Boolean approximate counting CSPs with weak conservativity, and implications for ferromagnetic two-spin

Miriam Backens* Andrei Bulatov† Leslie Ann Goldberg* Colin McQuillan Stanislav Živný*‡

15 December 2019

Abstract

We analyse the complexity of approximate counting constraint satisfactions problems #CSP($\mathcal{F}$), where $\mathcal{F}$ is a set of nonnegative rational-valued functions of Boolean variables. A complete classification is known in the conservative case, where $\mathcal{F}$ is assumed to contain arbitrary unary functions. We strengthen this result by fixing any permissive strictly increasing unary function and any permissive strictly decreasing unary function, and adding only those to $\mathcal{F}$: this is weak conservativity. The resulting classification is employed to characterise the complexity of a wide range of two-spin problems, fully classifying the ferromagnetic case. In a further weakening of conservativity, we also consider what happens if only the pinning functions are assumed to be in $\mathcal{F}$ (instead of the two permissive unaries). We show that any set of functions for which pinning is not sufficient to recover the two kinds of permissive unaries must either have a very simple range, or must satisfy a certain monotonicity condition. We exhibit a non-trivial example of a set of functions satisfying the monotonicity condition.

1 Introduction

A counting constraint satisfaction problem (counting CSP or #CSP) is parameterised by a finite set $\mathcal{F}$ of functions taking values in some ring. An instance Ω of #CSP($\mathcal{F}$) consists of a set of variables taking values in some domain D, and a set of constraints. Each constraint is a tuple containing a list of (not necessarily distinct) variables, called the scope, and a constraint function, which is an element of $\mathcal{F}$ whose arity is equal to the number of variables in the scope.

Any assignment of values to the variables yields a *weight*, which is the product of the resulting values of the constraint functions. Given the instance Ω, the computational problem is to determine (either exactly or approximately) the sum of the weights of all assignments.

Many counting problems can be expressed in the counting CSP framework. Consider, for example, the problem of counting the number of 2-colourings of a finite graph $G = (V, E)$. A 2-colouring is an assignment from V to $\{0, 1\}$ which has the property that any two vertices connected by an edge are assigned different values. This can be expressed as an instance Ω of #CSP($\{f\}$), where f is the symmetric binary function satisfying $f(0, 1) = f(1, 0) = 1$ and $f(0, 0) = f(1, 1) = 0$. The variables of the instance correspond to the vertices of the graph and the constraints correspond to the edges. An assignment of values in $\{0, 1\}$ to the variables has

*The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) ERC grant agreement no. 334828 (Backens and Goldberg) and Horizon 2020 research and innovation programme (grant agreement no. 714532, Živný). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

†Supported by an NSERC Discovery Grant.

‡Supported by a Royal Society University Research Fellowship.

weight 1 if it corresponds to a valid 2-colouring, and weight 0 otherwise. Thus the sum of the weights of all assignments is exactly the number of 2-colourings of G.

Counting CSPs are closely related to certain problems arising in statistical physics. Each variable can be thought of as an object which can be in one of several states. Adjacent objects interact and the strength of such an interaction is captured by a constraint function. Thus, an assignment associates states with objects. The sum of the weights of all assignments, denoted $Z(\Omega)$, is called the *partition function* of the physical system.

Throughout this paper, we will consider Boolean counting CSPs, which are counting CSPs in which the variables take values from the domain $D = \{0, 1\}$. Constraint functions will be assumed to take nonnegative rational values. The set of all arity-k nonnegative rational-valued functions of Boolean inputs is denoted $\mathcal{B}_k$, and we write $\mathcal{B} = \bigcup_{k\in\mathbb{N}} \mathcal{B}_k$ for the set of nonnegative rational-valued functions of Boolean inputs with arbitrary arity.

The complexity of exactly solving Boolean counting CSPs is fully classified, even when the constraint functions are allowed to take algebraic complex values [5]. This classification takes the form of a dichotomy: if $\mathcal{F}$ is a subset of one of two specific families of functions, the problem is in FP; otherwise it is #P-hard. When the ranges of constraint functions in $\mathcal{F}$ are restricted to be nonnegative rational values (or even algebraic real values), there is only one tractable family, known as *product-type functions* and denoted $\mathcal{N}$ (see Definition 20).

In this paper, we consider the complexity of *approximately* solving counting CSPs. We classify constraint families $\mathcal{F}$ according to whether or not there is a fully polynomial-time randomised approximation scheme (FPRAS) for the problem #CSP($\mathcal{F}$).

If all the constraints in $\mathcal{F}$ are Boolean functions, i.e. functions in $\mathcal{B}$ whose range is $\{0, 1\}$, then the approximation problem #CSP($\mathcal{F}$) is fully classified [12]. We state the precise classification of [12] as Theorem 29 of this paper. Informally, the classification takes the form of a trichotomy, separating problems into ones that are in FP, ones that are equivalent to the problem of counting independent sets in a bipartite graph (denoted #BIS), and ones that do not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$. The equivalence is under approximation-preserving reductions (AP-reductions); in the following, we write $A \leq_{AP} B$ if A can be reduced to B under AP-reductions.

There is also a *conservative* classification for nonnegative efficiently-computable real-valued functions, where "conservative" means that $\mathcal{F}$ is assumed to contain arbitrary unary functions [2]. This classification straightforwardly restricts to the case in which constraint functions take non-negative rational values [6]. The restriction is stated in this paper as Theorem 30. It uses the class of log-supermodular functions, denoted LSM. A k-ary function f is log-supermodular if $f(\mathbf{x} \vee \mathbf{y})f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x})f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \{0, 1\}^k$, where $\vee$ and $\wedge$ are applied bit-wise. The classification shows that #CSP($\mathcal{F}$) is at least as hard as #BIS if $\mathcal{F} \not\subseteq \mathcal{N}$. If $\mathcal{F} \not\subseteq \mathcal{N}$ and $\mathcal{F} \not\subseteq \mathsf{LSM}$ then it is (presumably) even harder – it is as hard as counting the satisfying assignments of a Boolean formula (so there is no FPRAS unless $\mathsf{NP} = \mathsf{RP}$). The paper [2] also implies a #BIS-easiness result for the case when $\mathcal{F} \subseteq \mathsf{LSM}$ and all constraint functions have arity at most three.

In this paper, we give a complexity classification for counting CSPs under a significantly weaker conservativity assumption: instead of adding arbitrary unary functions to $\mathcal{F}$, we fix *one* strictly increasing permissive unary function and *one* strictly decreasing permissive unary function, and we add (only) these to $\mathcal{F}$.

Theorem 1. *Let* up *be a permissive unary strictly increasing function, let* down *be a permissive unary strictly decreasing function, and let* $\mathcal{F} \subseteq \mathcal{B}$*. Then the following properties hold.*

1. *If* $\mathcal{F} \subseteq \mathcal{N}$*, then, for any finite subset* S *of* $\mathcal{F}$*,* #CSP($S \cup \{\mathsf{up}, \mathsf{down}\}$) *is in* FP*.*
2. *Otherwise, if* $\mathcal{F} \subseteq \mathsf{LSM}$*, then*
   (a) *there is a finite subset* S *of* $\mathcal{F}$ *such that* #BIS $\leq_{AP}$ #CSP($S \cup \{\mathsf{up}, \mathsf{down}\}$)*, and*

(b) *for every finite subset S of $\mathcal{F}$ such that all functions $f \in S$ have arity at most 2,* $\#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\}) \leq_{AP} \#\mathsf{BIS}$.

3. *Otherwise, there is a finite subset S of $\mathcal{F}$ such that* $\#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\})$ *does not have an FPRAS unless* $\mathsf{NP} = \mathsf{RP}$.

Our main application of Theorem 1 is a characterisation of the complexity of two-spin problems. A two-spin problem corresponds to the problem $\#\mathrm{CSP}(\{f\})$ where the constraint function f is in $\mathcal{B}_2$. These problems arise in statistical physics. For example, the problem of computing the partition function of the Ising model is a two-spin problem where, for some value β, $f(0,0) = f(1,1) = \beta$ and $f(0,1) = f(1,0) = 1$.

Our application requires the following definitions. We say that a binary function f is monotone if $f(0,0) \leq f(0,1) \leq f(1,1)$ and $f(0,0) \leq f(1,0) \leq f(1,1)$. The Fourier coefficients of f are defined by $\widehat{f}_{xy} = \frac{1}{4}\sum_{p,q\in\{0,1\}}(-1)^{px+qy}f(p,q)$ for all $x, y \in \{0,1\}$.

Let EQ be the binary equality function defined by $\mathrm{EQ}(0,0) = \mathrm{EQ}(1,1) = 1$ and $\mathrm{EQ}(0,1) = \mathrm{EQ}(1,0) = 0$. Let NEQ the binary disequality function defined by $\mathrm{NEQ}(0,0) = \mathrm{NEQ}(1,1) = 0$ and $\mathrm{NEQ}(0,1) = \mathrm{NEQ}(1,0) = 1$. A binary function f is *log-modular* if $f(0,1)f(1,0) = f(0,0)f(1,1)$. A binary function f is called *trivial* if it is log-modular or there is a unary function $g \in \mathcal{B}_1$ such that $f(x,y) = g(x)\mathrm{EQ}(x,y)$ or $f(x,y) = g(x)\mathrm{NEQ}(x,y)$. A binary function f is log-supermodular if and only if it is *ferromagnetic*, which means that $f(0,0)f(1,1) \geq f(0,1)f(1,0)$. Our classification theorem is as follows.

Theorem 2. *Let $f \in \mathcal{B}_2$.*

1. *If f is trivial, then* $\#\mathrm{CSP}(\{f\})$ *is in* FP.

2. *Otherwise, if f is ferromagnetic:*

   (a) *If $\widehat{f}_{01}\widehat{f}_{10} < 0$, then* $\#\mathrm{CSP}(\{f\})$ *is equivalent to* $\#\mathsf{BIS}$ *under AP-reductions.*

   (b) *Otherwise,* $\#\mathrm{CSP}(\{f\})$ *has an FPRAS.*

3. *Otherwise, if both $f(x,y)$ and $f(1-x,1-y)$ are non-monotone, then* $\#\mathrm{CSP}(\{f\})$ *does not have an FPRAS unless* $\mathsf{NP} = \mathsf{RP}$.

The classification in Theorem 2 is not exhaustive: $\#\mathrm{CSP}(\{f\})$ is now fully classified if f is ferromagnetic. For anti-ferromagnetic functions f, the complexity of $\#\mathrm{CSP}(\{f\})$ is still open, except in the doubly non-monotone case (where both $f(x,y)$ and $f(1-x,1-y)$ are non-monotone) and in the symmetric case where $f(x,y) = f(y,x)$. The symmetric case has been resolved with a classification into situations where $\#\mathrm{CSP}(\{f\})$ has a fully polynomial-time approximation scheme (FPTAS) and situations where $\#\mathrm{CSP}(\{f\})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$. More details are given in Section 2.5 however we mention here that the known classification depends on universal uniqueness and there is no simple closed form criterion, cf. Theorems 31 and 32, which are taken from [18].

Theorem 1 relaxes the conservativity assumption in known counting CSP classifications by adding only two permissive unaries to the set $\mathcal{F}$ of constraint functions. Pushing this idea even further, we consider what happens if we allow the pinning functions δ_0 and δ_1 defined by $\delta_0(0) = 1, \delta_0(1) = 0$ and $\delta_1(0) = 0, \delta_1(1) = 1$ instead of the permissive unaries up and down.

Given a strictly decreasing permissive unary function down, δ_0 can be *realised*: this means that the effect of a constraint using the function δ_0 can be (approximately) simulated by some combination of constraints using down. Similarly, δ_1 can be realised using up. This notion of realisation is formalised using the theory of functional clones and (ω, p)-clones in Section 2.3. It implies that allowing only the pinning functions instead of allowing both kinds of permissive unaries is a weaker assumption, though not necessarily a strictly weaker one.

For many sets of functions $\mathcal{F} \subseteq \mathcal{B}$, adding pinning is sufficient to realise two permissive unaries with the desired properties: then the complexity classification of $\#\mathrm{CSP}(\mathcal{F} \cup \{\delta_0, \delta_1\})$

follows from Theorem 1. If pinning does not yield both kinds of permissive unaries, we show that either every function $f \in \mathcal{F}$ has range $\{0, r_f\}$ for some nonnegative rational r_f, or all functions in $\mathcal{F}$ satisfy a certain monotonicity condition. These results form Theorem 58.

In Section 5.2, the goal is to identify a large functional clone that does not contain both a strictly increasing permissive unary function and a strictly decreasing permissive unary function. In other words, we are looking for a set of functions which provably does not allow both a strictly increasing permissive unary function and a strictly decreasing permissive unary function to be realised. The set of monotone functions fits the bill, but in some sense it is a trivial solution since it does not contain both δ_0 and δ_1. Theorem 59 identifies a functional clone that is strictly larger than the set of monotone functions and contains δ_0 and δ_1 but still does not contain both a strictly increasing permissive unary function and a strictly decreasing permissive unary function. This shows that the monotonicity property in the pinning classification can be satisfied in a nontrivial way.

2 Definitions and preliminaries

Throughout this paper, we consider nonnegative rational-valued pseudo-Boolean functions, i.e. functions from $\{0,1\}^k$ to $\mathbb{Q}_{\geq 0}$. We write $\mathcal{B}_k$ for the set of all nonnegative rational-valued pseudo-Boolean functions of arity k, and $\mathcal{B} = \bigcup_{k\in\mathbb{N}} \mathcal{B}_k$.

A function $f \in \mathcal{B}_k$ is *permissive* if $f(\mathbf{x}) > 0$ for all $\mathbf{x} \in \{0,1\}^k$, i.e. all its values are non-zero. The set of permissive unary strictly decreasing functions and the set of permissive unary strictly increasing functions will be of particular interest; we denote them by

$$\mathcal{B}_1^{>} := \{f \in \mathcal{B}_1 : f(0) > f(1) > 0\} \quad \text{and}$$
$$\mathcal{B}_1^{<} := \{f \in \mathcal{B}_1 : 0 < f(0) < f(1)\}.$$

These two sets differ from the sets $\mathcal{B}_1^{\mathsf{up},p}$ and $\mathcal{B}_1^{\mathsf{down},p}$ defined in [2] as the latter do not require strictness or permissiveness, and they allow polynomial-time computable real values rather than just rational values. We will also sometimes require normalised unary functions, thus we define $\mathcal{B}_1^{>,\mathrm{n}} := \{f \in \mathcal{B}_1^{>} : f(0) = 1\}$ and $\mathcal{B}_1^{<,\mathrm{n}} := \{f \in \mathcal{B}_1^{<} : f(1) = 1\}$.

The *relation underlying a function* $f \in \mathcal{B}_k$ (also called the "support of f") is defined as

$$R_f = \{(x_1, \ldots, x_k) : f(x_1, \ldots, x_k) \neq 0\}.$$

A relation is *affine* if it contains exactly the tuples specified by a set of linear equations over GF(2). If the underlying relation of f is affine, we say that f *has affine support*. A function is *pure affine* if has affine support and its range is $\{0, r\}$ for some $r > 0$ [11]. We extend this definition to say that any function is *pure* if its range is $\{0, r\}$ for some $r > 0$, regardless of its support.

A function $f \in \mathcal{B}_k$ is *log-supermodular* (or *lsm*) if $f(\mathbf{x} \vee \mathbf{y})f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x})f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \{0,1\}^k$, where $\vee$ and $\wedge$ are applied bit-wise. The class of all lsm functions of any arity is denoted LSM. A k-ary function f is *log-modular* if $f(\mathbf{x} \vee \mathbf{y})f(\mathbf{x} \wedge \mathbf{y}) = f(\mathbf{x})f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \{0,1\}^k$. It is straightforward to check that all unary functions are both lsm and log-modular.

For any positive integer n, we write $[n] := \{1, \ldots, n\}$.

Let δ_0 and δ_1 be the unary functions satisfying $\delta_0(0) = 1, \delta_0(1) = 0$ and $\delta_1(0) = 0, \delta_1(1) = 1$; these are often called the *pinning functions*. If f is a function in $\mathcal{B}_k$ with $k \geq 2$, then a *2-pinning* of f is a binary function g that arises from f by pinning all but two of the variables to a fixed value. Formally, g is of the form

$$g(x_p, x_q) = \sum_{(x_{i_1},\ldots,x_{i_{k-2}})\in\{0,1\}^{k-2}} f(x_1, \ldots, x_k) \prod_{i\in[k]\setminus\{p,q\}} \delta_{a_i}(x_i),$$

where p and q are distinct indices in $[k]$, $\{i_1, \ldots, i_{k-2}\} = [k] \setminus \{p, q\}$, and for each $i \in [k] \setminus \{p, q\}$, a_i is a value in $\{0, 1\}$.

A function $f \in \mathcal{B}_k$ is *monotone* if for any $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ with $\mathbf{a} \leq \mathbf{b}$ we have $f(\mathbf{a}) \leq f(\mathbf{b})$.

A function is *monotone on its support* if for any $\mathbf{a}, \mathbf{b} \in R_f$ with $\mathbf{a} \leq \mathbf{b}$ we have $f(\mathbf{a}) \leq f(\mathbf{b})$. All monotone functions are also monotone on their support, but the latter set is bigger: a function $f \in \mathcal{B}_k$ that is monotone on its support may have inputs $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ such that $\mathbf{a} \leq \mathbf{b}$ and $f(\mathbf{a}) > f(\mathbf{b})$, as long as $f(\mathbf{b}) = 0$. For example, δ_0 is trivially monotone on its support, but it is not monotone.

Let $\bar{x} = 1 - x$ for $x \in \{0, 1\}$. The *bit-flip* of a function $f \in \mathcal{B}_k$ is the function $\bar{f}(x_1, \ldots, x_k) = f(\bar{x}_1, \ldots, \bar{x}_k)$. The bit-flip of a set $\mathcal{F} \subseteq \mathcal{B}$ is $\bar{\mathcal{F}} := \{\bar{f} : f \in \mathcal{F}\}$.

