the proof of Lemma 11 (combined with the follow-up in Lemma 12) is good for showing that $\#\mathsf{CSP}(\mathcal{F})$ is #P-hard when $\mathcal{F}$ contains a *single* function. A very different situation arises when $\#\mathsf{CSP}(\{f\})$ and $\#\mathsf{CSP}(\{g\})$ are in FP but $\#\mathsf{CSP}(\{f, g\})$ is #P-hard due to *interactions* between f and g—we deal with that problem later.

LEMMA 11. *Suppose that $f \in \mathcal{F}_2$ does not have affine support. Then $\#\mathsf{CSP}(\{f\})$ is #P-hard.*

Proof. Let k be the arity of f, and let us denote the ith component of k-tuple $a \in R_f$ by a_i. The proof is by induction on k. The lemma is trivially true for $k = 1$, since all functions of arity 1 have affine support.

For $k = 2$, we note that since R_f is not affine, it is of the form $R_f = \{(\alpha, \beta), (\bar{\alpha}, \beta), (\bar{\alpha}, \bar{\beta})\}$ for some $\alpha \in \{0, 1\}$ and $\beta \in \{0, 1\}$. We can show that $\#\mathsf{CSP}(\{f\})$ is #P-hard by reduction from $\textsc{Eval}(H)$ using

$$H = \begin{pmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{pmatrix},$$

which has rank 2 and exactly one entry that is 0. Given an instance $G = (V, E)$ of $\textsc{Eval}(H)$, we construct an instance I of $\#\mathsf{CSP}(\{f\})$ as follows. The variables of I are the vertices of G. For each edge $e = (u, v)$ of G, add a constraint with function f and variable sequence u, v. Corollary 9 now tells us that $\textsc{Eval}(H)$ is #P-hard, so $\#\mathsf{CSP}(\{f\})$ is #P-hard.

Suppose $k > 2$. We start with some general arguments and notation. For any $i \in \{1, \ldots, k\}$ and any $\alpha \in \{0, 1\}$ let $f^{i=\alpha}$ be the function of arity $k-1$ derived from f by pinning the ith position to α. That is, $f^{i=\alpha}(x_1, \ldots, x_{k-1}) = f(x_1, \ldots, x_{i-1}, \alpha, x_{i+1}, \ldots, x_k)$. Also, let $f^{i=*}$ be the projection of f onto all positions apart from position i (see section 2.2). Note that $\#\mathsf{CSP}(\{f^{i=\alpha}\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f, \delta_0, \delta_1\})$, since $f^{i=\alpha}$ can obviously be simulated by $\{f, \delta_0, \delta_1\}$. Furthermore, by Lemma 8, $\#\mathsf{CSP}(\{f, \delta_0, \delta_1\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f\})$. Thus, we can assume that $f^{i=\alpha}$ has affine support—otherwise, we are finished by induction. Similarly, by Lemma 6, $\#\mathsf{CSP}(\{f^{i=*}\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f\})$. Thus we can assume that $f^{i=*}$ has affine support—otherwise, we are finished by induction.

Now, recall that R_f is not affine. Consider any $a, b, c \in R_f$ such that $d = a \oplus b \oplus c \notin R_f$. We have four cases.

Case 1. There are indices $1 \leq i < j \leq k$ such that $(a_i, b_i, c_i) = (a_j, b_j, c_j)$. Without loss of generality, suppose $i = 1$ and $j = 2$. Define the function f' of arity $(k-1)$ by $f'(r_2, \ldots, r_k) = f(r_2, r_2, \ldots, r_k)$. Note that $R_{f'}$ is not affine since the condition $a \oplus b \oplus c \notin R_f$ is inherited by $R_{f'}$. So, by induction, $\#\mathsf{CSP}(\{f'\})$ is #P-hard. Now note that $\#\mathsf{CSP}(\{f'\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f\})$. To see this, note that any instance I_1 of $\#\mathsf{CSP}(\{f'\})$ can be turned into an instance I of $\#\mathsf{CSP}(\{f\})$ by repeating the first variable in the sequence of variables for each constraint.

Case 2. There is an index $1 \le i \le k$ such that $a_i = b_i = c_i$. Since d is not in R_f and $d_i = a_i$, we find that $f^{i=a_i}$ does not have affine support, contrary to earlier assumptions.

Having finished Cases 1 and 2, we may assume without loss of generality that we are in Case 3 or 4 below, where $\{\alpha, \beta\} \in \{0,1\}$, $\bar{\alpha} = 1 - \alpha$, $\bar{\beta} = 1 - \beta$, and $a', b', c' \in \{0,1\}^{k-2}$.

