The Complexity of Weighted Boolean #CSP

Martin Dyer
School of Computing
University of Leeds
Leeds LS2 9JT, UK

Leslie Ann Goldberg
Department of Computer Science,
University of Liverpool,
Liverpool L69 3BX, UK

Mark Jerrum
School of Mathematical Sciences,
Queen Mary, University of London
Mile End Road, London E1 4NS, UK

28 April 2008

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 parameterised by a finite set \( F \) of non-negative 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 \( \text{FP}\#P \)-complete unless either (1) every function in \( F \) is of “product type”, or (2) every function in \( F \) is “pure affine”. In the remaining cases, computing the partition function is in \( \mathsf{P} \).

1 Introduction

This paper gives a dichotomy theorem for the complexity of the partition function of weighted Boolean constraint satisfaction problems. Such problems are parameterised by a set \( F \) of non-negative 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 \( F \) 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 a generalisation 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 \[23\] 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 colouring and the hard-core version of the ferromagnetic Potts model is trivial — acceptable configurations colour the entire graph with a single colour.) A corresponding weighted version of the decision CSP was investigated by Cohen, Cooper, Jeavons and Krokhin \[3\]. This results in optimisation problems.
We use \(\#\text{CSP}(\mathcal{F})\) 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 Theorem 3 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 \(\#\text{CSP}(\mathcal{F})\) instance is complete for the class \(\text{FP}^\#\text{P}\). 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 modelling decision problems, and has many practical applications. (See, for example [15].) 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 constraint satisfaction problems has become a major and active area within computational complexity [7][14].

A Constraint Satisfaction Problem (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* \(\Gamma\) 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 \(\Gamma\), 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 \(\Gamma\) of the same arity, which constrains the variables in the scope. A *configuration* \(\sigma\) is a function from \(V\) to \([q]\). The configuration \(\sigma\) 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 \(\sigma\) 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 \(\Gamma\), the *decision problem* \(\text{CSP}(\Gamma)\) asks us to determine whether any configuration is satisfying. The *counting problem* \(\#\text{CSP}(\Gamma)\) asks us to determine the *number* of (distinct) satisfying configurations.

Varying the constraint language \(\Gamma\) defines the classes \(\text{CSP}\) and \(\#\text{CSP}\) 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 \(\text{CSP}(\Gamma)\) is the classical 2-Satisfiability problem, which is in \(\text{P}\). On the other hand, there is a similar constraint language \(\Gamma'\) with four relations of arity 3 such that 3-Satisfiability (which is \(\text{NP}\)-complete) can be represented in \(\text{CSP}(\Gamma')\). It may happen that the counting problem is harder than the decision problem. If \(\Gamma\) is the constraint language of 2-Satisfiability above, then \(\#\text{CSP}(\Gamma)\) contains the problem of counting independent sets in graph, and is \(\#\text{P}\)-complete [22], even if restricted to 3-regular graphs [12].

\(^1\) 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.
Any decision problem \( \text{CSP}(\Gamma) \) is in \( \text{NP} \), but not every problem in \( \text{NP} \) 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 \( \text{P} \neq \text{NP} \), then there are problems which are neither in \( \text{P} \) nor \( \text{NP} \)-complete \[16\]. 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 which are in \( \text{P} \) and those which are \( \text{NP} \)-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 \( \text{CSP}(\Gamma) \) is in \( \text{P} \) for some constraint languages \( \Gamma \), and \( \text{CSP}(\Gamma) \) is \( \text{NP} \)-complete for all other constraint languages \( \Gamma \). This conjecture appeared in a seminal paper of Feder and Vardi \[10\], but has not yet been proved.

A similar dichotomy, between \( \text{FP} \) and \( \text{#P} \)-complete, is conjectured for \( \text{#CSP} \) \[2\]. The complexity classes \( \text{FP} \) and \( \text{#P} \) are the analogues of \( \text{P} \) and \( \text{NP} \) for counting problems. \( \text{FP} \) is simply the class of functions computable in deterministic polynomial time. \( \text{#P} \) is the class of integer functions that can be expressed as the number of accepting computations of a polynomial-time non-deterministic Turing machine. Completeness in \( \text{#P} \) is defined with respect to polynomial-time Turing reducibility \[17\, \text{Chap. 18}\]. Bulatov and Dalmau \[2\] have shown in one direction that, if \( \text{#CSP}(\Gamma) \) is solvable in polynomial time, then the constraints in \( \Gamma \) must have certain algebraic properties (assuming \( \text{P} \neq \text{NP} \)). 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 characterisation.

The conjectured dichotomies for CSP and \( \text{#CSP} \) are major open problems for computational complexity theory. There have been many important results for subclasses of CSP and \( \text{#CSP} \). We mention the most relevant to our paper here. The first decision dichotomy was that of Schaefer \[19\], for the Boolean domain \( \{0,1\} \). Schaefer’s result is as follows.