2.1 Binary functions

Much of this paper is concerned with binary functions, it is thus useful to introduce specific notation and results. A binary function is said to be *ferromagnetic* if it is log-supermodular and it is *anti-ferromagnetic* otherwise. We often write a binary function as a 2×2 matrix:

$$f(x, y) = \begin{pmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{pmatrix}.$$

Observation 3. A binary function f is log-supermodular (or, equivalently, it is ferromagnetic) if $f(0,0)f(1,1) \geq f(0,1)f(1,0)$.

The binary equality function EQ is defined by $\mathrm{EQ}(0,0) = \mathrm{EQ}(1,1) = 1$ and $\mathrm{EQ}(0,1) = \mathrm{EQ}(1,0) = 0$ and the binary disequality function NEQ is defined by $\mathrm{NEQ}(0,0) = \mathrm{NEQ}(1,1) = 0$ and $\mathrm{NEQ}(0,1) = \mathrm{NEQ}(1,0) = 1$. A binary function f is called *trivial* if it is log-modular or there is a unary function $g \in \mathcal{B}_1$ such that $f(x, y) = g(x)\mathrm{EQ}(x, y)$ or $f(x, y) = g(x)\mathrm{NEQ}(x, y)$. It is easy to see that a binary function f is log-modular if and only if $f(x, y) = g(x)h(y)$ for some unary functions g and h. We will use the following observation.

Observation 4. A permissive binary function is trivial if and only if it is log-modular.

A binary function f is *symmetric* if $f(0,1) = f(1,0)$ and it is an *Ising function* if it depends only on the parity of its input, i.e. $f(0,1) = f(1,0)$ and $f(0,0) = f(1,1)$.

It is often easier to work with symmetric binary functions than with general ones. The following lemma gives some properties of different methods for symmetrising a given function.

Lemma 5. *Let f be a nontrivial binary function.*

1. *If f is non-lsm, then $f'(x, y) = f(x, y)f(y, x)$ is nontrivial, symmetric, and non-lsm.*
2. *If f is lsm, then $f''(x, y) = \sum_{z \in \{0,1\}} f(x, z)f(y, z)$ is nontrivial, symmetric, and lsm.*
3. *If f is lsm and Ising, then $f'''(x, y) = \sum_{z \in \{0,1\}} f(x, z)f(y, z)\mathsf{up}(z)$ is nontrivial, symmetric, lsm, and not Ising, where up is any strictly increasing permissive unary function.*

Proof. Throughout, we write

$$f(x, y) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

For Property 1, suppose f is nontrivial and non-lsm, i.e. $ad - bc < 0$ and a, d are not both zero. Let $f'(x, y) = f(x, y)f(y, x)$, i.e. $f'(0,0) = a^2$, $f'(0,1) = bc = f'(1,0)$, and $f'(1,1) = d^2$. It is straightforward to see that f' is symmetric. As $b, c > 0$ and a, d are not both zero, f' does not have the form $g(x)\mathrm{EQ}(x, y)$ or $g(x)\mathrm{NEQ}(x, y)$ for any unary function g. Furthermore, $ad < bc$ and $a, b, c, d \geq 0$ implies that $a^2d^2 < b^2c^2 = (bc)^2$, so $f'(0,0)f'(1,1) < f'(0,1)f'(1,0)$.

This shows that f' is not log-modular and therefore nontrivial. Additionally, it also shows that f' is non-lsm, concluding the proof of the property.

For Property 2, suppose f is nontrivial and lsm, i.e. $ad - bc > 0$ and b, c are not both zero. Let

$$f''(x,y) = \sum_{z\in\{0,1\}} f(x,z)f(y,z) = \begin{pmatrix} a^2+b^2 & ac+bd \\ ac+bd & c^2+d^2 \end{pmatrix}.$$

This can easily be seen to be symmetric. Since at most one of a, b, c, d is zero, f'' is permissive. Finally, $f''(0,0)f''(1,1) - f''(0,1)f''(1,0) = (ad-bc)^2 > 0$, so (using Observation 4) f'' is nontrivial and lsm.

For Property 3, suppose f is nontrivial, lsm and Ising, i.e. $ad - bc > 0$, $a = d$, and $b = c$, with $a, b, c, d > 0$. We can thus write

$$f(x,y) = \begin{pmatrix} a & b \\ b & a \end{pmatrix};$$

the property of f being both nontrivial and lsm becomes $a^2 - b^2 > 0$. Now, let $u_0 := \mathsf{up}(0)$ and $u_1 := \mathsf{up}(1)$; these values satisfy $0 < u_0 < u_1$. Then

$$f'''(x,y) = \sum_{z\in\{0,1\}} f(x,z)f(y,z)\mathsf{up}(z) = \begin{pmatrix} a^2u_0 + b^2u_1 & ab(u_0+u_1) \\ ab(u_0+u_1) & a^2u_1 + b^2u_0 \end{pmatrix}.$$

Since a, b, u_0, u_1 are all positive, f''' is permissive. It is also clearly symmetric. Furthermore,

$$f'''(0,0)f'''(1,1) - f'''(0,1)f'''(1,0) = (a^2-b^2)^2u_0u_1 > 0.$$

Thus, using Observation 4, f''' is nontrivial and lsm. The function f''' is Ising if $a^2u_0 + b^2u_1 = a^2u_1 + b^2u_0$ or, equivalently, if $(a^2 - b^2)(u_0 - u_1) = 0$. But f being nontrivial implies that $a^2 - b^2 \neq 0$ and up being strictly increasing implies that $u_0 \neq u_1$, hence this equality is never satisfied. That means f''' cannot be Ising, so f''' has all the desired properties. □

2.2 Fourier transforms

The Fourier transform of a k-ary function $f \in \mathcal{B}_k$ is given by

$$\widehat{f}(x_1,\ldots,x_k) = \frac{1}{2^k} \sum_{p_1,\ldots,p_k\in\{0,1\}} (-1)^{p_1x_1+\ldots+p_kx_k} f(p_1,\ldots,p_k).$$

The values of $\widehat{f}$ are called the Fourier coefficients of f. The following fact is well-known. See, e.g., [9, Equation 2.1].

Observation 6. For any $f \in \mathcal{B}_k$,

$$f(x_1,\ldots,x_k) = \sum_{p_1,\ldots,p_k\in\{0,1\}} (-1)^{p_1x_1+\ldots+p_kx_k} \widehat{f}(p_1,\ldots,p_k).$$

We denote by $\mathcal{P}$ the set of all nonnegative functions whose Fourier coefficients are also nonnegative:

$$\mathcal{P} = \left\{ f \in \mathcal{B} : \widehat{f}(\mathbf{x}) \geq 0 \text{ for all } \mathbf{x} \in \{0,1\}^{\mathrm{arity}(f)} \right\},$$

where $\mathrm{arity}(f)$ denotes the arity of f. If f is binary, with

$$f(x,y) = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

its Fourier coefficients are sometimes written as

$$\widehat{f}_{00} := \widehat{f}(0,0) = \frac{1}{4}(a+b+c+d)$$
$$\widehat{f}_{01} := \widehat{f}(0,1) = \frac{1}{4}(a-b+c-d)$$
$$\widehat{f}_{10} := \widehat{f}(1,0) = \frac{1}{4}(a+b-c-d)$$
$$\widehat{f}_{11} := \widehat{f}(1,1) = \frac{1}{4}(a-b-c+d).$$

Observation 7. Let $f'(x,y) = \bar{f}(x,y) = f(1-x,1-y)$, where we are renaming the function temporarily in order to avoid clumsy notation. Then $\widehat{f'}_{01} = -\widehat{f}_{01}$ and $\widehat{f'}_{10} = -\widehat{f}_{10}$, while $\widehat{f'}_{00} = \widehat{f}_{00}$ and $\widehat{f'}_{11} = \widehat{f}_{11}$. In other words, a bit-flip changes the signs of $\widehat{f}_{01}$ and $\widehat{f}_{10}$ while leaving $\widehat{f}_{00}$ and $\widehat{f}_{11}$ unaffected.

Observation 8. Let $f''(x,y) = f(y,x)$. Then $\widehat{f''}_{01} = \widehat{f}_{10}$ and $\widehat{f''}_{10} = \widehat{f}_{01}$, while again $\widehat{f''}_{00} = \widehat{f}_{00}$ and $\widehat{f''}_{11} = \widehat{f}_{11}$. Hence, swapping the variables swaps the values of $\widehat{f}_{01}$ and $\widehat{f}_{10}$.

These properties will be useful when considering two-spin problems in Section 4.

Lemma 9. *Let $f \in \mathcal{P}$ with arity k. Then $f(x_1,\ldots,x_k) \le f(0,\ldots,0)$ for any $x_1,\ldots,x_k \in \{0,1\}$.*

Proof. In the following, we write $\mathbf{p}\cdot\mathbf{x} = p_1x_1 \oplus \ldots \oplus p_kx_k$, where $\mathbf{x} = (x_1,\ldots,x_k), \mathbf{p} = (p_1,\ldots,p_k) \in \{0,1\}^k$. By Observation 6, for any $f \in \mathcal{B}_k$,

$$f(\mathbf{x}) = \sum_{\mathbf{p}\in\{0,1\}^k} (-1)^{\mathbf{p}\cdot\mathbf{x}} \widehat{f}(\mathbf{p}). \tag{1}$$

For any $\mathbf{x} \in \{0,1\}^k$, let $S_{f;\mathbf{x}}$ be the set of all k-bit strings $\mathbf{p}$ such that $\widehat{f}(\mathbf{p})$ appears with coefficient -1 in (1):

$$S_{f;\mathbf{x}} = \left\{\mathbf{p} \in \{0,1\}^k : \mathbf{p}\cdot\mathbf{x} = 1\right\}.$$

Note that if $\mathbf{x}$ is the all-zeroes string, then $\mathbf{p}\cdot\mathbf{x} = 0$ for all $\mathbf{p} \in \{0,1\}$. This implies that $S_{f;0,\ldots,0} = \emptyset$ and $f(0,\ldots,0) = \sum_{\mathbf{p}\in\{0,1\}^k} \widehat{f}(\mathbf{p})$. Thus, (1) can be rewritten as

$$f(\mathbf{x}) = \sum_{\mathbf{p}\in\{0,1\}^k} \widehat{f}(\mathbf{p}) - 2\left(\sum_{\mathbf{p}\in S_{f;\mathbf{x}}} \widehat{f}(\mathbf{p})\right) = f(0,\ldots,0) - 2\left(\sum_{\mathbf{p}\in S_{f;\mathbf{x}}} \widehat{f}(\mathbf{p})\right) \le f(0,\ldots,0),$$

where the inequality holds because $f \in \mathcal{P}$ implies that all terms in the sum are nonnegative. □

2.3 Relational clones, functional clones, and ω-clones

Functional clones, ω-clones and (ω,p)-clones are sets of functions that are closed under certain operations. These operations are listed in Lemma 14 for functional clones; ω-clones and (ω,p)-clones additionally have different ways of taking limits. Their definitions build on the well-established theory of relational clones. To simplify our notation, we take our definitions from [1, 2] though relational clones have a longer history [20]. In the following, let Γ be a set of relations and let $V = \{v_1,\ldots,v_n\}$ be a set of variables.

Definition 10 ([2])**.** A *primitive positive formula* (pp-formula) over Γ in variables V is a formula of the form

$$\exists v_{n+1}\ldots v_{n+m} \bigwedge_i \varphi_i,$$

where each atomic formula φ_i is either a relation R from Γ or the equality relation, applied to some of the variables in $V' = \{v_1,\ldots,v_{n+m}\}$.

Definition 11 ([2])**.** The *relational clone* (or co-clone) $\langle\Gamma\rangle_R$ is the set of all relations expressible as pp-formulas over Γ.

The definition of a functional clone is similar, we again follow the explanation in [2]. Note that functional clones, ω-clones and (ω, p)-clones were originally defined for nonnegative real-valued functions. The definitions can be restricted to nonnegative rational-valued functions as explained just after Definition 15.

Let $\mathcal{F} \subseteq \mathcal{B}$ be a set of functions and $V = \{v_1, \ldots, v_n\}$ a set of variables. In the context of functions rather than relations, an atomic formula $\varphi = g(v_{i_1}, \ldots, v_{i_k})$ consists of a function $g \in \mathcal{F}$ and a scope $(v_{i_1}, \ldots, v_{i_k}) \in V^k$, where $k = \text{arity}(g)$. The scope may contain repeated variables. Given an assignment $\mathbf{x} : V \to \{0,1\}$, the atomic formula φ specifies a function $f_\varphi : \{0,1\}^n \to \mathbb{Q}_{\geq 0}$ given by

$$f_\varphi(\mathbf{x}) = g(x_{i_1}, \ldots, x_{i_k}),$$

where $x_j = \mathbf{x}(v_j)$ for all $j \in [n]$.

Definition 12. A *primitive product summation formula* (pps-formula) in variables V over $\mathcal{F}$ has the form

$$\psi = \sum_{v_{n+1},\ldots,v_{n+m}} \prod_{j=1}^{s} \varphi_j,$$

where φ_j are all atomic formulas over $\mathcal{F}$ in the variables $V' = \{v_1, \ldots, v_{n+m}\}$. The variables in V are called *free variables*, those in $V' \setminus V$ are called *bound variables*.

The pps-formula ψ represents a function $f_\psi : \{0,1\}^n \to \mathbb{Q}_{\geq 0}$ given by

$$f_\psi(\mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^m} \prod_{j=1}^{s} f_{\varphi_j}(\mathbf{x}, \mathbf{y}),$$

where $\mathbf{x}$ is an assignment $V \to \{0,1\}$ and $\mathbf{y}$ is an assignment $V' \setminus V \to \{0,1\}$. If f is represented by some pps-formula ψ over $\mathcal{F}$, it is said to be *pps-definable* over $\mathcal{F}$.

Definition 13 ([2])**.** The *functional clone* generated by $\mathcal{F}$ is the set of all functions in $\mathcal{B}$ that can be represented by a pps-formula over $\mathcal{F} \cup \{\text{EQ}\}$. It is denoted by $\langle\mathcal{F}\rangle$.

There is a another perspective on functional clones [1]. In the following, let $f \in \mathcal{B}_k$. We say a $(k+1)$-ary function h arises from f by *introduction of a fictitious argument* if $h(x_1, \ldots, x_{k+1}) = f(x_1, \ldots, x_k)$ for all $x_1, \ldots, x_k \in \{0,1\}$. Let $g \in \mathcal{B}_k$, then the *product* of f and g is the function h satisfying $h(x_1, \ldots, x_k) = f(x_1, \ldots, x_k)g(x_1, \ldots, x_k)$. The function h resulting from f by a *permutation of the arguments* $\pi : [k] \to [k]$ is $h(x_1, \ldots, x_k) = f(x_{\pi(1)}, \ldots, x_{\pi(k)})$. Furthermore, a $(k-1)$-ary function h arises from f by *summation* if $h(x_1, \ldots, x_{k-1}) = \sum_{x_k \in \{0,1\}} f(x_1, \ldots, x_k)$.

Lemma 14 ([1, Section 1.1])**.** *For any $\mathcal{F} \subseteq \mathcal{B}$, $\langle\mathcal{F}\rangle$ is the closure of $\mathcal{F} \cup \{\text{EQ}\}$ under introduction of fictitious arguments, products, permutation of arguments, and summation.*

We adopt the shorthand notation from [2], so if $\mathcal{F}_1, \ldots, \mathcal{F}_j$ are sets of functions and $g_1, \ldots, g_k$ are functions, then $\langle\mathcal{F}_1, \ldots, \mathcal{F}_j, g_1, \ldots, g_k\rangle := \langle\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_j \cup \{g_1, \ldots, g_k\}\rangle$.

Note that if $\mathcal{F} \subseteq \mathcal{B}$ and $g \in \langle\mathcal{F}\rangle$, then $\langle\mathcal{F}, g\rangle = \langle\mathcal{F}\rangle$ [2, Lemma 2.1].

Definition 15 ([2])**.** A function $f \in \mathcal{B}_k$ is *pps$_\omega$-definable* over $\mathcal{F} \subseteq \mathcal{B}$ if there exists a finite subset S_f of $\mathcal{F}$ such that, for every $\varepsilon > 0$, there exists a k-ary function f', pps-definable over S_f, such that

$$\left\|f' - f\right\|_\infty = \max_{\mathbf{x} \in \{0,1\}^k} \left|f'(\mathbf{x}) - f(\mathbf{x})\right| < \varepsilon.$$

Note that to be pps_ω-definable over some subset of $\mathcal{B}$, a function must itself be in $\mathcal{B}$, i.e. it must take rational values. This avoids complications resulting from the limits of some rational sequences being irrational.

Definition 16. Let $\mathcal{F} \subseteq \mathcal{B}$. The set of all functions that are pps_ω-definable over the set $\mathcal{F} \cup \{\text{EQ}\}$ is called the *ω-clone* generated by $\mathcal{F}$; it is denoted $\langle\mathcal{F}\rangle_\omega$.

In [2], ω-clones were originally called "pps_ω-definable functional clones"; we instead use the shorter terminology from [1].

Definition 17 ([2])**.** A function $f \in \mathcal{B}$ is *efficiently pps_ω-definable* over $\mathcal{F}$ if there is a finite subset S_f of $\mathcal{F}$ and a Turing machine $\mathcal{M}_{f,S_f}$ with the following property: on input $\varepsilon > 0$, $\mathcal{M}_{f,S_f}$ computes a pps-formula ψ over S_f such that f_ψ has the same arity as f and $\|f_\psi - f\|_\infty < \varepsilon$. The running time of $\mathcal{M}_{f,S_f}$ is at most a polynomial in $\log \varepsilon^{-1}$.

Definition 18. Let $\mathcal{F} \subseteq \mathcal{B}$. The set of all functions that are efficiently pps_ω-definable over the set $\mathcal{F} \cup \{\text{EQ}\}$ is called the *(ω, p)-clone* generated by $\mathcal{F}$; it is denoted $\langle\mathcal{F}\rangle_{\omega,p}$.