Case 3. $a = (\bar{\alpha}, \bar{\beta}, a')$, $b = (\bar{\alpha}, \beta, b')$, $c = (\alpha, \bar{\beta}, c')$. Since $R_{f^{1=*}}$ is affine and a, b, and c are in R_f, we must have either $d = (\alpha, \beta, d') \in R_f$ or $e = (\bar{\alpha}, \beta, d') \in R_f$, where $d' = a' \oplus b' \oplus c'$. In the first case, we are done (we have contradicted the assumption that $d \notin R_f$), so assume that $e \in R_f$ but $d \notin R_f$. Similarly, since $R_{f^{2=*}}$ is affine, we may assume that $g = (\alpha, \bar{\beta}, d') \in R_f$. Since $R_{f^{1=\bar{\alpha}}}$ is affine and a, b, and e are in R_f, we find that $h = a \oplus b \oplus e = (\bar{\alpha}, \bar{\beta}, c') \in R_f$. Since $R_{f^{2=\bar{\beta}}}$ is affine and a, c, and g are in R_f, we find that $i = (\bar{\alpha}, \bar{\beta}, b') \in R_f$. Also, since $R_{f^{2=\bar{\beta}}}$ is affine and a, h, and i are in R_f, we find that $j = (\bar{\alpha}, \bar{\beta}, d') \in R_f$. Let $f'(r_1, r_2) = f(r_1, r_2, d_3, \ldots, d_k)$. Since e, g, and j are in R_f but d is not, we have $(\bar{\alpha}, \beta), (\alpha, \bar{\beta}), (\bar{\alpha}, \bar{\beta}) \in R_{f'}$, but $(\alpha, \beta) \notin R_{f'}$. Thus, f' does not have affine support and $\#\mathsf{CSP}(\{f'\})$ is #P-hard by induction. Also, $\#\mathsf{CSP}(\{f'\}) \le_{\mathrm{T}} \#\mathsf{CSP}(\{f\})$ by Lemma 8.

Case 4. $a = (\bar{\alpha}, \alpha, a')$, $b = (\bar{\alpha}, \alpha, b')$, $c = (\alpha, \bar{\alpha}, c')$. Since $R_{f^{1=*}}$ is affine and a, b, and c are in R_f but d is not, we have $e = (\bar{\alpha}, \bar{\alpha}, d') \in R_f$. Similarly, since $R_{f^{2=*}}$ is affine and a, b, and c are in R_f but d is not, we have $g = (\alpha, \alpha, d') \in R_f$. Now since $R_{f^{1=\bar{\alpha}}}$ is affine and a, b, and e are in R_f, we have $h = (\bar{\alpha}, \bar{\alpha}, c') \in R_f$. Also, since $R_{f^{2=\alpha}}$ is affine and a, b, and g are in R_f, we have $i = (\alpha, \alpha, c') \in R_f$.

Let $f'(r_1, r_2) = f(r_1, r_2, c_3, \ldots, c_k)$. If $j = (\bar{\alpha}, \alpha, c') \notin R_f$, then f' does not have affine support (since c, h, and i are in R_f), so we finish by induction as in Case 3. Suppose $j \in R_f$. Since $R_{f^{1=\bar{\alpha}}}$ is affine and a, b, and j are in R_f, we have $\ell = (\bar{\alpha}, \alpha, d') \in R_f$. Let $f''(r_1, r_2) = f(r_1, r_2, d_3, \ldots, d_k)$. Then f'' does not have affine support (since e, g, and ℓ are in R_f but d is not), so we finish by induction as in Case 3. ∎

Lemma 11 showed that $\#\mathsf{CSP}(\{f\})$ is #P-hard when f does not have affine support. The following lemma gives another (rather technical, but useful) condition which implies that $\#\mathsf{CSP}(\{f\})$ is #P-hard. We start with some notation. Let f be an arity-k function. For a value $b \in \{0,1\}$, an index $i \in \{1, \ldots, k\}$, and a tuple $y \in \{0,1\}^{k-1}$, let $y^{i=b}$ denote the tuple $x \in \{0,1\}^k$ formed by setting $x_i = b$ and $x_j = y_j$ $(j \in \{1, \ldots, k\} \setminus \{i\})$.

We say that index i of f is *useful*, if there is a tuple y such that $f(y^{i=0}) > 0$ and $f(y^{i=1}) > 0$. We say that f is *product-like* if, for every useful index i, there is a rational number λ_i such that, for all $y \in \{0,1\}^{k-1}$,

$$f(y^{i=0}) = \lambda_i f(y^{i=1}). \tag{2}$$

If every position i of f is useful, then being product-like is the same as being of product type. However, being product-like is less demanding because it does not restrict indices that are not useful.

LEMMA 12. *If $f \in \mathcal{F}_2$ is not product-like, then* $\#\mathrm{CSP}(\{f\})$ *is* #P*-hard.*

Proof. We will use Corollary 9 to prove hardness, following an argument from [9]. Choose a useful index i so that there is no λ_i satisfying (2).

Suppose f has arity k. Let A be the $2 \times 2^{k-1}$ matrix such that for $b \in \{0,1\}$ and $y \in \{0,1\}^{k-1}$, $A_{b,y} = f(y^{i=b})$. Let $A' = AA^T$.