**Theorem 1 (Schaefer \[19\]).** Let \( \Gamma \) be a constraint language with domain \( \{0,1\} \). The problem \( \text{CSP}(\Gamma) \) is in \( \text{P} \) if \( \Gamma \) satisfies one of the conditions below. Otherwise, \( \text{CSP}(\Gamma) \) is \( \text{NP} \)-complete.

1. \( \Gamma \) is 0-valid or 1-valid.
2. \( \Gamma \) is weakly positive or weakly negative.
3. \( \Gamma \) is affine.
4. \( \Gamma \) is bijunctive.

We will not give detailed definitions of the conditions in Theorem 1, but the interested reader is referred to the paper \[19\] or to Theorem 6.2 of the textbook \[7\]. An interesting feature is that the conditions in \[7\, \text{Theorem 6.2}\] are all checkable. That is, there is an algorithm to determine whether \( \text{CSP}(\Gamma) \) is in \( \text{P} \) or \( \text{NP} \)-complete, given a constraint language \( \Gamma \) 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 \( \Gamma \) be a constraint language with domain \( \{0,1\} \). The problem \( \text{#CSP}(\Gamma) \) is in \( \text{FP} \) if \( \Gamma \) is affine. Otherwise, \( \text{#CSP}(\Gamma) \) is \( \text{#P} \)-complete.

A constraint language \( \Gamma \) 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 \( \text{GF}(2) \). 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 iff \( a, b, c \in R \) implies \( d = a \oplus b \oplus c \in R \). (We will use this characterisation below.) There is an algorithm for determining whether a Boolean constraint language \( \Gamma \) is affine, so there is an algorithm for determining whether \(#CSP(\Gamma)\) is in \( \text{FP} \) or \(#\text{P}\)-complete.

### 1.2 Weighted \(#CSP\)

The weighted framework of \([4]\) extends naturally to Constraint Satisfaction Problems. 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 non-negative 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 \( x = (x_1, \ldots, x_k) \in R_f \). The entry in row \( x \) and column \( j \) is \( x_j \), which is a value in \([q]\).

A weighted \(#CSP\) problem is parameterised by a finite subset \( \mathcal{F} \) of \( \mathcal{F}_q \), and will be denoted by \(#CSP(\mathcal{F})\). An instance \( I \) of \(#CSP(\mathcal{F})\) 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 \( s_C = (v_{C,1}, \ldots, v_{C,k_C}) \) of variables from \( V \). The variables \( v_{C,1}, \ldots, v_{C,k_C} \) need not be distinct. As in the unweighted case, a configuration \( \sigma \) for the instance \( I \) is a function from \( V \) to \([q]\). The weight of the configuration \( \sigma \) is given by

\[
w(\sigma) = \prod_{C \in \mathcal{C}} f_C(\sigma(v_{C,1}), \ldots, \sigma(v_{C,k_C})).
\]

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

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

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

Note that an (unweighted) CSP counting problem \(#CSP(\Gamma)\) 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 \(#CSP(\Gamma)\), 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 square 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 \(#CSP(\{h\})\).

---

\(^2\)We assume \( 0 \in \mathbb{N} \), so we allow non-negative constants.
Bulatov and Grohe [4] call $H$ connected if $H$ is connected and bipartite if $H$ is bipartite. They give the following dichotomy theorem for non-negative $H$.

**Theorem 3** (Bulatov and Grohe [4]). Let $H$ be a symmetric matrix with non-negative rational entries.

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 #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 #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 #P-hard.

Many partition functions arising in statistical physics may be viewed as weighted #CSP problems. An example is the $q$-state Potts model (which is, in fact, a weighted graph homomorphism problem). In general, weighted #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 [20].

### 1.3 Some Notation

We will call the class of (rational) weighted #CSP problems weighted #CSP. The sub-class having domain size $q = 2$ will be called weighted Boolean #CSP, and will be the main focus of this paper. We will give a dichotomy theorem for weighted Boolean #CSP.

Since weights can be arbitrary non-negative rational numbers, the solution to these problems is not an integer in general. Therefore #CSP($F$) is not necessarily in the class #P. However, Goldberg and Jerrum [11] have observed that $Z(I) = \tilde{Z}(I)/K(I)$, where $\tilde{Z}$ is a function in #P and $K(I)$ is a positive integer computable in FP. This follows because, for all $f \in F$, we can ensure that $f(\cdot) = \tilde{f}(\cdot)/K(I)$, where $\tilde{f}(\cdot) \in \mathbb{N}$, by “clearing denominators”.$^3$. The denominator $K(I)$ can obviously be computed in polynomial time, and it is straightforward to show that computing $\tilde{Z}(I)$ is in #P, so the characterisation of [11] follows. The resulting complexity class, comprising functions which are a function in #P divided by a function in FP, is named #P$_Q$ in [11], where it is used in the context of approximate counting. Clearly we have

$$\text{weighted #CSP} \subseteq \text{#P}_Q \subseteq \text{FP}^{\text{#P}}.$$  

On the other hand, if $Z(I) \in \text{weighted #CSP}$ is #P-hard, then, using an oracle for computing $Z(I)$, we can construct a #P oracle $\tilde{Z}(I)$ as outlined above. (Note that $Z(I) \notin \text{#P}$ in general.) Using this, we can compute any function in FP$^{\text{#P}}$ with a polynomial time-bounded oracle Turing machine. Thus any #P-hard function in weighted #CSP is complete for FP$^{\text{#P}}$.