The (ω, p)-clones were originally called "efficiently pps_ω-definable functional clones"; we shorten the terminology here to bring it in line with the term "ω-clone". We use the same shorthand as for functional clones, so $\langle\mathcal{F}_1, \ldots, \mathcal{F}_j, g_1, \ldots, g_k\rangle_\omega := \langle\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_j \cup \{g_1, \ldots, g_k\}\rangle_\omega$ and $\langle\mathcal{F}_1, \ldots, \mathcal{F}_j, g_1, \ldots, g_k\rangle_{\omega,p} := \langle\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_j \cup \{g_1, \ldots, g_k\}\rangle_{\omega,p}$

Note that if $\mathcal{F} \subseteq \mathcal{B}$ and $g \in \langle\mathcal{F}\rangle_\omega$, then $\langle\mathcal{F}, g\rangle_\omega = \langle\mathcal{F}\rangle_\omega$ [2, Lemma 2.2]. Similarly, if $\mathcal{F} \subseteq \mathcal{B}$ and $g \in \langle\mathcal{F}\rangle_{\omega,p}$, then $\langle\mathcal{F}, g\rangle_{\omega,p} = \langle\mathcal{F}\rangle_{\omega,p}$ [2, Lemma 2.4].

Observation 19. If $\mathsf{up} \in \mathcal{B}_1^{<,\text{n}}$, then $\delta_1 \in \langle\mathsf{up}\rangle_{\omega,p}$ and if $\mathsf{down} \in \mathcal{B}_1^{>,\text{n}}$, then $\delta_0 \in \langle\mathsf{down}\rangle_{\omega,p}$. (Indeed, let $\mathsf{up} \in \mathcal{B}_1^{<,\text{n}}$, that is $\mathsf{up}(0) = a, \mathsf{up}(1) = 1$ for some $a < 1$. Then $\mathsf{up}^k(0) = a^k$ and $\mathsf{up}^k(1) = 1$. Therefore $\lim_{k\to\infty} \mathsf{up}^k(x) = \delta_1$, witnessing that $\delta_1 \in \langle\mathsf{up}\rangle_{\omega,p}$.)

Recall the definition of NEQ from Section 2.1 and the definition of $\mathcal{B}_1$ from Section 2. We next define the functional clone $\mathcal{N}$, which arises in our classification theorems. Elements of $\mathcal{N}$ are called *product type functions*.

Definition 20. $\mathcal{N} := \langle\text{NEQ}, \mathcal{B}_1\rangle$.

Two collections of functions defined earlier in this paper are in fact ω-clones: the log-supermodular functions and the functions with nonnegative Fourier transform.

Lemma 21 ([2, Lemma 4.2])**.** *If $\mathcal{F} \subseteq \mathsf{LSM}$, then $\langle\mathcal{F}\rangle_\omega \subseteq \mathsf{LSM}$.*

Lemma 22 ([1, Theorem 28])**.** $\langle\mathcal{P}\rangle_\omega = \mathcal{P}$.

2.4 Approximate counting and counting CSPs

A counting constraint satisfaction problem $\#\text{CSP}(\mathcal{F})$ is parameterised by a finite set of functions $\mathcal{F}$ over some domain D, which we take to be $D = \{0, 1\}$. An instance Ω of $\#\text{CSP}(\mathcal{F})$ is specified by a finite set V of variables and a finite set C of constraints. Each constraint $c = (\mathbf{v}_c, f_c)$ consists of a tuple $\mathbf{v}_c$ of k variables for some $k \in \mathbb{N}$, which may involve repeated variables, and a function $f_c \in \mathcal{F}$ of arity k. Any assignment $\sigma : V \to \{0, 1\}$ of values to the variables is thus associated with a weight $w_\sigma = \prod_{c\in C} f_c(\sigma(\mathbf{v}_c))$, where $\sigma(\mathbf{v}_c)$ is computed component-wise. The *partition function* of instance Ω is the sum of the weights of all the different assignments:

$$Z(\Omega) = \sum_{\sigma:V\to\{0,1\}} w_\sigma = \sum_{\sigma:V\to\{0,1\}} \prod_{c\in C} f_c(\sigma(\mathbf{v}_c)). \tag{2}$$

We will be interested in the problem of approximating the partition function associated with a counting CSP. An approximation scheme for the problem $\#\mathrm{CSP}(\mathcal{F})$ is an algorithm that takes as input an error parameter ε and an instance Ω, and outputs a value $\tilde{Z}$, which satisfies

$$e^{-\varepsilon} Z(\Omega) \leq \tilde{Z} \leq e^{\varepsilon} Z(\Omega).$$

This algorithm is an *FPTAS* or *fully polynomial-time approximation scheme* if its running time is polynomial in the size of the instance and in ε^{-1}.

The problem $\#\mathrm{CSP}(\mathcal{F})$ has a *randomised approximation scheme* if there exists a randomised algorithm which takes as input an instance Ω and an error parameter $\varepsilon > 0$, and which outputs a value $\tilde{Z}$, satisfying

$$\Pr\left[e^{-\varepsilon} Z(\Omega) \leq \tilde{Z} \leq e^{\varepsilon} Z(\Omega)\right] \geq \frac{3}{4}$$

for all inputs Ω and ε. This algorithm is an *FPRAS* or *fully polynomial randomised approximation scheme* if its running time is polynomial in the size of the instance and in ε^{-1} [10].

An *approximation-preserving reduction* (or *AP-reduction*) from a counting problem C to a counting problem C' is an algorithm turning an FPRAS for C' into an FPRAS for C. A rigorous definition of this notion may be found in [10]. If there exists such a reduction, we write $C \leq_{AP} C'$ and say C is *AP-reducible* to C'. If AP-reductions exist in both directions, C and C' are said to be *AP-interreducible*, denoted $C =_{AP} C'$.

Observation 23. Let $\mathcal{F}$ be a finite subset of $\mathcal{B}$, then $\#\mathrm{CSP}(\mathcal{F}) =_{AP} \#\mathrm{CSP}(\bar{\mathcal{F}})$.

This holds because the value of the partition function (2) is unchanged if each constraint function in that equation is replaced by its bit-flip. We use the same shorthand for counting CSPs that we do for clones, so $\#\mathrm{CSP}(\mathcal{F}_1, \ldots, \mathcal{F}_j, g_1, \ldots, g_k) := \#\mathrm{CSP}(\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_j \cup \{g_1, \ldots, g_k\})$.

Lemma 24. *For all $f \in \mathcal{B}_1^{<}$ there is an $f' \in \mathcal{B}_1^{<,\mathrm{n}}$ such that, for all finite $\mathcal{F} \subseteq \mathcal{B}$, the problem $\#\mathrm{CSP}(\mathcal{F}, f) =_{AP} \#\mathrm{CSP}(\mathcal{F}, f')$. Similarly, for all $f \in \mathcal{B}_1^{>}$ there is an $f' \in \mathcal{B}_1^{>,\mathrm{n}}$ such that, for all finite $\mathcal{F} \subseteq \mathcal{B}$, $\#\mathrm{CSP}(\mathcal{F}, f) =_{AP} \#\mathrm{CSP}(\mathcal{F}, f')$.*

Proof. To see the first statement, note that $f \in \mathcal{B}_1^{<}$ implies f is permissive. The function $f'(x) = f(x)/f(1)$ is therefore well-defined, and an element of $\mathcal{B}_1^{<,\mathrm{n}}$. Consider an instance Ω of $\#\mathrm{CSP}(\mathcal{F}, f)$ with n constraints, m of which use the function f. Construct an instance Ω' of $\#\mathrm{CSP}(\mathcal{F}, f')$ by replacing each of those m constraints with a constraint using f'. Then $Z(\Omega) = (f(1))^m Z(\Omega')$, so since $m = \mathcal{O}(n)$, any FPRAS for $\#\mathrm{CSP}(\mathcal{F}, f')$ can be turned into an FPRAS for $\#\mathrm{CSP}(\mathcal{F}, f)$. Similarly, any FPRAS for $\#\mathrm{CSP}(\mathcal{F}, f)$ can be turned into an FPRAS for $\#\mathrm{CSP}(\mathcal{F}, f')$, so $\#\mathrm{CSP}(\mathcal{F}, f) =_{AP} \#\mathrm{CSP}(\mathcal{F}, f')$.

An analogous argument holds if $f \in \mathcal{B}_1^{>}$. □

The complexity of counting CSPs is closely linked to the theory of (ω, p)-clones, as can be seen from the following AP-reduction.

Lemma 25 ([2, Lemma 10.1])**.** *Suppose that $\mathcal{F}$ is a finite subset of $\mathcal{B}$. If $g \in \langle \mathcal{F} \rangle_{\omega,p}$, then*

$$\#\mathrm{CSP}(\mathcal{F}, g) \leq_{AP} \#\mathrm{CSP}(\mathcal{F}).$$

We will frequently use the following subset of CSPs called *Holant problems* [3].

Definition 26. A counting CSP in which each variable appears exactly twice is called a *holant problem*. Thus, an instance of $\mathsf{Holant}(\mathcal{F})$ is an instance $\Omega = (V, C)$ of $\#\mathrm{CSP}(\mathcal{F})$ such that each variable in V appears exactly twice in constraints in C. The objective of the problem $\mathsf{Holant}(\mathcal{F})$ is to compute $Z(\Omega)$.

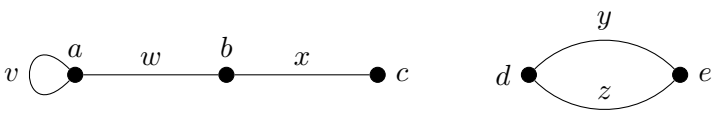

Figure 1: The multigraph representation of a holant instance $\Omega = (V, C)$ with variables $V = \{v, w, x, y, z\}$ and constraints $C = \{a, b, c, d, e\}$ with $a = ((v, v, w), f)$, $b = ((w, x), g)$, $c = ((x), h)$, $d = ((y, z), g)$, and $e = ((y, z), k)$, constructed as in Observation 28. Note that $V' = \{a, b, c, d, e\}$, $S = \{\{a, a\}\}$, $T = \{\{a, b\}, \{b, c\}, \{d, e\}, \{d, e\}\}$, and $E' = S \cup T$. The set of constraint functions used is $\mathcal{F} = \{f, g, h, k\}$.

There is an equivalent view of holant problems that will also be useful. Let EQ_3 be the ternary equality function, that is, $\mathrm{EQ}_3(x, y, z) = 1$ if and only if $x = y = z$ and $\mathrm{EQ}_3(x, y, z) = 0$ otherwise. It turns out (see [4, Proposition 1]) that the problem $\#\mathrm{CSP}(\mathcal{F})$ is equivalent to $\mathsf{Holant}(\mathcal{F}, \mathrm{EQ}_3)$ in following sense. For each instance Ω of $\#\mathrm{CSP}(\mathcal{F})$ there is an instance Ω' of $\mathsf{Holant}(\mathcal{F}, \mathrm{EQ}_3)$ such that $Z(\Omega) = Z(\Omega')$. There is an easy polynomial-time algorithm that turns Ω into Ω' and vice-versa.

Holant problems are an object of study in their own right and exhibit some properties not found in counting CSPs. For example, the holant framework allows reductions by *holographic transformations*, which transform all the constraint functions but nevertheless keep the partition function invariant. These transformations are easiest to define by exploiting the following bijection between functions in $\mathcal{B}_k$ and vectors in $(\mathbb{Q}_{\geq 0})^{2^k}$:

$$f \in \mathcal{B}_k \quad \leftrightarrow \quad \mathbf{f} = (f(0, \ldots, 0), f(0, \ldots, 0, 1), \ldots, f(1, \ldots, 1)).$$

Let M be an invertible 2×2 matrix with nonnegative rational values. The holographic transformation of f by M, denoted $M \circ f$, is the function corresponding to $(M \otimes \ldots \otimes M)\mathbf{f}$, where the tensor product contains k copies of M. For a set $\mathcal{F} \subseteq \mathcal{B}$, we define $M \circ \mathcal{F} := \{M \circ f : f \in \mathcal{F}\}$. We can now state a corollary of Valiant's Holant Theorem [24, 4], adapted to our setting. The result follows from [4, Proposition 5] because constant factors can be absorbed into AP-reductions.

Theorem 27 (Corollary of Valiant's Holant Theorem)**.** *Let $\mathcal{F}$ be a finite subset of $\mathcal{B}$ and let M be a 2×2 matrix such that $MM^T = cI$ where I is the identity matrix and c is a rational number. Then $\mathsf{Holant}(\mathcal{F}) \equiv_{AP} \mathsf{Holant}(M \circ \mathcal{F})$.*

A holant instance (V, C) can be represented as a multigraph with vertex set C. The edges of the multigraph correspond to the variables in V. Informally, there is a self-loop for each variable that appears twice in the same constraint. If the two appearances of a variable v are in different constraints, say c and c', then there is an edge of the multigraph from c to c' corresponding to v. This notion is formalised in the following observation and illustrated by example in Figure 1.

Observation 28. Any holant instance Ω with variables V and constraints C corresponds to a multigraph $G' = (V', E')$ where $V' = C$ and E' is defined as follows. Let S be the multiset whose elements are of the form $\{c, c\}$ where $c \in C$. The multiplicity of $\{c, c\}$ in S is equal to the number of variables that are repeated in the scope $\mathbf{v}_c$. Let T be the multiset whose elements are of the form $\{c, c'\}$ where $c \in C$ and $c' \in C$ but $c \neq c'$. The multiplicity of $\{c, c'\}$ in T is equal to the number of variables that are in the scope of $\mathbf{v}_c$ and the scope of $\mathbf{v}_{c'}$. E' is defined to be the multiset $S \cup T$.

For asymmetric constraints it is necessary to specify an enumeration of the edges incident on each vertex in order to be able to recover the original holant instance from the graph.

Many counting problems defined on graphs have natural expressions in the holant framework which are easily understood using this perspective. For example, the problem of counting the

perfect matchings of a graph corresponds to $\mathsf{Holant}(\{\text{Exact-One}_k : k \in \mathbb{N}_{>0}\})$ with

$$\text{Exact-One}_k(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } \sum_{i=1}^{k} x_i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

An edge is in the perfect matching if the corresponding variable is assigned 1. The constraint functions ensure that any assignment with non-zero weight includes exactly one edge incident on any vertex, and thus corresponds to a perfect matching of the graph.

The problem of counting the number of satisfying assignments of a Boolean formula in conjunctive normal form is denoted #SAT. This problem, and any problem that #SAT AP-reduces to, cannot have an FPRAS unless NP = RP [10].

The problem of counting the number of independent sets in a bipartite graph is denoted #BIS. While #BIS $\leq_{AP}$ #SAT, no AP-reduction from #SAT to #BIS is known. At the same time, #BIS is not known to have an FPRAS [10].

2.5 Existing results

There is a trichotomy for the complexity of counting CSPs where all constraints are Boolean relations. This can straightforwardly be transformed into a result about pure pseudo-Boolean functions as multiplying a constraint function by a nonnegative rational constant does not affect the complexity of the counting problem with respect to AP-reductions.

IM_2 is the set of relations that can be expressed as conjunctions of the unary relations $\{(0)\}$ and $\{(1)\}$ (which correspond to the functions δ_0 and δ_1) as well as the binary relation $\text{IMP} = \{(0,0),(0,1),(1,1)\}$.

Theorem 29 ([12, Theorem 3])**.** *Let Γ be a set of Boolean relations.*

1. *If every relation in Γ is affine then, for any finite subset S of Γ, #CSP(S) is in* FP*.*
2. *Otherwise, if every relation in Γ is in IM_2, then*
   - *for any finite subset S of Γ, #CSP(S) $\leq_{AP}$ #BIS, and*
   - *there is a finite subset S of Γ such that #BIS $\leq_{AP}$ #CSP(S).*
3. *Otherwise, there is a finite subset S of Γ such that #CSP(S) $=_{AP}$ #SAT.*

We also use the following complexity classification from [2]. It is called a "conservative" classification because of the way that unary functions (those in the set S below) are considered, even though they may not belong to $\mathcal{F}$. In essence, this theorem is Theorem 10.2 of [2]. However, we quote the version from [6, Lemma 7], where the ranges of functions are restricted to rational numbers.

Theorem 30 ([2])**.** *Suppose $\mathcal{F}$ is a finite subset of $\mathcal{B}$.*

1. *If $\mathcal{F} \subseteq \mathcal{N}$ then, for any finite subset S of $\mathcal{B}_1$, #CSP($\mathcal{F}, S$) is in* FP*.*
2. *Otherwise,*
   (a) *There is a finite subset S of $\mathcal{B}_1$ such that #BIS $\leq_{AP}$ #CSP($\mathcal{F}, S$).*
   (b) *If $\mathcal{F} \not\subseteq$ LSM then there is a finite subset S of $\mathcal{B}_1$ such that #SAT $=_{AP}$ #CSP($\mathcal{F}, S$).*

The statement of the theorem in [2, 6] only guarantees an FPRAS when $\mathcal{F} \subseteq \mathcal{N}$ but this is because [2] was working over (approximable) real numbers. The proof guarantees an exact algorithm in our setting where elements of the range of functions are rational.

We also require some results about the complexity of symmetric antiferromagnetic two-spin systems (on simple graphs) with external fields. A symmetric two-state spin system with

parameters $(\beta, \gamma, \lambda) \in \mathbb{Q}_{\geq 0}^2 \times \mathbb{Q}_{>0}$ corresponds to the problem #CSP(f, g), where g is the unary function $g(x) = \lambda^{1-x}$, and f is the symmetric binary function

$$f(x, y) = \begin{pmatrix} \beta & 1 \\ 1 & \gamma \end{pmatrix}.$$

The binary constraints are characterised by an undirected simple graph $G = (V, E)$ whose set of vertices V is the set of variables of the instance. Formally, all constraints must satisfy the following conditions:

- For every vertex $v \in V$, there is exactly one constraint of the form $((v), g)$, and there are no other unary constraints.
- For every undirected edge $e = \{v, w\} \in E$, there is exactly one constraint of the form $((v, w), f)$ or $((w, v), f)$ and there are no other constraints with scope (v, w) or (w, v).
- For any pair $v', w' \in V$ such that $\{v', w'\} \notin E$, there are no constraints with scope (v', w') or (w', v').

The spin system is antiferromagnetic if $\beta\gamma < 1$. By Observation 3, this is exactly the same as saying that f is not lsm. The following results additionally assume $\beta \leq \gamma$.

In the following, Δ denotes the maximum degree of the graph describing the binary constraints in the given instance of #CSP(f, g). The expression "$\Delta = \infty$" indicates that the result applies to graphs of unbounded degree.