First, we show that $\textsc{Eval}(A')$ is #P-hard. Note that A' is the following symmetric 2×2 matrix with nonnegative rational entries:

$$\begin{pmatrix} \sum_y A_{0,y}^2 & \sum_y A_{0,y}A_{1,y} \\ \sum_y A_{0,y}A_{1,y} & \sum_y A_{1,y}^2 \end{pmatrix} = \begin{pmatrix} \sum_y f(y^{i=0})^2 & \sum_y f(y^{i=0})f(y^{i=1}) \\ \sum_y f(y^{i=0})f(y^{i=1}) & \sum_y f(y^{i=1})^2 \end{pmatrix}.$$

Since index i is useful, all four entries of A' are positive. To show that $\textsc{Eval}(A')$ is #P-hard by Corollary 9, we just need to show that its determinant is nonzero. By the Cauchy–Schwarz equation, the determinant is nonnegative and is zero only if λ_i exists, which we have assumed not to be the case. Thus $\textsc{Eval}(A')$ is #P-hard by Corollary 9.

Now we reduce $\textsc{Eval}(A')$ to #CSP($\{f\}$). To do this, take an undirected graph G which is an instance of $\textsc{Eval}(A')$. Construct an instance Y of #CSP($\{f\}$). For every vertex v of G we introduce a variable x_v of Y. Also, for every edge e of G we introduce $k-1$ variables $x_{e,1}, \dots, x_{e,k-1}$ of Y. We introduce constraints in Y as follows. For each edge $e = (v, v')$ of G we introduce constraints $f(x_v, x_{e,1}, \dots, x_{e,k-1})$ and $f(x_{v'}, x_{e,1}, \dots, x_{e,k-1})$ into Y, where we have assumed, without loss of generality, that the first index is useful.

It is clear that $\textsc{Eval}(A')$ is exactly equal to the partition function of the #CSP($\{f\}$) instance Y. ∎

For $w \in \mathbb{Q}^+$, let U_w denote the unary function mapping 0 to 1 and 1 to w. Note that $U_0 = \delta_0$, and U_1 gives the constant (0-ary function) 1, occurrences of which leave the partition function unchanged. So, by Lemma 8, we can discard these constraints since they do not add to the complexity of the problem. Note, by the observation above about proportional functions, that the functions U_w include all unary functions except for δ_1 and the constant 0. We can discard δ_1 by Lemma 8, and if the constant 0 function is in $\mathcal{F}$, any instance I where it appears as a constraint has $Z(I) = 0$. So again we can discard these constraints since they do not add to the complexity of the problem.

Thus U_w will be called *nontrivial* if $w \notin \{0, 1\}$. Let $\oplus_k : \{0,1\}^k \to \{0,1\}$ be the arity-k parity function that is 1 if and only if its argument has an odd number of 1's. Let $\neg\oplus_k : \{0,1\}^k \to \{0,1\}$ be the function $1 - \oplus_k$. The following lemma shows that even a simple function like $\oplus_3$ can lead to intractable #CSP instances when it is combined with a nontrivial weight function U_λ.

LEMMA 13. *$\#\mathsf{CSP}(\oplus_3, U_\lambda, \delta_0, \delta_1)$ and $\#\mathsf{CSP}(\neg\oplus_3, U_\lambda, \delta_0, \delta_1)$ are both #P-hard, for any positive $\lambda \neq 1$.*

Proof. We give a reduction from computing the weight enumerator of a linear code, which was shown to be #P-hard in Lemma 10. In what follows, it is sometimes convenient to view $\oplus_k$, δ_0, etc., as relations as well as functions to $\{0, 1\}$.

We first argue that, for any k, the relation $\oplus_k$ can be simulated by $\{\oplus_3, \delta_0, \delta_1\}$. For example, to simulate $x_1 \oplus \cdots \oplus x_k$ for $k > 3$, take new variables y, z, and w and let $m = \lceil k/2 \rceil$ and use $x_1 \oplus \cdots \oplus x_m \oplus y$ and $x_{m+1} \oplus \cdots \oplus x_k \oplus z$ and $y \oplus z \oplus w$ and $\delta_0(w)$.

Since $\{\oplus_3, \delta_0, \delta_1\}$ can be used to simulate any relation $\oplus_k$, we can use $\{\oplus_3, \delta_0, \delta_1\}$ to simulate an arbitrary system of linear equations over GF(2). In particular, we can use them to simulate the subspace Υ of code words for a given generating matrix A.

Finally, we can use U_λ to simulate the function which evaluates the weight enumerator on Υ. Then, since $\lambda \neq 0, 1$, we can apply Lemma 10 to complete the argument. The same proof, with minor modifications, applies to $\neg\oplus_3$. ∎

LEMMA 14. *Suppose that $f \in \mathcal{F}_2$ is not of product type. Then, for any positive $\lambda \neq 1$, there exists a constant c, depending on f, such that $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda, U_c\})$ is $\#\mathsf{P}$-hard.*

Proof. If f does not have affine support, the result follows by Lemma 11. So suppose f has affine support. Consider the underlying relation R_f, viewed as a table. The rows of the table represent the tuples of the relation. Let J be the set of columns on which the relation is not constant. That is, if $i \in J$, then there is a row x with $x_i = 0$ and a row y with $y_i = 1$. Group the columns in J into equivalence classes: two columns are equivalent if and only if they are equal or complementary. Let k be the number of equivalence classes. Take one column from each of the k equivalence classes as a representative, and focus on the arity-k relation R induced by those columns.