We will use this observation to state our main result in terms of completeness for the class FP$^{\text{#P}}$.

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 #CSP($\mathcal{F} \cup \{f\}$), there is a polynomial time computable instance $I'$ of #CSP($\mathcal{F}$), such that $Z(I) = \varphi(I)Z(I')$ for some $\varphi(I) \in \mathbb{Q}$ which is FP-computable. This generalises the notion of parsimonious reduction [17] among problems in

---

$^3$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.
#P. We will use $\leq_T$ to denote the relation “is polynomial-time Turing-reducible to” between computational problems. Clearly, if $F$ simulates $f$, we have $\#\text{CSP}(F \cup \{f\}) \leq_T \#\text{CSP}(F)$. 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 $c,d \in [q]$.

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 $f \in F_2$ 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 $\{0, 1\}$ 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 $\text{GF}(2)$.

### 1.4 Our Result

Our main result is the following.

**Theorem 4.** Suppose $F \subseteq F_2 = \{ f : \{0, 1\}^k \rightarrow \mathbb{Q}^+ \mid k \in \mathbb{N}\}$. If every function in $F$ is of product type then $\#\text{CSP}(F)$ is in FP. If every function in $F$ is pure affine then $\#\text{CSP}(F)$ is in NP. Otherwise, $\#\text{CSP}(F)$ is $\text{FP}^{\#\text{P}}$-complete.

**Proof.** Suppose first that $F$ 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 $\sigma$ 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 $\alpha$, to the partition function if variables in $U_1$ are given value “0” by $\sigma$ and it contributes another weight, say $\beta$, to the partition function if variables in $U_1$ are given value “1” by $\sigma$. 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 $F$ contains only unary, equality or binary disequality constraints, we can also remove all functions involving variables in $U_1 \cup U_2$ to give $F'$. Then $I'$ is of product type with fewer variables, so we may compute $Z(I')$ recursively.

Suppose second that $F$ is pure affine. Then $Z(I) = \prod_{f \in F} w_f^{k_f} Z(I')$, 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 $\text{GF}(2)$, as Creignou and Hermann have observed.

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

**Lemma 5.** If $f \in F_2$ is not of product type and $g \in F_2$ is not pure affine then $\#\text{CSP}(\{f, g\})$ is $\#\text{P}$-hard.

---

4A more general disequality function is defined in the Appendix.
Note that the functions $f$ and $g$ in Lemma 5 may be one and the same function. So $\#\text{CSP}(\{f\})$ is $\#P$-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 $\#\text{CSP}$

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 $Z(I | \sigma(u_1) = c_1, \ldots, \sigma(u_k) = c_k)$ denote the contribution to $Z(I)$ from assignments $\sigma$ 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.

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 $j_1 < \cdots < j_r$. We say that a $k$-tuple $x' \in [q]^k$ extends an $r$-tuple $x \in [q]^r$ on $J$ (written $x' \supseteq_J x$) 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 $x \in [q]^k$, $g(x) = \sum_{x' \supseteq_J x} f(x')$.

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 $\#\text{CSP}(\mathcal{F} \cup \{g\}) \leq_T \#\text{CSP}(\mathcal{F})$.

**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 $\#\text{CSP}(\mathcal{F} \cup \{g\})$. We will construct an instance $I'$ of $\#\text{CSP}(\mathcal{F})$ 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 $jg$ is the $\ell$th element of $J$, then $v'_{C',jg} = v_{C,\ell}$. The other variables, $v_{C',j}$ ($j \notin J$), are distinct new variables. We have shown that $\mathcal{F}$ simulates $g$ with $\phi(I) = 1$. \hfill $\square$

2.3 Pinning

For $c \in [q]$, $\delta_c$ 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, generalises 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 $H$.

\footnote{It is not necessary to choose this particular ordering for $J$, but it is convenient to do so.}
Lemma 7. For every \( F \subseteq F_q \), \( \#\text{CSP}(F \cup \bigcup_{c \in [q]} \delta_c) \leq_T \#\text{CSP}(F) \).

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

Lemma 8. For every \( F \subseteq F_2 \), \( \#\text{CSP}(F \cup \{\delta_0, \delta_1\}) \leq_T \#\text{CSP}(F) \).