Theorem 31 ([18, Theorem 1.2])**.** *For any finite $\Delta \geq 3$ or $\Delta = \infty$, there exists an FPTAS for the partition function of the [symmetric] two-state antiferromagnetic spin system on graphs of maximum degree at most Δ if for all $d \leq \Delta$ the system parameters (β, γ, λ) lie in the interior of the uniqueness region of the infinite d-regular tree.*

See also related work of Sinclair, Srivastava and Thurley [21].

Theorem 32 ([22, 13], as stated in [18, Theorem 1.3])**.** *For any finite $\Delta \geq 3$ or $\Delta = \infty$, unless* NP = RP*, there does not exist an FPRAS for the partition function of the [symmetric] two-state antiferromagnetic spin system on graphs of maximum degree at most Δ if for some $d \leq \Delta$ the system parameters (β, γ, λ) lie in the interior of the non-uniqueness region of the infinite d-regular tree.*

These two theorems classify most symmetric antiferromagnetic two-spin system with external fields – the only case that is still open is that of system parameters on the boundary between the uniqueness and non-uniqueness regions.

We will only need a subset of the properties proved in [18, Lemma 3.1] and therefore state only these. In the following, "up-to-Δ unique" means that the system parameters lie in the interior of the uniqueness region of the infinite d-regular tree for every $d \leq \Delta$, and "universally unique" means that the system parameters lie in the interior of the uniqueness region of the infinite d-regular tree for every d.

Lemma 33 ([18, Lemma 3.1])**.** *Let (β, γ, λ) be antiferromagnetic.*

(2) *If $\gamma \leq 1$, then uniqueness does not hold on the infinite d-regular tree for all sufficiently large d.*

(5) *If $\beta = 0$, for any Δ, there exists a critical threshold $\lambda_c = \lambda_c(\gamma, \Delta) = \min_{1<d<\Delta} \frac{\gamma^{d+1} d^d}{(d-1)^{d+1}}$ such that (β, γ, λ) is up-to-Δ unique if and only if $\lambda \in (0, \lambda_c)$.*

(8) *If $\beta > 0$ and $\gamma > 1$, there exists an absolute positive constant $\lambda_c = \lambda_c(\beta, \gamma)$ such that (β, γ, λ) is universally unique if and only if $\lambda \in (0, \lambda_c)$.*

In fact, the proof of (2) in the full version of [18] (where this result is part of Lemma 21) shows that, for sufficiently large d, the system parameters lie in the interior of the non-uniqueness region. The proof of (5) and (8) similarly shows that the system parameters lie in the interior of the non-uniqueness region when $\lambda > \lambda_c$.

3 Counting CSPs with strictly increasing and strictly decreasing permissive unary functions

We first consider the complexity of counting CSPs where two permissive unary functions are available: one which is strictly increasing and one which is strictly decreasing, i.e. problems of the form $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$. The goal is to prove the complexity classification in Theorem 1.

The proof splits into several cases, given here as individual lemmas, depending on which binary functions are contained in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. The function $\mathrm{EQ}(x, y)$ is contained in any (ω, p)-clone, so there always exists some binary function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$.

In some lemmas, we will also assume that the permissive unary functions up and down are normalised. This can be achieved using AP-reductions, as shown in Lemma 24.

3.1 Non-lsm functions with nontrivial binaries

First, we consider the case where $\mathcal{F} \not\subseteq \mathsf{LSM}$ and $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ contains a nontrivial binary function.

Lemma 34. *Let $\mathcal{F} \subseteq \mathcal{B}$, $\mathsf{up} \in \mathcal{B}_1^{<,\mathrm{n}}$, and $\mathsf{down} \in \mathcal{B}_1^{>,\mathrm{n}}$. Suppose $f, g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$, where f is non-lsm and g is binary and nontrivial (but may be lsm). Then $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ contains a binary nontrivial non-lsm function.*

Proof. By Observation 19, $\delta_0, \delta_1 \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$.

Since f is non-lsm, there are $\mathbf{a}, \mathbf{b} \in \{0,1\}^r$, where r is the arity of f, such that

$$f(\mathbf{a})f(\mathbf{b}) > f(\mathbf{a} \wedge \mathbf{b})f(\mathbf{a} \vee \mathbf{b}).$$

Suppose $\mathbf{a}[i] = \mathbf{b}[i]$ for some $i \in [r]$. Then let

$$h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_r) = \sum_{x_i \in \{0,1\}} \delta_{\mathbf{a}[i]}(x_i) f(x_1, \ldots, x_r),$$

this function is contained in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. We say h is f with the i-th input pinned to $\mathbf{a}[i]$. Define $\mathbf{a}' = (\mathbf{a}[1], \ldots, \mathbf{a}[i-1], \mathbf{a}[i+1], \ldots, \mathbf{a}[r])$, and similarly $\mathbf{b}'$. Now,

$$h(\mathbf{a}')h(\mathbf{b}') = f(\mathbf{a})f(\mathbf{b}) > f(\mathbf{a} \wedge \mathbf{b})f(\mathbf{a} \vee \mathbf{b}) = h(\mathbf{a}' \wedge \mathbf{b}')h(\mathbf{a}' \vee \mathbf{b}'),$$

so h is non-lsm. Thus we may continue the proof with h in place of f. This process can be repeated until the bit-strings $\mathbf{a}, \mathbf{b}$, which witness that f is non-lsm, satisfy $\mathbf{a}[i] \neq \mathbf{b}[i]$ for all $i \in [r]$.

Without loss of generality, we may furthermore assume that $\mathbf{a} = (0^s, 1^t)$ and $\mathbf{b} = (1^s, 0^t)$: otherwise permute the arguments, which does not affect the non-lsm property. Now, identify the first s and the last t variables of f to obtain a binary function f' satisfying

$$f'(\mathbf{a}'')f'(\mathbf{b}'') > f'(\mathbf{a}'' \wedge \mathbf{b}'')f'(\mathbf{a}'' \vee \mathbf{b}''),$$

where $\mathbf{a}'' = (0,1), \mathbf{b}'' = (1,0), \mathbf{a}'' \wedge \mathbf{b}'' = (0,0)$ and $\mathbf{a}'' \vee \mathbf{b}'' = (1,1)$.

If $f'(0,0) \neq 0$ or $f'(1,1) \neq 0$, then f' is a binary nontrivial non-lsm function and we are done. Otherwise f' has the form $f''(x)\mathrm{NEQ}(x, y)$, say, $f''(0) = c, f''(1) = d$ with $c, d > 0$.

By the assumptions of the lemma, there is a binary nontrivial function $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. Suppose

$$g(x,y) = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

If g is non-lsm, we are done. Otherwise, we have $\alpha\delta > \beta\gamma$. This inequality is strict because g is not log-modular. Then let $g'(x,y)$ be given by

$$g'(x,y) = \sum_{z \in \{0,1\}} g(x,z) f'(z,y) = \begin{pmatrix} \beta d & \alpha c \\ \delta d & \gamma c \end{pmatrix}.$$

Thus,

$$g'(0,0)g'(1,1) = \beta\gamma cd < \alpha\delta cd = g'(0,1)g'(1,0),$$

that is, g' is non-lsm. It is also nontrivial because g being nontrivial implies that at least one of β, γ is strictly positive. This completes the proof. □

3.2 Log-supermodular functions with nontrivial binaries

Suppose now that $\mathcal{F} \subseteq \mathsf{LSM}$ and $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ contains a nontrivial binary function. The first property is equivalent to $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p} \subseteq \mathsf{LSM}$ by Lemma 21 and the fact that $\mathcal{B}_1 \subseteq \mathsf{LSM}$.

Lemma 35. *Let $\mathcal{F}$ be a finite subset of $\mathcal{B}$ and let $\mathsf{up} \in \mathcal{B}_1^<$, $\mathsf{down} \in \mathcal{B}_1^>$. If $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p} \subseteq \mathsf{LSM}$ and there is a nontrivial binary function $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$, then $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$ is $\#\mathsf{BIS}$-hard.*

Proof. The nontrivial binary function g may be assumed symmetric by replacing it with the function $\sum_{z \in \{0,1\}} g(x,z)g(y,z)$ (which is contained in $\langle g \rangle$) if necessary. By Lemma 5 (2), the proposed replacement function is also nontrivial and lsm. A nontrivial symmetric lsm function can be written as

$$g(x,y) = c \begin{pmatrix} \beta & 1 \\ 1 & \gamma \end{pmatrix},$$

where $\beta\gamma > 1$ (cf. Observation 3) and $c > 0$. Let $g'(x,y) := \frac{1}{c} g(x,y)$ and let $h(x) := \left(\frac{\mathsf{down}(x)}{\mathsf{down}(1)}\right)^k$ for some positive integer k that we will determine below. Note that $h(1) = 1$ and let $\mu := h(0)$, which is strictly greater than 1. Constant factors do not affect the complexity of CSPs, so $\#\mathrm{CSP}(g', h) \leq_{AP} \#\mathrm{CSP}(g, \mathsf{down})$. We now distinguish cases according to the relative size of β and γ.

Case 1. Suppose $\beta < \gamma$. Then by [19, Theorem 2], the problem $\#\mathrm{CSP}(g', h)$ is $\#\mathsf{BIS}$-hard if μ is sufficiently large. But $\mu = \left(\frac{\mathsf{down}(0)}{\mathsf{down}(1)}\right)^k$, which can be made arbitrarily large by choosing k large enough. Hence $\#\mathrm{CSP}(g, \mathsf{down})$ is $\#\mathsf{BIS}$-hard, and thus by Lemma 25, $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$ is $\#\mathsf{BIS}$-hard.

Case 2. Suppose $\beta > \gamma$. By Observation 23, $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down}) =_{AP} \#\mathrm{CSP}(\overline{\mathcal{F}}, \overline{\mathsf{up}}, \overline{\mathsf{down}})$. But $\overline{\mathsf{up}} \in \mathcal{B}_1^>$ and $\overline{\mathsf{down}} \in \mathcal{B}_1^<$, and $\overline{g}(0,0) < \overline{g}(1,1)$. Thus we can apply the argument of Case 1 to the bit-flipped functions to find that $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$ is again $\#\mathsf{BIS}$-hard.

Case 3. Suppose $\beta = \gamma$, i.e. g' is a ferromagnetic Ising function. Historically, this case was considered first, in a paper by Goldberg & Jerrum [14]. However, the main result of [14] (Theorem 1.1) is stated in a setting where the input has "local fields", which means that different unary functions are available for different variables. While [14] does contain a reduction from a problem with restricted local fields to the general problem, the proof of this result is only given for an explicit choice of fields. This proof is widely known to generalise, as e.g. noted in [19], but rather than writing out the general proof here, it will be shorter (if ahistorical) to note the following: Let $g''(x,y) = \sum_{z \in \{0,1\}} g(x,z)g(y,z)\mathsf{up}(z)$. By Lemma 5 (3), this function

is nontrivial, symmetric, lsm, and non-Ising. Thus the problem reduces to one of the previous cases.

Therefore, #CSP($\mathcal{F}, \mathsf{up}, \mathsf{down}$) is #BIS-hard whenever $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p} \subseteq \mathsf{LSM}$ and there exists a nontrivial binary function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. □

3.3 All binary functions are trivial

Having considered two cases with nontrivial binaries, we now look at (ω, p)-clones that do not contain any nontrivial binary functions. In the following, $\oplus_3(x, y, z)$ is the ternary indicator function for inputs of even parity, i.e.

$$\oplus_3(x, y, z) = \begin{cases} 1 & \text{if } x + y + z \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}$$

We will also use the following two lemmas.

Lemma 36 ([23], as stated in [2, Lemma 5.1])**.** *A permissive function $f \in \mathcal{B}$ is log-modular if and only if every 2-pinning is log-modular.*

Lemma 37. *A permissive function $f \in \mathcal{B}_n$ is log-modular if and only if it is a product of permissive unary functions, i.e. $f(x_1, \ldots, x_n) = \prod_{i=1}^n u_i(x_i)$, where $u_i \in \mathcal{B}_1$ is permissive for all $i \in [n]$.*

Proof. Any product of permissive unary functions must be permissive and log-modular, i.e. the "if" direction is immediate. It remains to prove that if f is permissive and log-modular, then it must be a product of permissive unary functions. For $n = 1$ that result is trivial and for $n = 2$ it follows straightforwardly from the definition of log-modularity.

Now assume the desired result holds for some $n \geq 2$ and consider a permissive log-modular function $f \in \mathcal{B}_{n+1}$. Define $f_a(x_1, \ldots, x_n) := f(x_1, \ldots, x_n, a)$ for $a \in \{0, 1\}$, then

$$f(x_1, \ldots, x_{n+1}) = f_0(x_1, \ldots, x_n)\delta_0(x_{n+1}) + f_1(x_1, \ldots, x_n)\delta_1(x_{n+1}).$$

Since f is permissive and log-modular, f_0 and f_1 must also be permissive and log-modular. Hence by the inductive assumption, there exist permissive unary functions $u_1, \ldots, u_n \in \mathcal{B}_1$ such that $f_0(x_1, \ldots, x_n) = \prod_{i=1}^n u_i(x_i)$, and there exist permissive unary functions $v_1, \ldots, v_n \in \mathcal{B}_1$ such that $f_1(x_1, \ldots, x_n) = \prod_{i=1}^n v_i(x_i)$. Thus we can write

$$f(x_1, \ldots, x_{n+1}) = \left(\prod_{i=1}^n u_i(x_i) \right) \delta_0(x_{n+1}) + \left(\prod_{i=1}^n v_i(x_i) \right) \delta_1(x_{n+1}).$$

Let $k \in [n]$ be arbitrary and consider some 2-pinning of f that leaves the variables k and $n + 1$ untouched, pinning each variable $i \in [n] \setminus \{k\}$ to the value b_i. This yields the function

$$g_k(x_k, x_{n+1}) := \left(\prod_{i \in [n] \setminus \{k\}} u_i(b_i) \right) u_k(x_k)\delta_0(x_{n+1}) + \left(\prod_{i \in [n] \setminus \{k\}} v_i(b_i) \right) v_k(x_k)\delta_1(x_{n+1}).$$

By Lemma 36, g_k is log-modular, i.e.

$$0 = g_k(0,0)g_k(1,1) - g_k(0,1)g_k(1,0) = \left(\prod_{i \in [n] \setminus \{k\}} u_i(b_i)v_i(b_i) \right) \left(u_k(0)v_k(1) - v_k(0)u_k(1) \right).$$

Since all the unary functions u_i and v_i are permissive and take non-negative rational values, this implies that there exists $c_k \in \mathbb{Q}_{>0}$ such that $v_k = c_k \cdot u_k$. But k was arbitrary, therefore

$$f(x_1, \ldots, x_{n+1}) = \left(\prod_{i=1}^n u_i(x_i) \right) \left(\delta_0(x_{n+1}) + \delta_1(x_{n+1}) \prod_{i=1}^n c_i \right) = \prod_{i=1}^{n+1} u_i(x_i),$$

where $u_{n+1}(0) = 1$ and $u_{n+1}(1) = \prod_{i=1}^{n} c_i$, so u_{n+1} is permissive and in $\mathcal{B}_1$. Hence f has the desired form, completing the proof. □

Lemma 38. *Let $\mathcal{F} \subseteq \mathcal{B}$, $\mathsf{up} \in \mathcal{B}_1^{<,\mathrm{n}}$, and $\mathsf{down} \in \mathcal{B}_1^{>,\mathrm{n}}$. Suppose $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ does not contain any nontrivial binary functions and $\mathcal{F} \not\subseteq \mathcal{N}$. Then there exists a ternary function $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ satisfying $g(x, y, z) = c \cdot \oplus_3(x, y, z)$, where $c > 0$ is a constant.*

Proof. The proof of the lemma has two parts. First, we show that if $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ does not contain any nontrivial binary functions, then every function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ has affine support. Next, we show that if every function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ has affine support and $\mathcal{F} \not\subseteq \mathcal{N}$, then there exists $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ which satisfies $g(x, y, z) = c \cdot \oplus_3(x, y, z)$ for some non-zero constant c.

To prove that every function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ has affine support, we assume the opposite and show this leads to a contradiction. In particular, assume there exists a function $f \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ which does not have affine support. All unary functions have affine support, so f cannot be unary. Furthermore, all binary functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ are trivial and a trivial binary function is either log-modular or it has a support on exactly 2 inputs. Recall from Section 2.1 that any log-modular binary function $b(x, y)$ can be written as $u(x)v(y)$ for some $u, v \in \mathcal{B}_1$; hence it has support on 1, 2 or 4 inputs. Now any subset of $\{0, 1\}^2$ of size 1, 2 or 4 is an affine relation – in other words all trivial binary functions have affine support. Therefore, the function f cannot be binary, so it has arity at least 3.

By Observation 19, the pinning functions δ_0 and δ_1 are contained in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. Now, the proof of Lemma 11 in [11] shows that, if f is a nonnegative function with $\mathrm{arity}(f) > 2$ and f does not have affine support, then $\langle f, \delta_0, \delta_1 \rangle$ (and thus $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$) contains a function of arity 2 which does not have affine support. This contradicts the assumption that all binary functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ are trivial. Therefore, every function in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ must have affine support.

To prove that $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ contains a scaled parity function, suppose there exists a function $h \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p} \setminus \mathcal{N}$. All unary functions are contained in $\mathcal{N}$ since they are generators of the functional clone $\mathcal{N}$. Furthermore, as noted in Section 2.1, any binary log-modular function can be written as $u(x)u'(y)$, where $u, u' \in \mathcal{B}_1$ are appropriate unary functions. Thus, all trivial binary functions are contained in $\mathcal{N}$. Hence, h has arity at least 3. It is not the all-zero function, as that is also contained in $\mathcal{N}$. By the first part of this proof, h has affine support. Since all binary functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ are trivial, all 2-pinnings of h (as defined in Section 2) must be trivial.