Case 1. Suppose that R is the complete relation of arity k. Let f^* be the projection of f onto the k columns of R. By Lemma 6,

$$\#\mathsf{CSP}(\{f^*\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f\}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda, U_c\}).$$

We will argue that $\#\mathsf{CSP}(\{f^*\})$ is $\#\mathsf{P}$-hard. To see this, note that every column of f^* is useful. Thus, if f^* were product-like, we could conclude that f^* was of product type. But this would imply that f is of product type, which is not the case by assumption. So f^* is not product-like, and hardness follows from Lemma 12.

Case 2. Suppose that R is not the complete relation of arity k. We had assumed that R_f is affine. This means that, given three vectors, x, y, and z in R_f, $x \oplus y \oplus z$ is in R_f as well. The arity-k relation R inherits this property, so is also affine.

Choose a minimal set of columns of R that do not induce the complete relation. This exists by assumption. Suppose that there are j columns in this minimal set. Observe that $j \neq 1$ because there are no constant columns in J. Also $j \neq 2$, since otherwise the two columns would be related by equality or disequality, contradicting the preprocessing step. The argument here is that on two columns, R cannot have exactly three tuples because it is affine, and having tuples x, y, and z in would require the fourth tuple $x \oplus y \oplus z$. But if it has two tuples, then, because there are no constant columns, the only possibilities are either $(0, 0)$ and $(1, 1)$ or $(0, 1)$ and $(1, 0)$. Both contradict the preprocessing step, so $j \geq 3$.

Let R' be the restriction of R to the j columns. Now R' of course has fewer than 2^j rows, and at least 2^{j-1} by minimality. It is affine, and hence must be $\oplus_j$ or $\neg\oplus_j$. To see this, first note that the size of R' has to be a power of 2 since R' is the solution to a system of linear equations. Hence the size of R' must be 2^{j-1}. Then, since there are j variables, there can only be one defining equation. And, since every subset of $j - 1$ variables induces a complete relation, this single equation must involve all variables. Therefore, the equation is $\oplus_j$ or $\neg\oplus_j$.

Let f' be the projection of f onto the j columns just identified. Let f'' be further obtained by pinning all but three of the j variables to 0. Pinning $j - 3$ variables to 0 leaves a single equation involving all three remaining variables. Thus $R_{f''}$ must be $\oplus_3$ or $\neg\oplus_3$.

Now define the symmetric function f''' by

$$f'''(a, b, c) = f''(a, b, c) f''(a, c, b) f''(b, a, c) f''(b, c, a) f''(c, a, b) f''(c, b, a).$$

Note that $R_{f'''}$ is $\oplus_3$ or $\neg\oplus_3$, since $R_{f''}$ is symmetric and hence $R_{f'''} = R_{f''}$.

To summarize: using f and the constant functions δ_0 and δ_1, we have simulated a function f''' such that its underlying relation $R_{f'''}$ is either $\oplus_3$ or $\neg\oplus_3$. Furthermore, if triples x and y have the same number of 1's, then $f'''(x) = f'''(y)$.

We can now simulate an unweighted version of $\oplus_3$ or $\neg\oplus_3$ using f''' and a unary function U_c, with c set to a conveniently chosen value. There are two cases. Suppose first that the affine support of f''' is $\neg\oplus_3$. Then let w_0 denote the value of f''' when applied to the 3-tuple $(0,0,0)$, and let w_2 denote $f'''(0,1,1) = f'''(1,0,1) = f'''(1,1,0)$. Recall that $f'''(x) = 0$ for any other 3-tuple x. Now let $c = (w_0/w_2)^{1/2}$. Note from the definition of f''' that w_0 and w_2 are squares of rational numbers, so c is also rational. Define a function g of arity 3 by $g(\alpha,\beta,\gamma) = U_c(\alpha)U_c(\beta)U_c(\gamma)f'''(\alpha,\beta,\gamma)$. Note that $g(0,0,0) = w_0$ and $g(0,1,1) = g(1,0,1) = g(1,1,0) = c^2 w_2 = w_0$. Thus, g is a pure affine function with affine support $\neg\oplus_3$ and range $\{0, w_0\}$. The other case, in which the affine support of f''' is $\oplus_3$, is similar.