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

Given an instance \( I \) of \( \#\text{CSP}(F \cup \{\delta_0, \delta_1\}) \) with variable set \( V \) we consider two instances \( I' \) and \( I'' \) of \( \#\text{CSP}(F) \). 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 \( \#\text{CSP}(F) \).) Let \( V_2 = V \setminus (V_0 \cup V_1) \).

The instance \( I' \) has variables \( V_0 \cup \{t_0, t_1\} \) where \( t_0 \) and \( t_1 \) are distinct new variables that are not in \( V \). Every constraint \( C \) of \( I \) involving a function \( f \in F \) 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 \( f \in F \) 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 \( t \).

Case 1. \( F \) is symmetric: By construction,

\[
Z(I') - Z(I'') = Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 0).
\]

By symmetry, the summands are the same, so

\[
Z(I') - Z(I'') = 2Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1) = 2Z(I).
\]

Case 2. \( F \) is not symmetric: Let \( f \) be an arity-\( k \) function in \( F \) 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

\[
Z(I'_x) = Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}) + Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 0)f(0, \ldots, 0) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 1)f(1, \ldots, 1) = Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}) + Z(I''_x).
\]

Thus we have two independent equations,

\[
Z(I'_x) - Z(I''_x) = Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1)f(x) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 0)f(\overline{x}),
\]

\[
Z(I') - Z(I'') = Z(I' | \sigma(t_0) = 0, \sigma(t_1) = 1) + Z(I' | \sigma(t_0) = 1, \sigma(t_1) = 0),
\]

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

Corollary 9 (Bulatov and Grohe). Let $H$ be a symmetric $2 \times 2$ matrix with non-negative 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 $\Upsilon$ generated by the rows of $A$ over GF(2). For any real number $\lambda$, 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 [23]). We could not find a proof, so we provide one here. We restrict attention to positive $\lambda$, 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 $c_j = \bigoplus_{e \in U} b_{ij}$, 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 $\lambda^k$, where $k$ is the number of edges in the cut $U, V \setminus U$. Thus $W_A(\lambda) = \frac{1}{Z_H(G)} Z_H(G)$, where $H$ is the symmetric weight matrix with $H_{11} = H_{22} = 1$ and $H_{12} = H_{21} = \lambda$. The $\frac{1}{Z_H(G)}$ 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 generalisation of a result of Creignou and Hermann [2], 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 \#CSP($\mathcal{F}$) is \#P-hard when $\mathcal{F}$ contains a single function. A very different situation arises when \#CSP($\{f\}$) and \#CSP($\{g\}$) are in FP but \#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 \#CSP($\{f\}$) is \#P-hard.

Proof. Let $k$ be the arity of $f$, and let us denote the $i$th 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 $R_f = \{(\alpha, \beta), (\tilde{\alpha}, \beta), (\alpha, \tilde{\beta})\}$ for some $\alpha \in \{0, 1\}$ and $\beta \in \{0, 1\}$. We can show that $\#\text{CSP}\{\{f\}\}$ is $\#\text{P}$-hard by reduction from $\text{Eval}(H)$ 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 $\text{Eval}(H)$ we construct an instance $I$ of $\#\text{CSP}\{\{f\}\}$ as follows. The variables of $I$ are the vertices of $G$. For each edge $(u, v)$ of $G$, add a constraint with function $f$ and variable sequence $u, v$. Corollary \[\text{[8]}\] now tells us that $\text{Eval}(H)$ is $\#\text{P}$-hard, so $\#\text{CSP}\{\{f\}\}$ is $\#\text{P}$-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 $i'$th position to $\alpha$. 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 \[\text{[9]}\]). Note that $\#\text{CSP}\{\{f^{i=\alpha}\}\} \leq_T \#\text{CSP}\{\{f, \delta_0, \delta_1\}\}$, since $f^{i=\alpha}$ can obviously be simulated by $\{f, \delta_0, \delta_1\}$. Furthermore, by Lemma \[\text{[8]}\] $\#\text{CSP}\{\{f, \delta_0, \delta_1\}\} \leq_T \#\text{CSP}\{\{f\}\}$. Thus, we can assume that $f^{i=\alpha}$ has affine support — otherwise, we are finished by induction. Similarly, by Lemma \[\text{[8]}\] $\#\text{CSP}\{\{f^{i=*}\}\} \leq_T \#\text{CSP}\{\{f\}\}$. 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 4 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, $\#\text{CSP}\{\{f'\}\}$ is $\#\text{P}$-hard. Now note that $\#\text{CSP}\{\{f^i\}\} \leq_T \#\text{CSP}\{\{f\}\}$.

To see this, note that any instance $I_1$ of $\#\text{CSP}\{\{f'\}\}$ can be turned into an instance $I$ of $\#\text{CSP}\{\{f\}\}$ by repeating the first variable in the sequence of variables for each constraint.