We now consider the relations underlying some of the functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. Recall from Section 2 that we denote by R_f the relation underlying a function f. According to Lemma 3.1 of [2], for any set of nonnegative functions $\mathcal{G}$, we have

$$\langle \{R_f \mid f \in \mathcal{G}\} \rangle_R = \{R_f \mid f \in \langle \mathcal{G} \rangle\}. \tag{3}$$

Note that the above result is about functional clones, not (ω, p)-clones, and allowing limits may change the set of underlying relations, as can be seen for example in Observation 19. Yet it is straightforward to see that

$$\langle \{R_f \mid f \in \mathcal{G}\} \rangle_R = \{R_f \mid f \in \langle \mathcal{G} \rangle\} \subseteq \{R_f \mid f \in \langle \mathcal{G} \rangle_{\omega,p}\}.$$

Furthermore, we already know that $\{R_f \mid f \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}\}$ contains only affine relations. These are the only facts we use below.

Let R_h be the relation underlying h, which is affine and non-empty. It can then be seen from Table 2 and the proof of Proposition 3 of [7] that the relational clone $\langle R_h, \delta_0, \delta_1 \rangle_R$ must be either $\mathrm{IR}_2 = \langle \mathrm{EQ}, \delta_0, \delta_1 \rangle_R$ or $\mathrm{ID}_1 = \langle \mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1 \rangle_R$ or IL_2, the relational clone containing all affine relations. For an example of the argument for this, see the proof of Theorem 9.1 in

[2]. The argument in the next few paragraphs is also (a simplified version of) an argument in that proof.

Suppose that $\langle R_h, \delta_0, \delta_1\rangle_R \subseteq \mathrm{ID}_1$ (i.e. $\langle R_h, \delta_0, \delta_1\rangle_R = \mathrm{IR}_2$ or $\langle R_h, \delta_0, \delta_1\rangle_R = \mathrm{ID}_1$), then in particular $R_h \in \mathrm{ID}_1$. The relation R_h is also non-empty because $h \notin \mathcal{N}$. Since R_h is affine, its elements are solutions to a set of linear equations (cf. Section 2). We can thus find a partition of the arguments of h into a set of *free* variables and a set of *dependent* variables such that for any assignment of values to the free variables there exists a unique assignment of values to the dependent variables for which the resulting tuple of bit values is in R_h. Let $n := \mathrm{arity}(h)$ and assume without loss of generality that $x_1, \ldots, x_k$ are the free variables and $x_{k+1}, \ldots, x_n$ are the dependent variables (if necessary, permute and rename variables). Define $h'(x_1, \ldots, x_k) := \sum_{x_{k+1},\ldots,x_n \in \{0,1\}} h(x_1, \ldots, x_n)$, then $h' \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$ and h' is permissive.

As $h' \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$, all its 2-pinnings are binary functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$ and are thus trivial by assumption. Since h' is permissive, by Observation 4 its 2-pinnings can be trivial only if they are log-modular. Therefore, by Lemma 36, h' is log-modular. Furthermore, by Lemma 37, h' is a product of unary functions, so $h' \in \mathcal{N}$.

Now, in going from h to h', we summed out the dependent variables. Thus, for any fixed $x_1, \ldots, x_k$, there is only a single assignment of $x_{k+1}, \ldots, x_n$ such that $h(x_1, \ldots, x_n)$ is non-zero. Therefore, $h(x_1, \ldots, x_n) = \chi_h(x_1, \ldots, x_n)h'(x_1, \ldots, x_k)$, where χ_h is the indicator function for the relation R_h – i.e. $\chi_h(x_1, \ldots, x_n) = 1$ if $(x_1, \ldots, x_n) \in R_h$ and $\chi_h(x_1, \ldots, x_n) = 0$ otherwise. We assumed $R_h \in \mathrm{ID}_1 = \langle \mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1\rangle_R$, therefore $\chi_h \in \mathcal{N}$ and thus $h \in \langle \chi_h, h'\rangle \subseteq \mathcal{N}$. But this contradicts the assumption that $h \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p} \setminus \mathcal{N}$. So instead we must have $\langle R_h, \delta_0, \delta_1\rangle_R \nsubseteq \mathrm{ID}_1$. The only way for this to happen is if $\langle R_h, \delta_0, \delta_1\rangle_R = \mathrm{IL}_2$ and thus $\mathrm{IL}_2 \subseteq \{R_f \mid F \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}\}$.

Now, the relation corresponding to $\oplus_3$ is

$$R_{\oplus_3} = \{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}.$$

This is affine and therefore contained in $\mathrm{IL}_2 = \langle R_h, \delta_0, \delta_1\rangle_R$. Hence, by letting $\mathcal{G}$ equal $\langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$ in (3), there must be a ternary function $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$ which has $R_{\oplus_3}$ as its underlying relation.

Let $a = g(0,0,0)$, $b = g(0,1,1)$, $c = g(1,0,1)$, and $d = g(1,1,0)$ be the non-zero values of g. Now consider the binary function $g'(x,y) = \sum_z g(x,y,z)$, which takes the form

$$g'(x,y) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

By construction, $g' \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$, so it must be trivial. As $a, b, c, d > 0$, g' must be log-modular, i.e. $ad = bc$. Similarly, by summing out y or x and applying the log-modularity condition to the resulting binary function, we deduce $ac = bd$ and $ab = cd$. Multiply together the first two of these equations to get $a^2cd = b^2cd$, which implies $a = b$ since all four values are positive. By symmetry, we find that in fact $a = b = c = d$, i.e. $g(x,y,z) = c \cdot \oplus_3(x,y,z)$. □

Lemma 39. *Let $\mathcal{F}$ be a finite subset of $\mathcal{B}$ and let $\mathsf{up} \in \mathcal{B}_1^<$, $\mathsf{down} \in \mathcal{B}_1^>$. Suppose $\langle \mathcal{F}, \mathsf{up}, \mathsf{down}\rangle_{\omega,p}$ contains a function $g(x,y,z) = c \cdot \oplus_3(x,y,z)$ for some constant $c > 0$. Then the problem $\#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.*

Proof. We show the desired result by reduction from the problem of approximating weight enumerators of linear codes. The definition and reduction follow [11, p. 1977 and Lemma 13], with some modifications because that paper was concerned with hardness of exact evaluation.

A *linear code* is specified by a binary generating matrix A. Let Υ be the linear subspace generated by the rows of A over GF(2), then any vector in Υ is a code word. The *weight enumerator* of the code specified by A with weight parameter $\lambda \in \mathbb{Q}$ is given by $W_A(\lambda) := \sum_{\mathbf{v} \in \Upsilon} \lambda^{|\mathbf{v}|}$, where $|\mathbf{v}|$ is the Hamming weight of $\mathbf{v}$. The computational problem $\mathrm{WE}(\lambda)$ takes as input a matrix A and outputs $W_A(\lambda)$.

The linear space Υ can be specified by a pure affine function h_A taking values in $\{0,1\}$, with the arity of h_A being equal to the number of columns of A. The set of $\{0,1\}$-valued pure affine functions is in bijection with the set of affine relations IL_2. Now, by adding new variables and breaking linear equations into pieces, we have $\langle \oplus_3, \delta_0, \delta_1 \rangle_R = \mathrm{IL}_2$. Thus, in particular, $h_A \in \langle \oplus_3, \delta_0, \delta_1 \rangle$ for any A. Define $u_\lambda(x) = \lambda^x$ for $x \in \{0,1\}$, then $W_A(\lambda) = \sum_{x_1,\ldots,x_n \in \{0,1\}} h_A(x_1,\ldots,x_n) \prod_{i=1}^n u_\lambda(x_i)$. Therefore, by Lemma 25,

$$\mathrm{WE}(\lambda) \leq_{AP} \#\mathrm{CSP}(\oplus_3, u_\lambda, \delta_0, \delta_1).$$

Suppose that $\lambda = \frac{\mathsf{up}(1)}{\mathsf{up}(0)}$, then $u_\lambda \in \mathcal{B}_1$ and $\mathsf{up}(0) \cdot u_\lambda = \mathsf{up}$. We have

$$\#\mathrm{CSP}(\oplus_3, u_\lambda, \delta_0, \delta_1) \leq_{AP} \#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down}, \oplus_3, u_\lambda, \delta_0, \delta_1) \leq_{AP} \#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$$

where the first reduction is because adding more constraint functions cannot make the problem easier and the second reduction is by repeated applications of Lemma 25, since all of $c \cdot \oplus_3$, $\mathsf{up}(0) \cdot u_\lambda, \delta_0, \delta_1$ are in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ and constant factors can be absorbed into AP-reductions. Hence, combining the different reductions, we find $\mathrm{WE}\left(\frac{\mathsf{up}(1)}{\mathsf{up}(0)}\right) \leq_{AP} \#\mathrm{CSP}(\mathcal{F}, \mathsf{up}, \mathsf{down})$.

Now by [15, Corollary 7], the problem of approximating the weight enumerator of a linear code with weight parameter $\lambda > 1$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$. But $\lambda = \frac{\mathsf{up}(1)}{\mathsf{up}(0)} > 1$ by the definition of up, hence the desired result follows. □

3.4 Putting the pieces together

Recall Theorem 1, which we can now prove.

Theorem 1 (restated). *Let* up *be a permissive unary strictly increasing function, let* down *be a permissive unary strictly decreasing function, and let* $\mathcal{F} \subseteq \mathcal{B}$*. Then the following properties hold.*

1. *If* $\mathcal{F} \subseteq \mathcal{N}$*, then, for any finite subset* S *of* $\mathcal{F}$*,* $\#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\})$ *is in* FP*.*
2. *Otherwise, if* $\mathcal{F} \subseteq \mathsf{LSM}$*, then*
   (a) *there is a finite subset* S *of* $\mathcal{F}$ *such that* $\#\mathsf{BIS} \leq_{AP} \#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\})$*, and*
   (b) *for every finite subset* S *of* $\mathcal{F}$ *such that all functions* $f \in S$ *have arity at most 2,* $\#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\}) \leq_{AP} \#\mathsf{BIS}$*.*
3. *Otherwise, there is a finite subset* S *of* $\mathcal{F}$ *such that* $\#\mathrm{CSP}(S \cup \{\mathsf{up}, \mathsf{down}\})$ *does not have an FPRAS unless* $\mathsf{NP} = \mathsf{RP}$*.*

Proof. By Lemma 24, there exist functions $\mathsf{up}' \in \mathcal{B}_1^{<,\mathrm{n}}$ and $\mathsf{down}' \in \mathcal{B}_1^{>,\mathrm{n}}$ such that

$$\#\mathrm{CSP}(S, \mathsf{up}, \mathsf{down}) =_{AP} \#\mathrm{CSP}(S, \mathsf{up}', \mathsf{down}')$$

for any finite $S \subseteq \mathcal{F}$. By replacing the problem on the left-hand side with the one on the right-hand side, we may therefore assume in the following that the permissive unary functions up and down are normalised.

Property 1 follows from Theorem 30.

For Property 2a, suppose $\mathcal{F} \subseteq \mathsf{LSM}$ and $\mathcal{F} \not\subseteq \mathcal{N}$. If all binary functions in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ are trivial, then by Lemma 38 there is a ternary function $g \in \langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ satisfying $g(x,y,z) = c \cdot \oplus_3(x,y,z)$ where $c > 0$. Note that

$$g(0,1,1)g(1,0,1) = c^2 > 0 = g(0,0,1)g(1,1,1)$$

i.e. g is not lsm. But this is a contradiction as the set of lsm functions is closed under taking ω-clones by Lemma 21. Hence we may assume that there is a nontrivial binary function f in

$\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. By the definition of (ω, p)-clone (Definition 18) there is a finite subset S_f of $\mathcal{F}$ such that $f \in \langle S_f, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. Then by Lemma 35, $\#\mathrm{CSP}(S_f, \mathsf{up}, \mathsf{down})$ is $\#\mathsf{BIS}$-hard.

Property 2b follows from Part 3 of [6, Theorem 6], noting that the property of "weak log-supermodularity" used there encompasses all binary log-supermodular functions.

Finally, suppose $\mathcal{F}$ is not a subset of $\mathcal{N}$, nor is it a subset of LSM. Then, if $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$ contains a nontrivial binary function, by Lemma 34 it contains a nontrivial binary non-lsm function g. It then contains the function $g(x,y)g(y,x)$, which is symmetric, nontrivial and non-lsm by Lemma 5 (1). This function can be written as

$$g'(x,y) = d \begin{pmatrix} \beta & 1 \\ 1 & \gamma \end{pmatrix}$$

for some $d > 0$ and $\beta, \gamma \geq 0$. By the definition of (ω, p)-clone, there is a finite subset S of $\mathcal{F}$ such that $g' \in \langle S, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$. We distinguish two subcases.

Case 1. Suppose $\beta \leq \gamma$. As g' is non-lsm, by Observation 3, we have $\beta\gamma < 1$: i.e., the function corresponds to an antiferromagnetic two-spin model. Let $f(x,y) = d^{-1}g'(x,y)$ and let $h(x) = \left(\frac{\mathsf{down}(x)}{\mathsf{down}(1)}\right)^k$ for some sufficiently large positive integer k (to be determined below). Then $h(1) = 1$; set $\lambda := h(0)$. Since constant factors can be absorbed into AP-reductions, $\#\mathrm{CSP}(f,h) =_{AP} \#\mathrm{CSP}(g', \mathsf{down})$. Now $\#\mathrm{CSP}(f,h)$ corresponds to the spin system (β, γ, λ), cf. Section 2.5. We have $0 \leq \beta \leq \gamma$ with $\beta\gamma < 1$ and λ can be made arbitrarily large by choosing k large enough.

- If $\gamma \leq 1$, then Lemma 33(2) says that for sufficiently large d, uniqueness does not hold on the infinite d-regular tree.
- Otherwise, if $\beta = 0$, then by Lemma 33(5), (β, γ, λ) is not up-to-Δ unique if λ is large enough.
- Otherwise we must have $\beta > 0$ and $\gamma > 1$. In that case, by Lemma 33(8), for large enough λ, universal uniqueness does not hold.

In each of these cases, the proof of Lemma 33 in the full version of [18] shows that (β, γ, λ) actually lies in the interior of the non-uniqueness region. Now, if (β, γ, λ) is in the interior of the non-uniqueness region, Theorem 32 (due to [22, 13]) shows that $\#\mathrm{CSP}(f,h)$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$. Hence, $\#\mathrm{CSP}(g', \mathsf{down})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.

Case 2. Suppose $\beta > \gamma$. By Observation 23, $\#\mathrm{CSP}(g', \mathsf{up}) =_{AP} \#\mathrm{CSP}(\overline{g'}, \overline{\mathsf{up}})$. Now

$$\overline{g'}(x,y) = d \begin{pmatrix} \gamma & 1 \\ 1 & \beta \end{pmatrix},$$

so $\overline{g'}(0,0) < \overline{g'}(1,1)$. Furthermore, $\overline{\mathsf{up}} \in \mathcal{B}_1^{>}$. Hence, by the argument of Case 1, $\#\mathrm{CSP}(\overline{g'}, \overline{\mathsf{up}})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$ and so $\#\mathrm{CSP}(g', \mathsf{up})$ does not have an FPRAS unless that condition is satisfied.

If, instead, there are no nontrivial binaries in $\langle \mathcal{F}, \mathsf{up}, \mathsf{down} \rangle_{\omega,p}$, we get the property of not having an FPRAS unless $\mathsf{NP} = \mathsf{RP}$ from Lemmas 38 and 39. This completes the proof of Property 3 and thus of the theorem. □

4 Two-spin

In this section, we partially classify the complexity of the problem $\#\mathrm{CSP}(f)$, where $f \in \mathcal{B}_2$. This is similar to the two-spin problem widely studied in statistical physics and computer science: the relevant quantity is the partition function arising from the pairwise interactions of neighbouring spins. Progress so far has mainly been restricted to the symmetric case, but we consider more

general interaction matrices. From Theorem 30, we know that #CSP(f) is in FP if $f \in \mathcal{N}$. We furthermore give a complete classification of the complexity of #CSP(f) if f is lsm, as well as determining the complexity if f is non-lsm and both f and $\bar{f}$ are non-monotone.

Recall from Section 2.1 that a binary function is *trivial* if it is log-modular or if $f(x,y) = g(x)\mathrm{EQ}(x,y)$ or $f(x,y) = g(x)\mathrm{NEQ}(x,y)$ for some unary function $g \in \mathcal{B}_1$. Recall also that $\widehat{f}$ is the Fourier transform of f (cf. Section 2.2). We often write $\widehat{f}_{xy}$ to denote $\widehat{f}(x,y)$.

The proof of Theorem 2 is split into various lemmas that are stated and proved individually before being assembled into the proof of the theorem in Section 4.5. Throughout, we write

$$f(x,y) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

4.1 Functions whose middle Fourier coefficients have opposite signs

First, we show that we can realise a strictly increasing and a strictly decreasing permissive unary function if the middle Fourier coefficients of f, i.e. $\widehat{f}_{01}$ and $\widehat{f}_{10}$, have opposite signs. Availability of these two unary functions reduces the problem to the case considered in Section 3.

Lemma 40. *Let $f \in \mathcal{B}_2$ be a nontrivial binary function and suppose $\widehat{f}_{01}\widehat{f}_{10} < 0$. Then both $\langle f \rangle \cap \mathcal{B}_1^<$ and $\langle f \rangle \cap \mathcal{B}_1^>$ are non-empty.*

Proof. The condition $\widehat{f}_{01}\widehat{f}_{10} < 0$ implies that the Fourier coefficients have opposite signs. Without loss of generality, we may assume that $\widehat{f}_{01} > 0$ and $\widehat{f}_{10} < 0$, i.e. (cf. Section 2.2)

$$\begin{aligned} a + c &> b + d \\ a + b &< c + d. \end{aligned}$$

Otherwise, replace $f(x,y)$ by $f''(x,y) := f(y,x)$. Then, by Observation 8, $\widehat{f''}_{01} = \widehat{f}_{10} > 0$ and $\widehat{f''}_{10} = \widehat{f}_{01} < 0$ so the replacement function satisfies the desired property.

Let $\mathsf{up}(x) := \sum_y f(x,y)$ and $\mathsf{down}(y) := \sum_x f(x,y)$; then $\mathsf{up}, \mathsf{down} \in \langle f \rangle$. These functions satisfy:

$$\begin{aligned} \mathsf{down}(0) = a + c > b + d &= \mathsf{down}(1) \\ \mathsf{up}(0) = a + b < c + d &= \mathsf{up}(1). \end{aligned}$$

Furthermore, since f is nonnegative and nontrivial, up and down are permissive. Hence, $\mathsf{up} \in \langle f \rangle \cap \mathcal{B}_1^<$ and $\mathsf{down} \in \langle f \rangle \cap \mathcal{B}_1^>$, as desired. □

Thus, if f satisfies $\widehat{f}_{01}\widehat{f}_{10} < 0$, we can use the complexity classification from Theorem 1.