We have established a reduction from either $\#\mathsf{CSP}(\oplus_3, U_\lambda, \delta_0, \delta_1)$ or $\#\mathsf{CSP}(\neg\oplus_3, U_\lambda, \delta_0, \delta_1)$, which are both #P-hard by Lemma 13. ∎

LEMMA 15. *If $f \in \mathcal{F}_2$ is not of product type, then $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda\})$ is #P-hard for any positive $\lambda \neq 1$.*

Proof. Take an instance I of $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda, U_c\})$, from Lemma 14, with n variables $x_1, x_2, \ldots, x_n$. We want to compute the partition function $Z(I)$ using only instances of $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda\})$, that is, instances which avoid using constraints U_c. For each i, let m_i denote the number of copies of U_c that are applied to x_i, and let $m = \sum_{i=1}^n m_i$. Then we can write the partition function as $Z(I) = Z(I; c)$, where

$$Z(I;w) = \sum_{\sigma \in \{0,1\}^n} \hat{Z}(\sigma) \prod_{i:\sigma_i = 1} w^{m_i} = \sum_{\sigma \in \{0,1\}^n} \hat{Z}(\sigma) w^{\sum_{i=1}^n m_i \sigma_i},$$

where $\hat{Z}(\sigma)$ denotes the value corresponding to the assignment $\sigma(x_i) = \sigma_i$, ignoring constraints applying U_c, and w is a variable. So $\hat{Z}(\sigma)$ is the weight of σ, taken over all constraints other than those applying U_c. Note also that $Z(I;w)$ is a polynomial of degree m in w. We can evaluate $Z(I;w)$ at the point $w = \lambda^j$ by replacing each U_c constraint with j copies of a U_λ constraint. This evaluation is an instance of $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda\})$. So, using m different values of j and interpolating, we learn the coefficients of the polynomial $Z(I;w)$. Then we can set $w = c$ to evaluate $Z(I)$. ∎

LEMMA 16. *Suppose that $f \in \mathcal{F}_2$ is not of product type and $g \in \mathcal{F}_2$ is not pure affine. Then $\#\mathsf{CSP}(\{f, g, \delta_0, \delta_1\})$ is #P-hard.*

Proof. If g does not have affine support, we are done by Lemma 11. So suppose that g has affine support. Since g is not pure affine, the range of g contains at least two nonzero values.

The high-level idea will be to use pinning and bisection to extract a nontrivial unary weight function U_λ from g. Then we can reduce from $\#\mathsf{CSP}(\{f, \delta_0, \delta_1, U_\lambda\})$, which we proved #P-hard in Lemma 15.

Look at the relation R_g, viewed as a table. If every column were constant, then g would be pure affine, so this is not the case. Select a nonconstant column with index h. If there are two nonzero values in the range of g amongst the rows of R_g that are 0 in column h, then we derive a new function g' by pinning column h to 0. The new function g' is not pure affine, since the two nonzero values prevent this. So we will show inductively that $\#\mathsf{CSP}(\{f, g', \delta_0, \delta_1\})$ is #P-hard. This will give the result since $\#\mathsf{CSP}(\{f, g', \delta_0, \delta_1\})$ trivially reduces to $\#\mathsf{CSP}(\{f, g, \delta_0, \delta_1\})$.

If we don't finish this way, or symmetrically by pinning column h to 1, then we know that there are distinct positive values w_0 and w_1 such that, for every row x of R_g with 0 in column h, $g(x) = w_0$ and, for every row x of R_g with 1 in column h, $g(x) = w_1$. Now note that, because the underlying relation R_g is affine, it has the same number of 0's in column h as 1's. This is because R_g is the solution of a set of linear equations. Adding the equation $x_h = 0$ or $x_h = 1$ exactly halves the set of solutions in either case. We now project onto the index set $\{h\}$. We obtain the unary weight function U_λ, with $\lambda = w_1/w_0$, on using the earlier observation about proportional functions. This was our goal and completes the proof. □

Lemma 5 now follows from Lemmas 8 and 16, completing the proof of Theorem 4.

Appendix. The purpose of this appendix is to prove Lemma 7 for an arbitrary fixed domain $[q]$. We used only the special case $q = 2$, which we stated and proved as Lemma 8. However, pinning appears to be a useful technique for studying the complexity of #CSP, so we give a proof of the general Lemma 7, which we believe will be applicable elsewhere.

In order to prove the lemma, we introduce a useful, but less natural, variant of #CSP. Suppose $\mathcal{F} \subseteq \mathcal{F}_q$. An instance I of #$\mathsf{CSP}^{\neq}(\mathcal{F})$ consists of a set V of variables and a set $\mathcal{C}$ of constraints, just like an instance of #$\mathsf{CSP}(\mathcal{F})$. In addition, the instance may contain a *single* extra constraint C applying the arity-q *disequality* relation $\chi_{\neq}$ with scope $(v_{C,1}, \dots, v_{C,q})$.

The disequality relation $\chi_{\neq}$ is defined by $\chi_{\neq}(x_1, \dots, x_q) = 1$ if $x_1, \dots, x_q \in [q]$ are pairwise distinct, that is, if they are a permutation of the domain $[q]$. Otherwise, $\chi_{\neq}(x_1, \dots, x_q) = 0$.

Lemma 7 follows immediately from Lemmas 17 and 18 below.

LEMMA 17. *For every* $\mathcal{F} \subseteq \mathcal{F}_q$, #$\mathsf{CSP}(\mathcal{F} \cup \bigcup_{c\in[q]} \delta_c) \leq_{\mathrm{T}}$ #$\mathsf{CSP}^{\neq}(\mathcal{F})$.