**Case 2:** There is an index $1 \leq i \leq k$ such that $a_i = b_i = c_i$: Since $d$ is not in $R_f$ and $d_i = a_i$, we find that $f^{i=\alpha}$ 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 Case 4 below, where $\{\alpha, \beta\} \in \{0, 1\}$, $\alpha = 1 - \alpha$, $\beta = 1 - \beta$ and $a', b', c' \in \{0, 1\}^{k-2}$.

**Case 3:** $a = (\tilde{\alpha}, \beta, a')$, $b = (\alpha, \beta, b')$, $c = (\alpha, \beta, c')$: Since $R_{f_1=\tilde{\alpha}}$ is affine and $a, b$ and $c$ are in $R_f$, we must have either $d = (\alpha, \beta, d') \in R_f$ or $e = (\tilde{\alpha}, \beta, d') \in R_f$, 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 $R_{f_2=\tilde{\beta}}$ is affine, we may assume that $g = (\alpha, \tilde{\beta}, d') \in R_f$. Since $R_{f_1=\tilde{\alpha}}$ and $a, b$ and $c$ are in $R_f$, we find that $h = a \oplus b \oplus c = (\tilde{\alpha}, \beta, c') \in R_f$. Since $R_{f_2=\tilde{\beta}}$ is affine and $a, c$ and $g$ are in $R_f$, we find that $i = (\tilde{\alpha}, \beta, b') \in R_f$. Also, since $R_{f_2=\tilde{\beta}}$ is affine and $a, h$ and $i$ are in $R_f$, we find that $j = (\tilde{\alpha}, \beta, d') \in R_f$. 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 $f'(r_1, r_2) \in R_f$. Thus, $f$ does not have affine support and $\#\text{CSP}\{\{f'\}\}$ is $\#\text{P}$-hard by induction. Also, $\#\text{CSP}\{\{f'\}\} \leq_T \#\text{CSP}\{\{f\}\}$ by Lemma \[\text{[8]}\].

**Case 4:** $a = (\tilde{\alpha}, \alpha, a')$, $b = (\alpha, \alpha, b')$, $c = (\alpha, \alpha, c')$: Since $R_{f_1=\tilde{\alpha}}$ is affine and $a, b$ and $c$ are in $R_f$ but $d$ is not, we have $e = (\alpha, \tilde{\alpha}, d') \in R_f$. Similarly, since $R_{f_2=\alpha}$ 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 $R_{f_1=\alpha}$ is affine and $a, b$ and $e$ are in $R_f$, we have $h = (\tilde{\alpha}, \tilde{\alpha}, c') \in R_f$. Also, since $R_{f_2=\alpha}$ is affine and $a, b$ and $g$ are in
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 11 showed that $\#\text{CSP} \{\{f\}\}$ is $\#\text{P}$-hard when $f$ does not have affine support. The following lemma gives another (rather technical, but useful) condition which implies that $\#\text{CSP} \{\{f\}\}$ is $\#\text{P}$-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 $\lambda_i$ such that, for all $y \in \{0, 1\}^{k-1}$,

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

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 $\#\text{CSP} \{\{f\}\}$ is $\#\text{P}$-hard.

**Proof.** We’ll use Corollary 9 to prove hardness, following an argument from [9]. Choose a useful index $i$ so that there is no $\lambda_i$ 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 Eval($A'$) is $\#\text{P}$-hard. Note that $A'$ is the following symmetric $2 \times 2$ matrix with non-negative 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 Eval($A'$) is $\#\text{P}$-hard by Corollary 9 we just need to show that its determinant is non-zero. By Cauchy-Schwartz, the determinant is non-negative, and is zero only if $\lambda_i$ exists, which have assumed not to be the case. Thus Eval($A'$) is $\#\text{P}$-hard by Corollary 9.

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

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

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 $\delta_1$ and the constant 0. We can discard $\delta_1$ 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 not add to the complexity of the problem.

Thus $U_w$ will be called nontrivial if $w \notin \{0, 1\}$. Let $\bigoplus_k : \{0, 1\}^k \to \{0, 1\}$ be the arity-$k$ parity function that is 1 iff its argument has an odd number of 1s. Let $\neg \bigoplus_k : \{0, 1\}^k \to \{0, 1\}$ be the function $1 - \bigoplus_k$. The following lemma shows that even a simple function like $\bigoplus_3$ can lead to intractable #CSP instances when it is combined with a nontrivial weight function $U_\lambda$.

**Lemma 13.** #CSP($\bigoplus_3, U_\lambda, \delta_0, \delta_1$) and #CSP($\neg \bigoplus_3, U_\lambda, \delta_0, \delta_1$) 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 $\bigoplus_k$, $\delta_0$, etc., as relations as well as functions to $\{0, 1\}$.