4.2 Log-supermodular functions

The case of an lsm function whose middle Fourier coefficients have opposite signs is included in Lemma 40. Thus, it only remains to consider the case of an lsm function where the two middle Fourier coefficients have the same sign (or at least one of them is zero).

Lemma 41. *Suppose f is a nontrivial binary lsm function with $\widehat{f}_{01}\widehat{f}_{10} \geq 0$. Then $\widehat{f}_{11} \geq 0$.*

Proof. By Observation 3, the lsm condition is $ad > bc$, where the inequality is strict as f is nontrivial. We distinguish cases according to whether the Fourier coefficients are both zero, both nonpositive or both nonnegative.

Case 1. $\widehat{f}_{01} = \widehat{f}_{10} = 0$, i.e.

$$\begin{aligned} a + c &= b + d, \\ a + b &= c + d. \end{aligned}$$

This implies $a = d$ and $b = c$, i.e. f is an Ising function. Now,

$$b + c = 2\sqrt{bc} < 2\sqrt{ad} = a + d,$$

where the first step uses $b = c$, the second step the lsm property, and the last step uses $a = d$. But $a + d > b + c$ is equivalent to $\widehat{f}_{11} > 0$, the desired result.

Case 2. $\widehat{f}_{01}, \widehat{f}_{10} \leq 0$ and $\widehat{f}_{01}, \widehat{f}_{10}$ are not both zero. Without loss of generality, assume that $\widehat{f}_{01} < 0$, the argument is analogous if $\widehat{f}_{10} < 0$ instead. Thus, we have

$$a + c < b + d, \tag{4}$$
$$a + b \leq c + d, \tag{5}$$

as well as the lsm condition $ad > bc$. We will show the result by contradiction, i.e. assume for a contradiction that $\widehat{f}_{11} < 0$ or, equivalently,

$$a + d < b + c. \tag{6}$$

Adding the inequalities (4) and (5), we have $a < d$. Adding (4) and (6) gives $a < b$, and adding up (5) and (6) we obtain $a < c$, with all of these inequalities being strict. Furthermore, the lsm condition $ad > bc$ now implies $b, c < d$.

Next we show that $a^{2^k} + d^{2^k} < b^{2^k} + c^{2^k}$ for any k. Since $d > a, b, c$, this is a contradiction if k is large enough. We prove the property by induction on k. The base case, $k = 0$, is $\widehat{f}_{11} < 0$. For the induction step, suppose $a^{2^k} + d^{2^k} < b^{2^k} + c^{2^k}$ for some $k \in \mathbb{N}$. By squaring both sides of this inequality we get

$$a^{2^{k+1}} + 2a^{2^k} d^{2^k} + d^{2^{k+1}} < b^{2^{k+1}} + 2b^{2^k} c^{2^k} + c^{2^{k+1}}. \tag{7}$$

Now, $ad > bc$ implies $a^{2^k} d^{2^k} > b^{2^k} c^{2^k}$. Subtracting this from (7) we find

$$a^{2^{k+1}} + d^{2^{k+1}} < b^{2^{k+1}} + c^{2^{k+1}},$$

so the property does indeed hold for all k. Yet, since d is strictly the largest of the four values, this inequality cannot be true for large k, a contradiction. Thus, the assumption $\widehat{f}_{11} < 0$ must have been false and in this case, too, we have $\widehat{f}_{11} \geq 0$.

Case 3. $\widehat{f}_{01}, \widehat{f}_{10} \geq 0$ and $\widehat{f}_{01}, \widehat{f}_{10}$ are not both zero. Let $f'(x, y) = \bar{f}(x, y)$, where we are renaming the function to avoid clumsy notation. Note that, if f is nontrivial and lsm, then so is f'. By Observation 7, we have $\widehat{f'}_{00} = \widehat{f}_{00}$, $\widehat{f'}_{01} = -\widehat{f}_{01}$, $\widehat{f'}_{10} = -\widehat{f}_{10}$, and $\widehat{f'}_{11} = \widehat{f}_{11}$. Now, f' is a nontrivial binary lsm function with $\widehat{f'}_{01}, \widehat{f'}_{10} \leq 0$ and $\widehat{f'}_{01}, \widehat{f'}_{10}$ not both zero. Thus, by Case 2 of this proof, $\widehat{f'}_{11}$ is nonnegative. This immediately implies that $\widehat{f}_{11} \geq 0$.

Thus, by exhaustive case analysis, we have shown that $\widehat{f}_{11} \geq 0$ whenever $\widehat{f}_{01}\widehat{f}_{10} \geq 0$. □

4.3 FPRAS for binary functions with nonnegative Fourier coefficients

Recall from Section 2.2 that $\mathcal{P}$ is the set of functions in $\mathcal{B}$ whose Fourier transform takes nonnegative values. Recall from Section 2.4 that, for a finite set of constraint functions $\mathcal{F}$, the problem $\mathsf{Holant}(\mathcal{F})$ is the restriction of $\#\mathrm{CSP}(\mathcal{F})$ to instances where every variable appears exactly twice.

In this section, we show that for all binary functions $f \in \mathcal{P} \cap \mathcal{B}_2$, the problem $\#\mathrm{CSP}(f)$ has an FPRAS. This result, Corollary 56 below, arises as a corollary of a stronger statement (Theorem 55).

An arity-k function is said to be *self-dual* if $f(x_1, \ldots, x_k) = f(\bar{x}_1, \ldots, \bar{x}_k)$. Let SDP be the set of self-dual functions in $\mathcal{P}$. The set SDP, introduced in [1], is a functional clone. Let $\mathsf{SDP}_3 := \mathsf{SDP} \cap \mathcal{B}_3$. We show in Theorem 55 that, for any finite subset $\mathcal{F} \subseteq \mathsf{SDP}_3$, the problem

#CSP($\mathcal{F}$) has an FPRAS. This is a somewhat surprising result because there are functions in SDP_3 that are not log-supermodular. See [1, Theorem 14].

To prove Theorem 55, in Lemma 46 we transform #CSP($\mathcal{F}$) to a suitable "even subgraphs" holant problem. In Lemma 48 we find a bound on the weight of "near-assignments" of this holant problem. In Lemma 51 we reduce the holant problem to a perfect matchings problem. In Lemma 52 we bound the number of nearly perfect matchings in terms of near-assignments. Finally, we apply a result of Jerrum and Sinclair (Lemma 54, from [17]) to approximate the solution to the perfect matching problem.

Corollary 56 follows from Theorem 55 via an AP-reduction #CSP(f) $\leq_{AP}$ #CSP(f'), where $f'(x,y,z) := f(x \oplus z, y \oplus z)$ is shown to be in SDP_3 for any $f \in \mathcal{P} \cap \mathcal{B}_2$.

Our proofs will use the following characterisation of SDP.

Lemma 42 ([1, Lemma 38])**.** *Suppose $f \in \mathcal{B}$, then $f \in \mathsf{SDP}$ if and only if the Fourier transform of f is nonnegative on inputs of even Hamming weight and is zero on inputs of odd Hamming weight.*

As before, let $\oplus_3$ be the ternary indicator function for inputs of even parity:

$$\oplus_3(x,y,z) = \begin{cases} 1 & \text{if } x+y+z \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}$$

The following observation is well-known and arises frequently in holographic transformations. See, e.g., [3]. It also arises as a special case of [1, Lemma 27].

Observation 43. The Fourier transform of $\oplus_3$ is $\widehat{\oplus_3} = \frac{1}{2}\mathrm{EQ}_3$.

Lemma 44. *Suppose $f \in \mathcal{B}_2$ and let $f'(x,y,z) := f(x \oplus z, y \oplus z)$. Then*

$$\widehat{f'}(x,y,z) = \widehat{f}(x,y) \cdot \oplus_3(x,y,z).$$

Furthermore, if $f \in \mathcal{P}$, then $f' \in \mathsf{SDP}_3$.

Proof. Let $g(x,y,z) := \widehat{f}(x,y) \oplus_3 (x,y,z)$, so in the first part of the lemma we are aiming to prove $\widehat{f'} = g$. Note that by Observation 6 and the definition of the Fourier transform, we have

$$f'(x,y,z) = 8\widehat{\widehat{f'}}(x,y,z). \tag{8}$$

By taking the Fourier transform, the desired equality $\widehat{f'} = g$ is equivalent to $\widehat{\widehat{f'}} = \widehat{g}$. Substituting into (8), this is equivalent to $f' = 8\widehat{g}$. For the first part of the lemma, it thus suffices to show that the Fourier transform of g is equal to $\frac{1}{8}f'$; this is more technically elegant than evaluating the Fourier transform of f' directly.

It will be useful to first define another function related to f. Let $h(x,y,z) = \widehat{f}(x,y)$ be the function arising from $\widehat{f}$ by introduction of a fictitious argument. By [1, Lemma 25], we have $\widehat{h}(x,y,1) = 0$ and

$$\widehat{h}(x,y,0) = \widehat{\widehat{f}}\,(x,y) = \frac{1}{4}\sum_{p,q\in\{0,1\}} (-1)^{px+qy}\widehat{f}(p,q) = \frac{1}{4}f(x,y),$$

where the second equality is the definition of the Fourier transform of $\widehat{f}$ and the third is by Observation 6.

It is straightforward to see that $g(x,y,z) = h(x,y,z) \oplus_3 (x,y,z)$. Thus, the Fourier transform of g can be expressed as a convolution:[1]

$$\widehat{g}(x,y,z) = \sum_{p,q,r\in\{0,1\}} \widehat{h}(p,q,r)\widehat{\oplus_3}(p \oplus x, q \oplus y, r \oplus z).$$

[1] This is a well-known result in the theory of Fourier transforms. For pseudo-Boolean functions it follows from [9] together with the straightforward-to-derive property that Fourier transforms are involutive up to scalar factor.

Substitute for $\widehat{h}$, and for $\widehat{\oplus_3}$ using Observation 43, then the Fourier transform becomes

$$\widehat{g}(x,y,z) = \sum_{p,q\in\{0,1\}} \frac{1}{4} f(p,q) \frac{1}{2} \mathrm{EQ}_3(p\oplus x, q\oplus y, z).$$

Now, the ternary equality function enforces $p\oplus x = z$ and $q\oplus y = z$, which is equivalent to $p = x\oplus z$ and $q = y\oplus z$. Hence, the expression simplifies to

$$\widehat{g}(x,y,z) = \frac{1}{8} f(x\oplus z, y\oplus z) = \frac{1}{8} f'(x,y,z),$$

the desired result.

For the second part, note that $f\in\mathcal{P}$ implies that f has a nonnegative Fourier transform. Then, $\widehat{f'}$ is the product of two nonnegative functions, so it is nonnegative, and therefore $f'\in\mathcal{P}$. Yet $\widehat{f'}$ is 0 on inputs of odd Hamming weight because of the factor of $\oplus_3$. Thus, by Lemma 42, $f'\in\mathsf{SDP}_3$. □

Lemma 45. *Suppose $f\in\mathcal{B}_2$ and $f'(x,y,z) = f(x\oplus z, y\oplus z)$, then $\#\mathrm{CSP}(f)\le_{AP}\#\mathrm{CSP}(f')$.*

Proof. Consider an instance $\Omega = (V,C)$ of $\#\mathrm{CSP}(f)$. Suppose $V = \{x_1,\dots,x_n\}$ and label each constraint $((x_i,x_j),f)\in C$ by the tuple (i,j) as a shorthand. This fully specifies the constraint as there is only one constraint function. Then

$$Z(\Omega) = \sum_{\mathbf{x}\in\{0,1\}^n} \prod_{(i,j)\in C} f(x_i,x_j).$$

Because of the sum over all $\mathbf{x}$, the following holds:

$$\begin{aligned}\sum_{\mathbf{x}\in\{0,1\}^n} \prod_{(i,j)\in C} f(x_i,x_j) &= \frac{1}{2}\left(\sum_{\mathbf{x}\in\{0,1\}^n} \prod_{(i,j)\in C} f(x_i,x_j) + \sum_{\mathbf{x}\in\{0,1\}^n} \prod_{(i,j)\in C} f(x_i\oplus 1, x_j\oplus 1)\right)\\ &= \frac{1}{2}\sum_{y\in\{0,1\}}\sum_{\mathbf{x}\in\{0,1\}^n} \prod_{(i,j)\in C} f(x_i\oplus y, x_j\oplus y).\end{aligned}$$

Now, the latter formula is $\frac{1}{2}$ times the partition function for an instance of $\#\mathrm{CSP}(f')$ with set of variables $V' = \{x_1,\dots,x_n,y\}$ and set of constraints $C' = \{((x_i,x_j,y),f') \mid (i,j)\in C\}$. Thus we have $\#\mathrm{CSP}(f)\le_{AP}\#\mathrm{CSP}(f')$. □

Lemma 46. *For any finite subset $\mathcal{F}\subseteq\mathsf{SDP}_3$ we have $\#\mathrm{CSP}(\mathcal{F})\le_{AP}\mathsf{Holant}(\widehat{\mathcal{F}},\oplus_3)$, where $\widehat{\mathcal{F}} = \{\widehat{f} : f\in\mathcal{F}\}$.*

Proof. We will argue

$$\#\mathrm{CSP}(\mathcal{F})\le_{AP}\mathsf{Holant}(\mathcal{F},\mathrm{EQ}_3)\le_{AP}\mathsf{Holant}(\widehat{\mathcal{F}},\oplus_3).$$

The first reduction is exactly a transformation to the holant framework; see Section 2.4 or [4, Proposition 1].

For the second reduction, let

$$M = \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}.$$

Note that $MM^T = 2I$. The entry of the k-fold tensor product $M\otimes\dots\otimes M$ corresponding to row $(x_1,\dots,x_k)$ and column $(p_1,\dots,p_k)$ is $(-1)^{p_1x_1+\dots+p_kx_k}$. Thus, for any k-ary function h,

$$(M\circ h)(x_1,\dots,x_k) = \sum_{p_1,\dots,p_k\in\{0,1\}} (-1)^{p_1x_1+\dots+p_kx_k} h(p_1,\dots,p_k) = 2^k\widehat{h}(x_1,\dots,x_k).$$

Hence if f has arity 3, then $M \circ f = 2^3\widehat{f}$. In particular, $M \circ \mathrm{EQ}_3 = 2^3\widehat{\mathrm{EQ}_3} = 2{\oplus_3}$ by Observations 6 and 43.

Now by a corollary of Valiant's Holant Theorem, given here as Theorem 27, and the above connection between holographic transformations under M and the Fourier transform, we have

$$\mathsf{Holant}(\mathcal{F}, \mathrm{EQ}_3) \equiv_{AP} \mathsf{Holant}(M \circ \mathcal{F}, M \circ \mathrm{EQ}_3) \equiv_{AP} \mathsf{Holant}\left(\mathcal{F}', 2{\oplus_3}\right),$$

where $\mathcal{F}' = \{2^3\widehat{f} : f \in \mathcal{F}\}$. Recall that if a constraint function is multiplied by a constant, this factor can be absorbed into an AP-reduction: hence $\mathsf{Holant}\left(\mathcal{F}', 2{\oplus_3}\right) \equiv_{AP} \mathsf{Holant}(\widehat{\mathcal{F}}, {\oplus_3})$. This completes the chain of reductions. □

Recall from Section 2.4 that an assignment σ of a holant instance Ω with variables V and constraints C is a function $\sigma\colon V \to \{0,1\}$ choosing a spin from $\{0,1\}$ for each variable in V. The assignment has a weight w_σ.

Definition 47. A *near-assignment* of a holant instance $\Omega = (V, C)$ is defined as follows: take two distinct variables $u, v \in V$. Replace the two occurrences of u in constraints by two new variables u', u'' and add a new constraint $((u', u''), \mathrm{NEQ})$. Similarly, replace the two occurrences of v with new variables v', v'' and add a new constraint $((v', v''), \mathrm{NEQ})$. This gives a new holant instance $\Omega_{u,v}$. Any assignment σ of $\Omega_{u,v}$ with $\sigma(u') \neq \sigma(u'')$ and $\sigma(v') \neq \sigma(v'')$ is called a near-assignment of Ω.

Recall that $Z(\Omega)$ is the total weight of all assignments of a holant instance Ω (see (2)). Define

$$Z'(\Omega) := \sum_{\{u,v\} \subseteq V} Z(\Omega_{u,v}) \tag{9}$$

to be the total weight of all near-assignments.

Note that assignments of $\Omega_{u,v}$ which are not near-assignments of Ω according to Definition 47 do not satisfy both disequality constraints. The contribution of these assignments to the partition function $Z(\Omega_{u,v})$ is thus 0; hence $Z'(\Omega)$ is indeed the total weight of all near-assignments.

Lemma 48. *For any n-variable holant instance Ω which uses only functions in $\mathcal{P}$, we have*

$$Z'(\Omega) \leq 2n^2 Z(\Omega).$$

Proof. Let $\Omega = (V, C)$ and note that $|V| = n$. Recall the definition of pps-formula (Definition 12). Like any CSP instance, Ω can be viewed as a pps-formula ψ with no free variables. The atomic formulas of ψ correspond to the constraints in C.

Given any pair of distinct variables $u, v \in V$, let $\psi_{u,v}$ be the pps-formula obtained from ψ by removing the (bound) variables u and v and introducing four new free variables, w_1, w_2, w_3 and w_4. The two occurrences of u in atomic formulas are replaced with w_1 and w_2. The two occurrences of v in atomic formulas are replaced with w_3 and w_4.

By construction, the function $f_{\psi_{u,v}}$ represented by the pps-formula $\psi_{u,v}$ (see the text below Definition 12) is in $\langle\mathcal{P}\rangle$. But $\mathcal{P}$ is a functional clone (see Lemma 22), so $f_{\psi_{u,v}}(x_1, x_2, x_3, x_4) \in \mathcal{P}$.