Proof. We follow the proof lines of Lemma 8, but instead of subtracting the contribution corresponding to configurations in which some t_i's get the same value, we use the disequality relation to restrict the partition function to configurations in which they get distinct values.

Say that $\mathcal{F}$ is *symmetric* if it is the case that for every arity-k function $f \in \mathcal{F}$, every tuple $x \in [q]^k$, and every permutation $\pi : [q] \to [q]$, $f(x_1, \dots, x_k) = f(\pi(x_1), \dots, \pi(x_k))$.

Let I be an instance of #$\mathsf{CSP}(\mathcal{F} \cup \bigcup_{c\in[q]} \delta_c)$ with variable set V. Let V_c be the set of variables $v \in V$ to which the constraint $\delta_c(v)$ is applied. Assume without loss of generality that the sets V_c are pairwise disjoint. Let $V_q = V \setminus \bigcup_{c\in[q]} V_c$. We construct an instance I' of #$\mathsf{CSP}^{\neq}(\mathcal{F})$. The instance has variables $V_q \cup \{t_0, \dots, t_{q-1}\}$. Every constraint C of I involving a function $f \in \mathcal{F}$ corresponds to a constraint C' of I'. Here C' is the same as C except that variables in V_c are replaced with t_c, for each $c \in [q]$. Also, we add a new disequality constraint to the new variables $t_0, \dots, t_{q-1}$.

Case 1. $\mathcal{F}$ is symmetric. By construction, $Z(I') = \sum_{y_0,\dots,y_{q-1}} Z(I' \mid \sigma(t_0) = y_0, \dots, \sigma(t_{q-1}) = y_{q-1})$, where the sum is over all permutations $y_0, \dots, y_{q-1}$ of $[q]$. By symmetry, the summands are all the same, so $Z(I') = q!Z(I' \mid \sigma(t_0) = 0, \dots, \sigma(t_{q-1}) = q-1) = q!Z(I)$.

Case 2. $\mathcal{F}$ is not symmetric. Say that two permutations $\pi_1 : [q] \to [q]$ and $\pi_2 : [q] \to [q]$ are *equivalent* if, for every $f \in \mathcal{F}$ and every tuple $x \in [q]^k$, $f(\pi_1(x_1), \dots, \pi_1(x_k)) = f(\pi_2(x_1), \dots, \pi_2(x_k))$. Partition the permutations $\pi : [q] \to [q]$ into equivalence classes. Let h be the number of equivalence classes and n_i be the size of the ith equivalence class, so $n_1 + \dots + n_h = q!$.[6] Let $\{\pi_1, \dots, \pi_h\}$ be a set of representatives of the equivalence classes with π_1 being the identity. We know that $n_1 \neq q!$ since $\mathcal{F}$ is not symmetric.

For a positive integer ℓ we will now build an instance I'_ℓ by adding new constraints to I'. For each π_i other than π_1 we add constraints as follows. Choose a function $f_i \in \mathcal{F}$

[6]In fact, it can be shown that these equivalence classes are cosets of the symmetry group of f, and hence are of equal size, though we do not use this fact here.

and a tuple y such that $f_i(y_1,\dots,y_k) \neq f_i(\pi_i(y_1),\dots,\pi_i(y_k))$. If $f_i(y_1,\dots,y_k) > f_i(\pi_i(y_1),\dots,\pi_i(y_k))$, then define the k-tuple x^i by $(x_1^i,\dots,x_k^i) = (y_1,\dots,y_k)$. Otherwise, let n be the order of the permutation π_i and let g_r denote $f_i(\pi_i^r(y_1),\dots,\pi_i^r(y_k))$. Since $g_0 < g_1$ and $g_n = g_0$ there exists an $\xi \in \{1,\dots,n-1\}$ such that $g_\xi > g_{\xi+1}$. Let $(x_1^i,\dots,x_k^i) = (\pi^\xi(y_1),\dots,\pi^\xi(y_k))$ so $f_i(x_1^i,\dots,x_k^i) > f_i(\pi_i(x_1^i),\dots,\pi_i(x_k^i))$.

Let w_{ij} denote $f_i(\pi_j(x_1^i),\dots,\pi_j(x_k^i))$ so, since π_1 is the identity, we have just ensured that $w_{i1} > w_{ii}$. Let $s^i = (t_{x_1^i},\dots,t_{x_k^i})$, and let $0 \le z_i \le h$ $(i = 2,\dots,h)$ be positive integers, which we will determine below. Add ℓz_i new constraints to I'_ℓ with relation f_i and scope s^i. Let $\lambda_i = \prod_{\gamma=2}^h w_{\gamma i}^{z_\gamma}$. Note that, given $\sigma(t_0) = \pi_i(0),\dots,\sigma(t_{q-1}) = \pi_i(q-1)$, the contribution to $Z(I'_\ell)$ for the new constraints is

$$\begin{aligned}\prod_{\gamma=2}^h f_\gamma(\sigma(t_{x_1^\gamma}),\dots,\sigma(t_{x_k^\gamma}))^{z_\gamma \ell} &= \prod_{\gamma=2}^h f_\gamma(\pi_i(x_1^\gamma),\dots,\pi_i(x_k^\gamma))^{z_\gamma \ell}\\ &= \prod_{\gamma=2}^h w_{\gamma,i}^{z_\gamma \ell} = \left(\prod_{\gamma=2}^h w_{\gamma,i}^{z_\gamma}\right)^\ell = {\lambda_i}^\ell.\end{aligned}$$