We first argue that for any $k$, the relation $\bigoplus_k$ can be simulated by $\{\bigoplus_3, \delta_0, \delta_1\}$. 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 $\{\bigoplus_3, \delta_0, \delta_1\}$ can be used to simulate any relation $\bigoplus_k$, we can use $\{\bigoplus_3, \delta_0, \delta_1\}$ to simulate an arbitrary system of linear equations over GF(2). In particular we can use them to simulate the subspace $\Upsilon$ of code words for a given generating matrix $A$.

Finally, we can use $U_\lambda$ to simulate the function which evaluates the weight enumerator on $\Upsilon$. Then, since $\lambda \neq 0, 1$, we can apply Lemma $[10]$ to complete the argument. The same proof, with minor modifications, applies to $\neg \bigoplus_3$. \hfill $\Box$

**Lemma 14.** Suppose $f \in \mathcal{F}_2$ is not of product type. Then, for any positive $\lambda \neq 1$, there exists a constant $c$, depending on $f$, such that #CSP($\{f, \delta_0, \delta_1, U_\lambda, U_c\}$) is #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 iff 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 $R$ is the complete relation of arity $k$.**

Let $f^*$ be the projection of $f$ onto the $k$ columns of $R$. By Lemma $[6]$,

$$\text{#CSP}(\{f^*\}) \leq \text{T} \ \text{CSP}(\{f\}) \leq \text{T} \ \text{CSP}(\{f, \delta_0, \delta_1, U_\lambda, U_c\}).$$

We will argue that #CSP($\{f^*\}$) is #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 $[12]$.\[12\]

**Case 2: Suppose $R$ 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 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 $R''$ must be $\oplus_3$ or $\neg \oplus_3$.

Now define the symmetric function $f'''$ by

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

Note that $R'''$ is $\oplus_3$ or $\neg \oplus_3$, since $R''$ is symmetric and hence $R'''(x) = f'''(y)$.

To summarise: using $f$ and the constant functions $\delta_0$ and $\delta_1$, we have simulated a function $f'''$ such that its underlying relation $R'''$ is either $\oplus_3$ or $\neg \oplus_3$. Furthermore, if triples $x$ and $y$ have the same number of 1s 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 $g(0, 1, 1) = g(1, 0, 1) = g(1, 1, 0) = c^2w_2 = w_0$. 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 $\#\text{CSP}(\oplus_3, U_\lambda, \delta_0, \delta_1)$ or $\#\text{CSP}(\neg \oplus_3, U_\lambda, \delta_0, \delta_1)$, which are both $\#P$-hard by Lemma 13.

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

**Proof.** Take an instance $I$ of $\#\text{CSP}(\{f, \delta_0, \delta_1, U_\lambda, U_c\})$, from Lemma 13, with $n$ variables $x_1, x_2, \ldots, x_n$. We want to compute the partition function $Z(I)$ using only instances of $\#\text{CSP}(\{f, \delta_0, \delta_1, U_\lambda\})$. 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

$$Z(I; w) = \sum_{\sigma \in \{0, 1\}^n} \hat{Z}(\sigma) \prod_{i: \sigma_i = 1} w_i = \sum_{\sigma \in \{0, 1\}^n} \hat{Z}(\sigma) w_{\sum_{i=1}^n m_i \sigma_i}.$$
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 $\sigma$, 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 $U_\lambda$ constraint. This evaluation is an instance of $\#\text{CSP}(\{ f, \delta_0, \delta_1, U_\lambda \})$. So, using $m$ different values of $j$ and interpolating, we learn the coefficients of the polynomial $Z(I; w)$. Then we can put $w = c$ to evaluate $Z(I)$.

Lemma 16. Suppose $f \in \mathcal{F}_2$ is not of product type, and $g \in \mathcal{F}_2$ is not pure affine. Then $\#\text{CSP}(\{ f, g, \delta_0, \delta_1 \})$ is $\#\text{P}$-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 non-zero values.

The high-level idea will be to use pinning and bisection to extract a non-trivial unary weight function $U_\lambda$ from $g$. Then we can reduce from $\#\text{CSP}(\{ f, \delta_0, \delta_1, U_\lambda \})$, which we proved $\#\text{P}$-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 non-constant column with index $h$. If there are two non-zero 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 non-zero values prevent this. So we will show inductively that $\#\text{CSP}(\{ f, g', \delta_0, \delta_1 \})$ is $\#\text{P}$-hard. This will give the result since $\#\text{CSP}(\{ f, g', \delta_0, \delta_1 \})$ trivially reduces to $\#\text{CSP}(\{ f, g, \delta_0, \delta_1 \})$.

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 $h$, $g(x) = w_1$. 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 $U_\lambda$, 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 Lemma 8 and Lemma 16, completing the proof of Theorem 4.

References

[1] G. Brightwell and P. Winkler, Graph homomorphisms and phase transitions, *Journal of Combinatorial Theory (Series B)* 77 (1999), 221–262

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

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

[4] A. Bulatov and M. Grohe, The complexity of partition functions, *Theoretical Computer Science* 348 (2005), 148–186.
[5] B. Cipra, An Introduction to the Ising Model, *American Mathematical Monthly* **94** (1987), 937–959.