The construction of $\psi_{u,v}$ guarantees that for any pair of distinct vertices u, v, we have

$$Z(\Omega) = \sum_{x,y \in \{0,1\}} f_{\psi_{u,v}}(x, x, y, y). \tag{10}$$

Furthermore,

$$\sum_{x,y \in \{0,1\}} f_{\psi_{u,v}}(x, 1-x, y, 1-y)$$

is exactly the value $Z(\Omega_{u,v})$ arising in Definition 47. Thus, by (9), the total weight of near-assignments of Ω is

$$\begin{aligned} Z'(\Omega) &= \sum_{\{u,v\}\subseteq V} \sum_{x,y\in\{0,1\}} f_{\psi_{u,v}}(x, 1-x, y, 1-y) \\ &\leq \sum_{\{u,v\}\subseteq V} \sum_{x,y\in\{0,1\}} f_{\psi_{u,v}}(0,0,0,0) \\ &\leq \binom{n}{2} \cdot 4 \cdot Z(\Omega) \leq 2n^2 Z(\Omega). \end{aligned}$$

Here, the first inequality uses Lemma 9 and the second inequality uses (10). □

An *edge-weighted multigraph* is a multigraph in which each edge e is assigned a non-negative rational weight $w(e)$. Given an edge-weighted multigraph G, a *matching* of G is a subset $M \subseteq E(G)$ such that each vertex is incident to at most one edge in M. The *weight* of M, denoted $w(M)$, is the product of the weights of the edges in M. A matching M is *perfect* if every vertex is incident to exactly one edge in M. A matching M is *nearly perfect* if every vertex is incident to exactly one edge in M, except for two vertices which are not incident to any edges in M; the concept of near-perfect matchings will be important in Lemma 54. The *total weight of perfect matchings* of the multigraph G is defined by $Z_{\mathrm{PM}}(G) = \sum_M w(M)$, where the sum is over all perfect matchings M of G. The total weight of near-perfect matchings is similarly defined as $Z_{\mathrm{NPM}}(G) = \sum_M w(M)$, where the sum is over all near-perfect matchings M of G.

Define $\mathsf{WtEven3} \subseteq \mathcal{B}_3$ to be the set of all ternary functions f with $f(0,0,0) = 1$ and $f(0,0,1) = f(0,1,0) = f(1,0,0) = f(1,1,1) = 0$.

Observation 49. The set $\mathsf{WtEven3}$ contains $\oplus_3$. Also, for every $f \in \mathsf{SDP}_3$, other than the all-zero function, $\widehat{f}(0,0,0) > 0$. Furthermore, by Lemma 42, $\widehat{f}$ is zero on inputs of odd Hamming weight. Thus $\mathsf{WtEven3}$ contains a scalar multiple of $\widehat{f}$.

An *even assignment* of a holant instance $\Omega = (V, C)$ is an assignment σ with the property that there is an even number of 1 spins in the scope of each constraint. Formally, $\sigma : V \to \{0,1\}$ is an even assignment if, for all constraints $c \in C$,

$$|\sigma(\mathbf{v}_c)| := \sum_{w\in\mathbf{v}_c} \sigma(w) \equiv 0 \pmod 2,$$

where the sum is over all variables w in the tuple $\mathbf{v}_c$ with their correct multiplicities. If all constraint functions in the holant instance are taken from $\mathsf{WtEven3}$, then only even assignments contribute to the partition function, as the contribution of each constraint is zero unless an even number of the variables in its scope are 1.

Any holant instance using only constraint functions in $\mathsf{WtEven3}$ can be transformed into the problem of counting perfect matchings on an edge-weighted graph with non-negative rational edge weights, as shown in the following definition and lemma. This result is different from known results reducing the problem of computing holant values to counting perfect matchings on edge-weighted graphs, which generally use negative edge weights, even when the constraint functions satisfy parity conditions [8, Lemma 2.25 & Theorem 2.28].

Definition 50. Suppose that $\Omega = (V, C)$ is a holant instance using only constraint functions in $\mathsf{WtEven3}$. Let $G'' = (V'', E'')$ be the edge-weighted multigraph defined as follows:

- G'' contains three distinct vertices $c^{(1)}, c^{(2)}, c^{(3)}$ for each constraint $c \in C$.
- G'' contains the following weighted edges for each $c = (\mathbf{v}_c, f_c) \in C$:
  - an edge $\{c^{(1)}, c^{(2)}\}$ with weight $f_c(1,1,0)$,

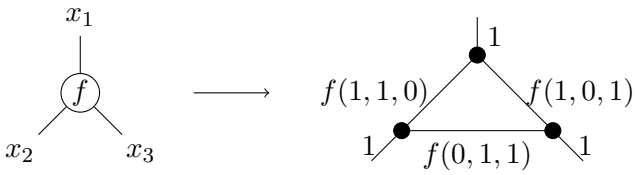

Figure 2: A constraint $((x_1, x_2, x_3), f)$ in a holant problem using only constraint functions from WtEven3 corresponds to a triangle in the edge-weighted graph constructed in Definition 50.

  - an edge $\{c^{(1)}, c^{(3)}\}$ with weight $f_c(1,0,1)$, and
  - an edge $\{c^{(2)}, c^{(3)}\}$ with weight $f_c(0,1,1)$.

  These edges are called *edges within triangles*.

- G'' furthermore contains an edge $\{c^{(i)}, d^{(j)}\}$ with weight 1 for any pair of distinct constraints $c, d \in C$ and any $i, j \in \{1,2,3\}$ such that the same variable appears in the i-th position of $\mathbf{v}_c$ and in the j-th position of $\mathbf{v}_d$. Finally, G'' contains an edge $\{c^{(i)}, c^{(j)}\}$ with weight 1 for any $c \in C$ and any pair $i, j \in \{1,2,3\}$ with $i < j$ such that the same variable appears in both the i-th and j-th position of $\mathbf{v}_c$. These edges are called *edges between triangles*.

Note that G'' may not be simple. For example, if some variable appears in position 1 and position 3 of a constraint $c \in C$ then G'' has the edge $\{c^{(1)}, c^{(3)}\}$ with weight $f_c(1,0,1)$ but also the edge $\{c^{(1)}, c^{(3)}\}$ with weight 1.

The "triangle" terminology is used because the graph G'' can be constructed by taking the graph representation G' of Ω as defined in Observation 28 and then expanding each vertex of G' into a triangle, as shown in Figure 2.

The following lemma relates the total weight of perfect matchings $Z_{\text{PM}}(G'')$ to the partition function $Z(\Omega)$ of the corresponding holant instance.

Lemma 51. *Given any n-variable holant instance Ω using functions in* WtEven3*, let G'' be the corresponding edge-weighted multigraph according to Definition 50. Then $Z_{PM}(G'') = Z(\Omega)$.*

Proof. Let $\Omega = (V, C)$ and $G'' = (V'', E'')$. Note that all assignments contributing a non-zero weight to the partition function $Z(\Omega)$ are even since the constraint functions have support only on inputs of even Hamming weight.

We define a weight-preserving bijection between the sets

$$\mathcal{E} = \{\sigma : \sigma \text{ is an even assignment of } \Omega\} \quad \text{and}$$
$$\mathcal{M} = \{M : M \text{ is a perfect matching of } G''\}.$$

Consider $\sigma \in \mathcal{E}$. For every variable $v \in V$, there is exactly one between-triangles edge $e_v = \{c^{(i)}, d^{(j)}\} \in E''$ where $c, d \in C$ such that v appears in the i-th position of $\mathbf{v}_c$ and in the j-th position of $\mathbf{v}_d$. Let $S_\sigma = \{e_v : \sigma(v) = 0\}$. This set is a matching because each vertex in V'' is incident on exactly one between-triangles edge, and S_σ is a subset of the edges between triangles. Let T_σ be the following subset of within-triangle edges:

$$T_\sigma = \left\{\{c^{(i)}, c^{(j)}\} : c \in C \text{ with } \sigma(\mathbf{v}_c[i]) = \sigma(\mathbf{v}_c[j]) = 1 \text{ and } 1 \le i < j \le 3\right\}.$$

A portion of the set $M_\sigma = S_\sigma \cup T_\sigma$ is illustrated in Figure 3. We will show that M_σ is a perfect matching. To see this, consider some vertex $c^{(i)} \in V''$, which corresponds to a variable $\mathbf{v}_c[i]$. If $\sigma(\mathbf{v}_c[i]) = 0$ then $c^{(i)}$ is matched by an edge in S_σ, this edge is unique in S_σ as there is only one edge between triangles incident on any given vertex. Furthermore, $c^{(i)}$ cannot appear

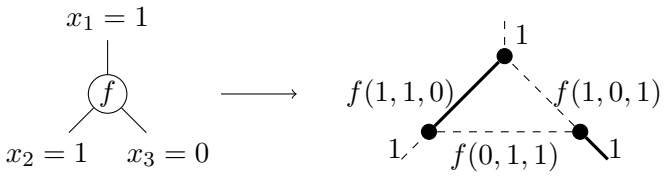

Figure 3: Suppose that the assignment σ maps $(x_1, x_2, x_3) \mapsto (1, 1, 0)$. The thickened edge labelled "1" is in S_σ and the other thickened edge is in T_σ. The dashed edges are not in M_σ.

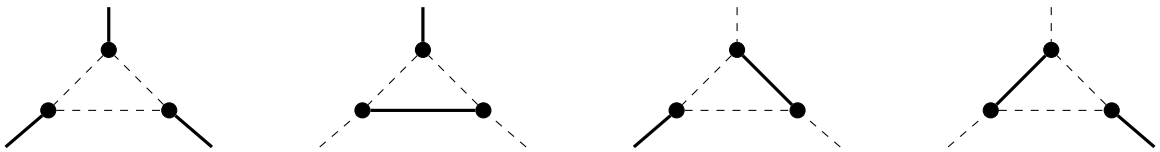

Figure 4: The different matchings for a triangle, where thick lines denote edges in the matching and dashed lines denote edges not in the matching.

in any edges in T_σ. If $\sigma(\mathbf{v}_c[i]) = 1$ then $c^{(i)}$ cannot appear in any edges in S_σ. Yet since σ is even, there must be a unique $j \in \{1, 2, 3\} \setminus \{i\}$ such that $\sigma(\mathbf{v}_c[j]) = 1$, and thus a unique edge in T_σ which is incident on $c^{(i)}$. Thus M_σ is indeed a perfect matching.

Conversely, given $M \in \mathcal{M}$, define an assignment of Ω based on whether e_v, the unique between-triangles edge corresponding to variable v, is in M:

$$\sigma_M(v) = \begin{cases} 0 & \text{if } e_v \in M, \\ 1 & \text{otherwise.} \end{cases}$$

This definition fully specifies σ_M and each assignment specified in this way is even. The latter property arises because for M to be a perfect matching each triangle must satisfy one of the following two properties, cf. Figure 4:

- either, none of the within-triangle edges and all three adjacent between-triangle edges are in M, or
- one of the within-triangle edges and one of the adjacent between-triangle edges are in M.

A matching can contain at most one edge of a triangle, so this covers all cases. Thus, $\sigma_M \in \mathcal{E}$.

It is straightforward to see that the maps $\sigma \mapsto M_\sigma$ and $M \mapsto \sigma_M$ are inverse to each other. Hence they define a bijection between $\mathcal{E}$ and $\mathcal{M}$. It remains to show this bijection is weight-preserving.

Consider $\sigma \in \mathcal{E}$, then $w(\sigma) = \prod_{c \in C} f_c(\sigma(\mathbf{v}_c))$. Recall that all edges between triangles have weight 1, so

$$w(M_\sigma) = \prod_{e \in M_\sigma} w(e) = \prod_{e \in T_\sigma} w(e).$$

Observe from Definition 50 that for any $\{c^{(i)}, c^{(j)}\} \in T_\sigma$, $w(\{c^{(i)}, c^{(j)}\}) = f_c(\sigma(\mathbf{v}_c)$. Let $C' = \{c \in C : |\sigma(\mathbf{v}_c)| = 2\}$, where $|\cdot|$ denotes the Hamming weight, then $w(M) = \prod_{c \in C'} f_c(\sigma(\mathbf{v}_c))$. But σ is an even assignment and $f_c(0, 0, 0) = 1$ for all c. Thus $w(\sigma) = w(M_\sigma)$. □

Lemma 52. *Given any n-variable holant instance Ω using functions in* WtEven3*, let G'' be the corresponding edge-weighted multigraph according to Definition 50. Then $Z_{NPM}(G'') \leq nZ(\Omega) + Z'(\Omega)$.*

Proof. Let $\Omega = (V, C)$ where $n = |V|$. We now define three sets. Let

$$\begin{aligned}
\mathcal{M} &= \{M \mid M \text{ is a near-perfect matching of } G'' \text{ with } w(M) > 0\}.\\
A &= \{(v, \sigma) \mid v \in V \text{ and } \sigma \text{ is an even assignment of } \Omega \text{ with } \sigma(v) = 0\}, \text{and}\\
B &= \{(u, v, \sigma) \mid u \in V \text{ and } v \in V \text{ are distinct and } \sigma \text{ is an assignment of } \Omega_{u,v} \text{ that is}\\
&\qquad \text{even at all constraints except } ((u', u''), \text{NEQ}) \text{ and } ((v', v''), \text{NEQ})\}.
\end{aligned}$$

Since the only assignments of Ω that contribute to $Z(\Omega)$ are even assignments, we have $\sum_{(v,\sigma)\in A} w_\sigma \leq nZ(\Omega)$. Similarly, since the only assignments of $\Omega_{u,v}$ that contribute to $Z(\Omega_{u,v})$ are even at all ternary constraints, $\sum_{(u,v,\sigma)\in B} w_\sigma = Z'(\Omega)$. We will define a weight-preserving injection τ from $\mathcal{M}$ into $A \cup B$.

Consider $M \in \mathcal{M}$. Every vertex of G'' is incident to exactly one edge in M except for two vertices, which are not incident to any edges in M. Say that these two vertices are $c_1^{(i_1)}$ and $c_2^{(i_2)}$ for some $c_1, c_2 \in C$ and $i_1, i_2 \in \{1, 2, 3\}$. Note that c_1 might coincide with c_2 and i_1 might coincide with i_2, but $c_1^{(i_1)} \neq c_2^{(i_2)}$.

Now, for every variable $v \in V$ there is exactly one between-triangle edge, say $\{c_3^{(i_3)}, c_4^{(i_4)}\}$, such that v occurs in position position i_3 of c_3 and position i_4 of c_4. The idea will be to define $\sigma_M(v) \in \{0, 1\}$ based on whether this edge is in M.

First, if $\{c_3^{(i_3)}, c_4^{(i_4)}\}$ is disjoint from $\{c_1^{(i_1)}, c_2^{(i_2)}\}$, we make the following definition, which is similar to the construction in the proof of Lemma 51:

$$\sigma_M(v) = \begin{cases} 0, & \text{if } \{c_3^{(i_3)}, c_4^{(i_4)}\} \in M, \\ 1, & \text{otherwise.} \end{cases} \tag{11}$$

Now recall that $c_1^{(i_1)}$ and $c_2^{(i_2)}$ are not matched in M. There are two cases to consider, depending on whether $\{c_1^{(i_1)}, c_2^{(i_2)}\}$ is an edge of G'' or not. In each case, we define $\tau(M)$ and argue that it is weight-preserving. Later, we will argue that τ is an injection.

Case 1. Suppose that $\{c_1^{(i_1)}, c_2^{(i_2)}\}$ is an edge of G''. In this case, there is exactly one variable v for which $\sigma_M(v)$ is not defined by (11) and this variable v is in position i_1 of c_1 and position i_2 of c_2. So define $\sigma_M(v) = 0$ and define $\tau(M) = (v, \sigma_M)$. The fact that $w(M) = w_{\sigma_M}$ is similar to the argument that we already made in the proof of Lemma 51. Let M' be the perfect matching of G'' consisting of M and the edge $\{c_1^{(i_1)}, c_2^{(i_2)}\}$. Since $\{c_1^{(i_1)}, c_2^{(i_2)}\}$ is a between-triangle edge (with weight 1), we have $w(M') = w(M)$. Since $\sigma_M(v) = 0$, the assignment σ_M coincides with the assignment $\sigma_{M'}$ constructed in the proof of Lemma 51 so $w_{\sigma_M} = w_{\sigma_{M'}}$. But we have already argued, in the proof of Lemma 51 that $\sigma_{M'} = w(M')$. So we have proved that $w(M) = w_{\sigma_M}$. Since $w(M) > 0$, this ensures that σ_M is an even assignment, so $\tau(M) \in A$.

Case 2. Suppose that $\{c_1^{(i_1)}, c_2^{(i_2)}\}$ is not an edge of G''. Then there are exactly two variables u_1 and u_2 for which $\sigma_M(u_1)$ and $\sigma_M(u_2)$ are not defined by (11). Variable u_1 appears in position i_1 of c_1 (and somewhere else). Variable u_2 appears in position i_2 of c_2 (and somewhere else). Consider the instance Ω_{u_1,u_2} constructed as in Definition 47. Suppose that the variables replacing u_1 in Ω_{u_1,u_2} are u_1' and u_1'' with u_1' appearing in $c_1^{(i_1)}$ and that the variables replacing u_2 in Ω_{u_1,u_2} are u_2' and u_2'' with u_2' appearing in $c_2^{(i_2)}$. Then set $\sigma_M(u_1') = \sigma_M(u_2') = 0$ and $\sigma_M(u_1'') = \sigma_M(u_2'') = 1$. This gives an assignment for Ω_{u_1,u_2} which satisfies the disequality constraints and which has an even number of spin-1 variables in each ternary constraint. So define $\tau(M) = (u_1, u_2, \sigma_M)$ and note that $\tau(M) \in B$ and (using the same arguments as in the proof of Lemma 51) that $w(M) = \sigma_M$.