So

$$Z(I'_\ell) = \sum_{i=1}^h n_i\, Z(\, I' \mid \sigma(t_0) = \pi_i(0),\dots,\sigma(t_{q-1}) = \pi_i(q-1))\, \lambda_i^\ell.$$

We have ensured that $\lambda_1 > 0$, since $w_{i1} > w_{ii} \ge 0$, so $w_{i1} > 0$ for all $i = 2,\dots,h$. We now choose the z_i's so that $\lambda_i \neq \lambda_1$ for all $i = 2,\dots,h$. If $w_{\gamma i} = 0$ for any $\gamma = 2,\dots,h$, we have $\lambda_i = 0$ and hence $\lambda_i \neq \lambda_1$. Thus we will assume, without loss of generality, that $w_{\gamma i} > 0$ for all $\gamma = 2,\dots,h$ and $i = 2,\dots,h'$, where $h' \le h$. Then we have

$$\frac{\lambda_i}{\lambda_1} \;=\; \prod_{\gamma=2}^h \left(\frac{w_{\gamma i}}{w_{\gamma 1}}\right)^{z_\gamma} \;=\; e^{\sum_{\gamma=2}^h \alpha_{\gamma i} z_\gamma} \qquad (i = 2,\dots,h'),$$

where $\alpha_{\gamma i} = \ln(w_{\gamma i}/w_{\gamma 1})$. Note that $\alpha_{ii} < 0$, since $w_{ii} < w_{i1}$. We need to find an integer vector $z = (z_2,\dots,z_h)$ so that none of the linear forms $\mathcal{L}_i(z) = \sum_{\gamma=2}^h \alpha_{\gamma i} z_\gamma$ is zero, for $i = 2,\dots,h'$. We do this using a proof method similar to the Schwartz–Zippel lemma. (See, for example, [20].) None of the $\mathcal{L}_i(z)$ is identically zero, since $\alpha_{ii} \neq 0$. Consider the integer vectors $z \in [h]^{h-1}$. At most h^{h-2} of these can make $\mathcal{L}_i(z)$ zero for any i, since the equation $\mathcal{L}_i(z) = 0$ makes z_i a linear function of z_γ $(\gamma \neq i)$. Therefore there are at most $(h'-1)h^{h-2} < h^{h-1}$ such z which make any $\mathcal{L}_i(z)$ zero. Therefore there must be a vector $z \in [h]^{h-1}$ for which none of the $\mathcal{L}_i(z)$ is zero, and this is the vector we require.

Now, by combining terms with equal λ_i and ignoring terms with $\lambda_i = 0$, we can view $Z(I'_\ell)$ as a sum $Z(I'_\ell) = \sum_i c_i \lambda_i^\ell$, where the λ_i's are positive and pairwise distinct and

$$c_1 = n_1 Z(I' \mid \sigma(t_0) = 0,\dots,\sigma(t_{q-1}) = q-1).$$

Thus, by Lemma 3.2 of [8] we can interpolate to recover c_1. Dividing by n_1, we get

$$Z(I' \mid \sigma(t_0) = 0,\dots,\sigma(t_{q-1}) = q-1) = Z(I). \qquad \square$$

LEMMA 18. *For every* $\mathcal{F} \subseteq \mathcal{F}_q$, $\#\mathsf{CSP}^{\neq}(\mathcal{F}) \leq_{\mathrm{T}} \#\mathsf{CSP}(\mathcal{F})$.

Proof. We use Möbius inversion for posets, following the lines of the proof of [2, Theorem 8].[7] Consider the set of partitions of $[q]$. Let $\underline{0}$ denote the partition with q singleton classes. Consider the partial order in which $\eta \leq \theta$ if and only if every class of η is a subset of some class of θ. Define $\mu(\underline{0}) = 1$, and for any $\theta \neq \underline{0}$ define $\mu(\theta) = -\sum_{\eta \leq \theta, \eta \neq \theta} \mu(\eta)$. Consider the sum $\sum_{\eta \leq \theta} \mu(\eta)$. Clearly, this sum is 1 if $\theta = \underline{0}$. From the definition of μ, it is also easy to see that the sum is 0 otherwise, since

$$\sum_{\eta \leq \theta} \mu(\eta) = \mu(\theta) + \sum_{\eta \leq \theta, \eta \neq \theta} \mu(\eta) = 0.$$

Now let I be an instance of $\#\mathsf{CSP}^{\neq}(\mathcal{F})$ with a disequality constraint applied to variables $t_0, \dots, t_{q-1}$. Let V be the set of variables of I. Given a configuration $\sigma : V \to [q]$, let $\vartheta(\sigma)$ be the partition of $[q]$ induced by of $(\sigma(t_0), \dots, \sigma(t_{q-1}))$. Thus i and j in $[q]$ are in the same class of $\vartheta(\sigma)$ if and only if $\sigma(t_i) = \sigma(t_j)$. We say that a partition η is consistent with σ (written $\eta \preccurlyeq \sigma$) if $\eta \leq \vartheta(\sigma)$. Note that $\eta \preccurlyeq \sigma$ means that for any i and j in the same class of η, $\sigma(t_i) = \sigma(t_j)$.