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

[7] N. Creignou, S. Khanna, M. Sudan, *Complexity classifications of Boolean constraint satisfaction problems*, SIAM Press, 2001.

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

[9] M. Dyer, L.A. Goldberg and M. Paterson, On counting homomorphisms to directed acyclic graphs, in *Proc. 33rd International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **4051**, Springer, 2006, pp. 38–49.

[10] T. Feder and M. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory, *SIAM Journal on Computing* **28** (1999), 57–104.

[11] L.A. Goldberg and M. Jerrum, Inapproximability of the Tutte polynomial, [http://arxiv.org/abs/cs.CC/0605140](http://arxiv.org/abs/cs.CC/0605140), 2006.

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

[13] P. Hell and J. Nešetřil, On the complexity of H-coloring, *Journal of Combinatorial Theory (Series B)* **48** (1990), 92–110.

[14] P. Hell and J. Nešetřil, *Graphs and homomorphisms*, Oxford University Press, 2004.

[15] L. Lovász, Operations with structures, *Acta Mathematica Hungarica* **18** (1967), 321–328.

[16] R. Ladner, On the structure of polynomial time reducibility, *Journal of the Association for Computing Machinery* **22** (1975), 155–171.

[17] C. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994.

[18] F. Rossi, P. van Beek and T. Walsh (Eds.), *Handbook of constraint programming*, Elsevier, 2006.

[19] T. Schaefer, The complexity of satisfiability problems, in *Proc. 10th Annual ACM Symposium on Theory of Computing*, ACM Press, 1978, pp. 216–226.

[20] A. Scott and G. Sorkin, Polynomial constraint satisfaction: a framework for counting and sampling CSPs and other problems, [http://arxiv.org/abs/cs/0604079](http://arxiv.org/abs/cs/0604079).

[21] J. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the Association for Computing Machinery* **27** (1980), 701–717.

[22] L. Valiant, The complexity of enumeration and reliability problems, *SIAM Journal on Computing* **8** (1979), 410–421.

[23] D. Welsh, *Complexity: knots, colourings and counting*, LMS Lecture Note Series, vol. 186, Cambridge University Press, 1993.
4 Appendix

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

Lemma\textsuperscript{7}. For every \( \mathcal{F} \subseteq \mathcal{F}_q \), \( \#\text{CSP}(\mathcal{F} \cup \bigcup_{c \in [q]} \delta_c) \leq_T \#\text{CSP}(\mathcal{F}) \).

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 \#CSP\textsuperscript{\#}(\mathcal{F}) consists of a set \( V \) of variables and a set \( C \) of constraints, just like an instance of \#CSP(\mathcal{F}). In addition, the instance may contain a single extra constraint \( C \) applying the arity-\( q \) disequality relation \( \chi_\neq \) with scope \((v_{C,1}, \ldots, v_{C,q})\).

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

Lemma\textsuperscript{7} follows immediately from Lemma\textsuperscript{17} and \textsuperscript{18} below.

Lemma 17. For every \( \mathcal{F} \subseteq \mathcal{F}_q \), \( \#\text{CSP}(\mathcal{F} \cup \bigcup_{c \in [q]} \delta_c) \leq_T \#\text{CSP}\textsuperscript{\#}(\mathcal{F}) \).

**Proof.** We follow the proof lines of Lemma\textsuperscript{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 \( f \in \mathcal{F} \) and every tuple \( x \in [q]^k \) and every permutation \( \pi : [q] \rightarrow [q], f(x_1, \ldots, x_k) = f(\pi(x_1), \ldots, \pi(x_k)) \).

Let \( I \) be an instance of \#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 \#CSP\textsuperscript{\#}(\mathcal{F}). The instance has variables \( V_q \cup \{t_0, \ldots, t_{q-1}\} \). 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 \( t_0, \ldots, t_{q-1} \).

**Case 1.** \( \mathcal{F} \) is symmetric:

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

**Case 2.** \( \mathcal{F} \) is not symmetric:

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

For a positive integer \( \ell \) we will now build an instance \( I'_\ell \) by adding new constraints to \( I' \). For each \( \pi_1 \) other than \( \pi_1 \) we add constraints as follows. Choose a function \( f_i \in \mathcal{F} \) and a tuple \( y \) such that \( f_i(y_1, \ldots, y_k) \neq f_i(\pi_1(y_1), \ldots, \pi_1(y_k)) \). If \( f_i(y_1, \ldots, y_k) > f_i(\pi_1(y_1), \ldots, \pi_1(y_k)) \) then define the \( k \)-tuple \( x^i \) by \( (x^i_1, \ldots, x^i_k) = (y_1, \ldots, y_k) \). Otherwise, let \( n \) be the order of the

\footnote{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.}
Thus, by Lemma 3.2 of [8] we can interpolate to recover \( \gamma \in \mathbb{Z}_z \) (integer vectors equation \( \ell z \)).

Let \( \xi \) denote \( \mathcal{L}_{i,y} \) so that \( \xi \) is the identity, we have just ensured that \( w_{i,i} > w_{i,i} \). Let \( s^i = (t_{x^i}, t_{x^i}) \), and let \( 0 \leq z_i \leq h \) (i.e., \( i = 2, \ldots, h \)) be positive integers, which we will determine below. Add \( \ell z_i \) new constraints to \( I'_i \) with relation \( f_i \) and scope \( s^i \). Let \( \lambda_i = \prod \gamma=2 \mathcal{L}_{i,y} \). Note that, given \( \sigma(t_0) = \pi_i(0), \ldots, \sigma(t_{q-1}) = \pi_i(q-1) \), the contribution to \( Z(I'_i) \) for the new constraints is

\[
\prod_{\gamma=2}^h f_{i}(\sigma(t_{x_{i}^\gamma}), \ldots, \sigma(t_{x_{i}^{\gamma}}))^{z_{i}\ell} = \prod_{\gamma=2}^h f_{i}(\pi_i(x_{i}^\gamma), \ldots, \pi_i(x_{i}^{\gamma}))^{z_{i}\ell} = \prod_{\gamma=2}^h w_{i,\gamma}^{z_{i}\ell} = \left( \prod_{\gamma=2}^h w_{i,\gamma}^{z_{i}\ell} \right)^{\ell} = \lambda_i^{\ell}.
\]

So

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

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

\[
\frac{\lambda_i}{\lambda_1} = \prod_{\gamma=2}^h \frac{w_{i,i}^{z_{i}\ell}}{w_{i,i}^{z_{i}\ell}} = e^{-\sum_{\gamma=2}^h \alpha_{i} z_{i}\ell} = (i = 2, \ldots, h'),
\]

where \( \alpha_{i} = \ln(w_{i,i}/w_{i,i}) \). Note that \( \alpha_{ii} < 0 \), since \( w_{ii} < w_{i,i} \). We need to find an integer vector \( z = (z_2, \ldots, z_h) \) so that none of the linear forms \( \mathcal{L}_{i}(z) = \sum_{\gamma=2}^h \alpha_{i} z_{i}\ell \) is zero, for \( i = 2, \ldots, h' \). We do this using a proof method similar to the Schwartz-Zippel Lemma. (See, for example, [21].) 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 \( z_{\gamma} \) (\( \gamma \neq i \)). Therefore there are at most \( (h')^{h-2} < h^{h-1} \) 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 \( \lambda_i \) and ignoring terms with \( \lambda_i = 0 \), we can view \( Z(I'_i) \) as a sum \( Z(I'_i) = \sum c_i \lambda_i^{\ell} \) where the \( \lambda_i \)'s are positive and pairwise distinct and

\[
c_1 = n_1 Z(I'_{i} \mid \sigma(t_0) = 0, \ldots, \sigma(t_{q-1}) = q-1).
\]

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

\[
Z(I' \mid \sigma(t_0) = 0, \ldots, \sigma(t_{q-1}) = q-1) = Z(I).
\]

**Lemma 18.** For every \( \mathcal{F} \subset \mathcal{F}_q \), \#CSP\(^\mathcal{F} \) \( \leq \) \#CSP\(^\mathcal{F} \).

**Proof.** We use Möbius inversion for posets, following the lines of the proof of [2] Theorem 8.\(^4\)

\(^4\) Lovász [15] had previously used Möbius inversion in a similar context.
of \( \theta \). Define \( \mu(\emptyset) = 1 \) and for any \( \theta \neq \emptyset \) 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 \( \theta = \emptyset \). From the definition of \( \mu \), it is also easy to see that the sum is 0 otherwise, since

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

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

Let \( \Omega \) be the set of configurations \( \sigma \) that satisfy all constraints in \( I \) except possibly the disequality constraint. Then \( Z(I) = \sum_{\sigma \in \Omega} w(\sigma) \mathbbm{1}_\sigma \), where \( \mathbbm{1}_\sigma = 1 \) if \( \sigma \) respects the disequality constraint, meaning that \( \vartheta(\sigma) = \emptyset \), and \( \mathbbm{1}_\sigma = 0 \) 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

\[
Z(I) = \sum_{\eta} \mu(\eta) \sum_{\eta \leq \theta} \sum_{\sigma : \eta \mathbin{\bowtie} \sigma} w(\sigma) = \sum_{\eta} \mu(\eta) \sum_{\sigma : \eta \mathbin{\bowtie} \sigma} \sum_{\sigma \in \Omega : \vartheta(\sigma) = \emptyset} w(\sigma).
\]

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