Let Ω be the set of configurations σ that satisfy all constraints in I except possibly the disequality constraint. Then $Z(I) = \sum_{\sigma \in \Omega} w(\sigma) \mathbb{1}_\sigma$, where $\mathbb{1}_\sigma = 1$ if σ respects the disequality constraint, meaning that $\vartheta(\sigma) = \underline{0}$, and $\mathbb{1}_\sigma = 0$ otherwise. By the Möbius inversion formula derived above,

$$Z(I) \ = \sum_{\sigma \in \Omega} w(\sigma) \sum_{\eta \leq \vartheta(\sigma)} \mu(\eta).$$

Changing the order of summation, we get

$$Z(I) \ = \sum_{\eta} \mu(\eta) \sum_{\eta \leq \theta} \ \sum_{\sigma \in \Omega : \vartheta(\sigma) = \theta} w(\sigma) \ = \sum_{\eta} \mu(\eta) \sum_{\sigma \in \Omega : \eta \preccurlyeq \sigma} w(\sigma).$$

Now note that $\sum_{\sigma : \eta \preccurlyeq \sigma} w(\sigma)$ is the partition function $Z(I_\eta)$ of an instance I_η of $\#\mathsf{CSP}(\mathcal{F})$. The instance I_η is formed from I by ignoring the disequality constraint and identifying variables in $t_0, \dots, t_{q-1}$ whose indices are in the same class of η. Thus we can compute all the $Z(I_\eta)$ in $\#\mathsf{CSP}(\mathcal{F})$. Finally, $Z(I) = \sum_\eta \mu(\eta) Z(I_\eta)$, completing the reduction. ∎

REFERENCES

[1] G. Brightwell and P. Winkler, *Graph homomorphisms and phase transitions*, J. Combin. Theory Ser. B, 77 (1999), pp. 221–262.

[2] A. Bulatov and V. Dalmau, *Towards a dichotomy theorem for the counting constraint satisfaction problem*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 2003, pp. 562–573.

[3] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin, *The complexity of soft constraint satisfaction*, Artificial Intelligence, 170 (2006), pp. 983–1016.

[4] A. Bulatov and M. Grohe, *The complexity of partition functions*, Theoret. Comput. Sci., 348 (2005), pp. 148–186.

[5] B. Cipra, *An introduction to the Ising model*, Amer. Math. Monthly, 94 (1987), pp. 937–959.

[6] N. Creignou and M. Hermann, *Complexity of generalized satisfiability counting problems*, Inform. and Comput., 125 (1996), pp. 1–12.

[7] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monogr. Discrete Math. Appl. 7, SIAM, Philadelphia, 2001.

[7] Lovász [14] had previously used Möbius inversion in a similar context.

[8] M. Dyer and C. Greenhill, *The complexity of counting graph homomorphisms*, Random Structures Algorithms, 17 (2000), pp. 260–289.

[9] M. Dyer, L. A. Goldberg, and M. Paterson, *On counting homomorphisms to directed acyclic graphs*, in Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 4051, Springer, New York, 2006, pp. 38–49.

[10] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104.

[11] L. A. Goldberg and M. Jerrum, *Inapproximability of the Tutte polynomial*, Inform. Comput., 206 (2008), pp. 908–929.

[12] C. Greenhill, *The complexity of counting colourings and independent sets in sparse graphs and hypergraphs*, Comput. Complexity, 9 (2000), pp. 52–72.

[13] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford University Press, London, 2004.

[14] L. Lovász, *Operations with structures*, Acta Math. Hungarica, 18 (1967), pp. 321–328.

[15] R. Ladner, *On the structure of polynomial time reducibility*, J. ACM, 22 (1975), pp. 155–171.

[16] C. Papadimitriou, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.

[17] F. Rossi, P. van Beek, and T. Walsh, eds., *Handbook of Constraint Programming*, Elsevier, New York, 2006.

[18] T. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1978, pp. 216–226.

[19] A. Scott and G. Sorkin, *Polynomial Constraint Satisfaction: A Framework for Counting and Sampling CSPs and Other Problems*, online at http://arxiv.org/abs/cs/0604079.

[20] J. Schwartz, *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM, 27 (1980), pp. 701–717.

[21] L. G. Valiant, *The complexity of enumeration and reliability problems*, SIAM J. Comput., 8 (1979), pp. 410–421.

[22] D. Welsh, *Complexity: Knots, Colourings and Counting*, LMS Lecture Note Ser. 186, Cambridge University Press, Cambridge, UK, 1993.