To conclude the proof, we must argue that τ is an injection. This is straightforward. From $(v, \sigma) \in A$ there is a unique edge $\{c_1^{(i_1)}, c_2^{(i_2)}\}$ of G'' corresponding to v. Leave this out of M and recover the rest of the intersection of M and the between-triangle edges of G'' using

Equation (11). There is a unique extension to the edges within triangles which gives a near-perfect matching of G'' where $c_1^{(i_1)}$ and $c_2^{(i_2)}$ are unmatched. Similarly, given $(u_1, u_2, \sigma) \in B$ it is easy to identify the between-triangle edges of G'' corresponding to u_1 and u_2. Leave u_1 and u_2 unmatched in M and recover the rest of the between-triangle edges of M from (11). Again, there is a unique extension to edges within triangles. □

Observation 53. Let $G = (V, E)$ be an edge-weighted multigraph in which each edge $e \in E$ has a non-negative rational weight $w(e)$. Let d be the least common denominator of the positive edge weights of G. Let $G' = (V, E')$ be the *unweighted* multigraph defined as follows. For each $\{u, v\} \subseteq V$, let $\{e_1, \ldots, e_{k_{u,v}}\}$ be the set of edges from u to v in E. The number of edges $\{u, v\}$ in E' is defined to be $d\sum_{j=1}^{k_{u,v}} w(e_j)$. Similarly, for each $v \in V$, let $\{e_1, \ldots, e_{k_v}\}$ be the set of self-loops on v in E. The number of self-loops on v in E' is defined to be $d\sum_{j=1}^{k_v} w(e_j)$. Then $Z_{\mathrm{PM}}(G') = d^{|V|/2} Z_{\mathrm{PM}}(G)$ and $Z_{\mathrm{NPM}}(G') = d^{|V|/2-1} Z_{\mathrm{NPM}}(G)$

Proof. Let $n = |V|$. Let H be an edge-weighted multigraph with the same vertices and edges as G except that the weight of each edge e is $dw(e)$. Since each perfect matching of G has $n/2$ edges and each near-perfect matching has $n/2 - 1$ edges, $Z_{\mathrm{PM}}(H) = d^{n/2} Z_{\mathrm{PM}}(G)$ and $Z_{\mathrm{NPM}}(H) = d^{n/2-1} Z_{\mathrm{NPM}}(G)$. Note that all of the edge-weights of H are non-negative integers. Then it is easy to see that $Z_{\mathrm{PM}}(G') = Z_{\mathrm{PM}}(H)$ and $Z_{\mathrm{NPM}}(G') = Z_{\mathrm{NPM}}(H)$. □

Observation 53 allows us to use the following result, where $M_n(G)$ denotes the number of perfect matchings of a $2n$-vertex multigraph G, and $M_{n-1}(G)$ is the number of near-perfect matchings of G. While the result was originally stated for graphs, its proof also applies to multigraphs.

Lemma 54 ([17, Corollary 3.7])**.** *Let q be any fixed polynomial. There exists an FPRAS for $|M_n(G)|$ when the input is restricted to be a $2n$-vertex multigraph G satisfying $|M_{n-1}(G)| \leq q(n)\,|M_n(G)|$.*

Theorem 55. *Suppose that $\mathcal{F}$ is a finite subset of SDP_3. Then $\#\mathrm{CSP}(\mathcal{F})$ has an FPRAS.*

Proof. If $\mathcal{F}$ contains the all-zero function f_0, then any instance Ω of $\#\mathrm{CSP}(\mathcal{F})$ which contains a constraint using f_0 satisfies $Z(\Omega) = 0$. The property of whether Ω contains a constraint using f_0 can be checked in polynomial time. Thus, $\#\mathrm{CSP}(\mathcal{F}) \leq_{AP} \#\mathrm{CSP}(\mathcal{F} \setminus \{f_0\})$. In other words, it suffices to consider sets $\mathcal{F}$ that do not contain the all-zero function.

By Lemma 46, we have $\#\mathrm{CSP}(\mathcal{F}) \leq_{AP} \mathsf{Holant}(\widehat{\mathcal{F}}, \oplus_3)$, where $\widehat{\mathcal{F}} = \{\widehat{f} : f \in \mathcal{F}\}$. For any $f \in \mathcal{F}$, since f is not the all-zero function and all values are nonnegative,

$$c_f := \widehat{f}(0,0,0) = \frac{1}{8} \sum_{x,y,z \in \{0,1\}} f(x,y,z) > 0.$$

Let $f''(x,y,z) = c_f^{-1} \widehat{f}(x,y,z)$ be a normalised version of the Fourier transform of f, and let $\mathcal{F}'' = \{f'' : f \in \mathcal{F}\}$. Then $\mathsf{Holant}(\widehat{\mathcal{F}}, \oplus_3) =_{AP} \mathsf{Holant}(\mathcal{F}'', \oplus_3)$. Now, $\mathcal{F} \subseteq \mathsf{SDP}_3$, so by Observation 49, $\mathcal{F}'' \cup \{\oplus_3\} \subseteq \mathsf{WtEven3}$. Hence, given any instance Ω of the problem $\mathsf{Holant}(\mathcal{F}'', \oplus_3)$ we can use Definition 50 to construct an edge-weighted multigraph G'' so that, by Lemma 51, $Z(\Omega) = Z_{\mathrm{PM}}(G'')$. Let n be the number of variables of Ω. To avoid trivialities, we assume $n \geq 1$.

By Lemma 52, $Z_{\mathrm{NPM}}(G'') \leq nZ(\Omega) + Z'(\Omega)$ and by Lemma 48, $Z'(\Omega) \leq 2n^2 Z(\Omega)$ so $Z_{\mathrm{NPM}}(G'') \leq 3n^2 Z_{\mathrm{PM}}(G'')$.

Using Observation 53, it is easy to define an unweighted multigraph G' such that $Z_{\mathrm{PM}}(G') = d^{|V(G'')|/2} Z_{\mathrm{PM}}(G'')$ and $Z_{\mathrm{NPM}}(G') = d^{|V(G'')|/2-1} Z_{\mathrm{NPM}}(G'')$, where d is the least common denominator of the positive edge weights of G''.

The FPRAS for computing $Z_{\mathrm{PM}}(\cdot)$ from Lemma 54 can be used to approximate $Z_{\mathrm{PM}}(G')$ which gives an approximation to $Z_{\mathrm{PM}}(G'') = Z(\Omega)$.

As $\#\mathrm{CSP}(\mathcal{F}) \leq_{AP} \mathsf{Holant}(\mathcal{F}'', \oplus_3)$, this implies the existence of an FPRAS for $\#\mathrm{CSP}(\mathcal{F})$. □

Corollary 56. *For all functions $f \in \mathcal{P} \cap \mathcal{B}_2$, the problem $\#\mathrm{CSP}(f)$ has an FPRAS.*

Proof. Let $f'(x, y, z) := f(x \oplus z, y \oplus z)$, then $\#\mathrm{CSP}(f) \leq_{AP} \#\mathrm{CSP}(f')$ by Lemma 45. Furthermore, $f' \in \mathsf{SDP}_3$ by Lemma 44, so $\#\mathrm{CSP}(f')$ has an FPRAS by Theorem 55. Therefore, there exists an FPRAS for $\#\mathrm{CSP}(f)$. □

4.4 Non-monotone nontrivial non-lsm functions

The remaining case in the statement of Theorem 2 is that of a nontrivial non-lsm non-monotone function.

Lemma 57. *If $f \in \mathcal{B}_2$ is nontrivial and non-lsm, and both f and $\bar{f}$ are non-monotone, then $\#\mathrm{CSP}(f)$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.*

Proof. Suppose f is nontrivial and non-lsm, then $ad < bc$ by Observation 3 and a, d are not both zero.

If $\widehat{f}_{01}\widehat{f}_{10} < 0$, then both $\langle f \rangle \cap \mathcal{B}_1^{<}$ and $\langle f \rangle \cap \mathcal{B}_1^{>}$ are non-empty by Lemma 40. Hardness then follows by Theorem 1 and Lemma 25, noting that the reduction used in the latter result is actually a simulation (cf. the proof of Lemma 39).

Hence, suppose instead that $\widehat{f}_{01}\widehat{f}_{10} \geq 0$. Without loss of generality, assume both Fourier coefficients are nonpositive, i.e.

$$a + c \leq b + d, \tag{12}$$

$$a + b \leq c + d. \tag{13}$$

(If instead both Fourier coefficients are nonnegative, replace f with $f' := \bar{f}$, then $\#\mathrm{CSP}(f') =_{AP} \#\mathrm{CSP}(f)$ by Observation 23 and $\widehat{f'}_{01}, \widehat{f'}_{10}$ are both nonpositive by Observation 7. Furthermore, the assumptions of the lemma remain unchanged since both f and $\bar{f}$ are non-monotone and the properties of being nontrivial and non-lsm are invariant under bit-flips.)

Adding (12) and (13) gives

$$a \leq d. \tag{14}$$

The function f is monotone if $a \leq b \leq d$ and $a \leq c \leq d$. Without loss of generality, it suffices to consider two non-monotone cases: $b < a$ or $d < b$; if only c fails to satisfy monotonicity, replace $f(x, y)$ by $f(y, x)$ and proceed as before. By Observation 8, swapping the variables swaps the Fourier coefficients $\widehat{f}_{01}$ and $\widehat{f}_{10}$, so the assumption that both are nonpositive remains valid.

If $b < a$, then (12) implies that $c < d$. But multiplying these two inequalities yields $bc < ad$, contradicting the assumption that f is non-lsm.

Thus the only case to consider is

$$d < b, \tag{15}$$

which together with (13) implies

$$a < c. \tag{16}$$

We distinguish two subcases.

1. Suppose that equality holds in both (12) and (13). Then, by adding or subtracting the two equations, we find $a = d$ and $b = c$. Since f is nontrivial, $b \neq 0$, so let $f' := \frac{1}{b} \cdot f$. As constant factors can be absorbed into AP-reductions, we have $\#\mathrm{CSP}(f) =_{AP} \#\mathrm{CSP}(f')$. Now, f' is an antiferromagnetic Ising function and, by [16, Theorem 3], $\#\mathrm{CSP}(f')$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.

2. We may now assume that at least one of (12) and (13) is a strict inequality. If (12) is strict, let $\mathsf{up}(x) := \sum_y f(y,x)$, which satisfies $\mathsf{up}(0) = a + c < b + d = \mathsf{up}(1)$. Otherwise, (13) is strict, so $\mathsf{up}(x) := \sum_y f(x,y)$ satisfies $\mathsf{up}(0) = a + b < c + d = \mathsf{up}(1)$. In either case, up is a strictly increasing unary function in $\langle f \rangle$. It is also permissive as (14), (15) and (16), together with nonnegativity of all values, imply that $(a+b)$ and $(a+c)$ are both strictly positive. By Lemma 25, $\#\mathrm{CSP}(f, \mathsf{up}) \leq_{AP} \#\mathrm{CSP}(f)$.

   Now, according to Lemma 24, there exists $\mathsf{up}' \in \mathcal{B}_1^{<,\mathrm{n}}$ such that $\#\mathrm{CSP}(f, \mathsf{up}') \leq_{AP} \#\mathrm{CSP}(f, \mathsf{up}) \leq_{AP} \#\mathrm{CSP}(f)$. It thus suffices to show that $\#\mathrm{CSP}(f, \mathsf{up}')$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.

   Following from Observation 19, $\delta_1 \in \langle f, \mathsf{up}' \rangle_{\omega,p}$. We can therefore pin inputs to the value 1. In particular, $\mathsf{down}(x) := f(x,1)$ is a unary function in $\langle f, \mathsf{up}' \rangle_{\omega,p}$; it is strictly decreasing as $f(0,1) = b > d = f(1,1)$ by (15). It remains to check that down is permissive. To see this, note that f is nontrivial and $0 \leq a \leq d$. Together, these two properties imply that $d > 0$, because $0 = a = d$ would make f trivial. So down is indeed permissive. Using Lemma 25 again, we find that $\#\mathrm{CSP}(f, \mathsf{up}', \mathsf{down}) \leq_{AP} \#\mathrm{CSP}(f)$. But, by Theorem 1, $\#\mathrm{CSP}(f, \mathsf{up}', \mathsf{down})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$; the same condition thus applies to $\#\mathrm{CSP}(f)$.

By exhaustive analysis of all cases, we have therefore shown that, whenever f is a nontrivial non-lsm function and both f and $\bar{f}$ are non-monotone, $\#\mathrm{CSP}(f)$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$. □

4.5 Proof of Theorem 2

We now have all the pieces required to prove the main theorem of this section.

Theorem 2 (restated). *Let $f \in \mathcal{B}_2$.*

1. *If f is trivial, then $\#\mathrm{CSP}(\{f\})$ is in FP.*
2. *Otherwise, if f is ferromagnetic:*
   (a) *If $\widehat{f}_{01}\widehat{f}_{10} < 0$, then $\#\mathrm{CSP}(\{f\})$ is equivalent to $\#\mathsf{BIS}$ under AP-reductions.*
   (b) *Otherwise, $\#\mathrm{CSP}(\{f\})$ has an FPRAS.*
3. *Otherwise, if both $f(x,y)$ and $f(1-x,1-y)$ are non-monotone, then $\#\mathrm{CSP}(\{f\})$ does not have an FPRAS unless $\mathsf{NP} = \mathsf{RP}$.*

The only cases not covered by Theorem 2 are when either function $f(x,y)$ or function $f(1-x,1-y)$ is monotone. We address these cases to some extent in the next section.

Proof. For Property 1, note that any trivial binary function f is in $\mathcal{N}$, and $\#\mathrm{CSP}(f)$ has an FPRAS by Theorem 30.

If f is a nontrivial binary lsm function whose Fourier coefficients $\widehat{f}_{01}$ and $\widehat{f}_{10}$ have opposite signs, then there exists $\mathsf{up} \in \langle f \rangle \cap \mathcal{B}_1^{<}$ and $\mathsf{down} \in \langle f \rangle \cap \mathcal{B}_1^{>}$ by Lemma 40. Thus, $\#\mathrm{CSP}(f)$ is $\#\mathsf{BIS}$-hard by Lemma 35. It is also $\#\mathsf{BIS}$-easy by [6, Theorem 47]. This establishes Property 2a.

If f is a nontrivial binary lsm function whose Fourier coefficients $\widehat{f}_{01}$ and $\widehat{f}_{10}$ are both nonnegative or both nonpositive, then by Observations 7 and 23, it suffices to consider the case $\widehat{f}_{01}, \widehat{f}_{10} \geq 0$: otherwise, replace $\#\mathrm{CSP}(f)$ with $\#\mathrm{CSP}(\bar{f})$, which is AP-interreducible with the former. Now, $\widehat{f}_{00} \geq 0$ for all $f \in \mathcal{B}_2$ by the definition of the Fourier transform, and $\widehat{f}_{11} \geq 0$ for the given f by Lemma 41, so all Fourier coefficients of f are nonnegative. The problem $\#\mathrm{CSP}(f)$ thus has an FPRAS by Corollary 56, which proves Property 2b.

If none of the above cases apply, f is nontrivial and non-lsm. Now, if both f and $\bar{f}$ are non-monotone, then, by Lemma 57, #CSP(f) does not have an FPRAS unless NP = RP. This is Property 3.

We have thus established all the desired properties. □

With the current state of knowledge, it is possible to determine the complexity of #CSP(f) in some cases beyond the ones given in the above theorem. As we noted earlier, if f is a nontrivial anti-ferromagnetic monotone *symmetric* function, the complexity of #CSP(f) can be determined, but there is no known closed form that indicates when the counting CSP has an FPRAS and when it cannot have an FPRAS unless NP = RP, cf. Theorems 31 and 32 (from [18]).

5 Permissive unaries from pinning

In this section, we consider the following question: Given a set of functions $\mathcal{F}$, when does $\langle \mathcal{F}, \delta_0, \delta_1 \rangle$ contain both a strictly increasing permissive unary function and a strictly decreasing permissive unary function? In Section 5.1 we prove Theorem 58 which shows that if $\langle \mathcal{F}, \delta_0, \delta_1 \rangle$ does not contain these, then $\mathcal{F}$ satisfies (at least) one of three specified properties. In Section 5.2 the goal is to identify a large functional clone that does not contain both a strictly increasing permissive unary function and a strictly decreasing permissive unary function. The clone MON, from [1] fits the bill, but in some sense it is a trivial solution since it does not contain both δ_0 and δ_1. Theorem 59 identifies a clone that is strictly larger than MON and contains δ_0 and δ_1 but still does not contain both a strictly increasing permissive unary function and a strictly decreasing permissive unary function.

5.1 Condition for having both kinds of unaries

The following theorem shows that, unless each element of a set of functions $\mathcal{F}$ satisfies certain monotonicity conditions on its support, the clone $\langle \mathcal{F}, \delta_0, \delta_1 \rangle$ contains both a strictly increasing and a strictly decreasing permissive unary function. Recall from Section 2 that $\bar{f}$ denotes the bit-flip of a function f, $\bar{\mathcal{F}}$ the set containing the bit-flips of all the functions in $\mathcal{F}$, and a pure function is a constant times a relation. Furthermore, recall from Section 2 that $f \in \mathcal{B}$ is monotone on its support if for any $\mathbf{a}, \mathbf{b} \in R_f$ such that $\mathbf{a} \leq \mathbf{b}$ it holds $f(\mathbf{a}) \leq f(\mathbf{b})$.

Theorem 58. *Let $\mathcal{F}$ be a set of functions in $\mathcal{B}$. Then at least one of the following is true:*

1. *Every function in $\mathcal{F}$ is pure.*
2. *Every function in $\mathcal{F}$ has the following properties*
   - *it is monotone on its support, and*
   - *its support is closed under $\vee$.*

   Furthermore, there is some function $f \in \mathcal{F}$ such that $\bar{f}$ is not monotone on its support.
3. *$\bar{\mathcal{F}}$ satisfies Property 2.*
4. *$\langle \mathcal{F}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^{\leq}$ and $\langle \mathcal{F}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^{\geq}$ are both non-empty.*

Proof. We distinguish cases according to the monotonicity properties of functions in $\mathcal{F}$.

Case 1. Suppose there exists a function $f \in \mathcal{F}$ which is not monotone on its support and a function $g \in \mathcal{F}$ such that $\bar{g}$ is not monotone on its support (these may be the same function).

As f is not monotone on its support, there must be a pair $\mathbf{a}, \mathbf{b} \in R_f$ such that $\mathbf{a} \leq \mathbf{b}$ and $f(\mathbf{a}) > f(\mathbf{b}) > 0$. By pinning in all places where $\mathbf{a}$ and $\mathbf{b}$ agree, we can obtain a function

$f' \in \langle f, \delta_0, \delta_1 \rangle$ satisfying $f'(0,\dots,0) > f'(1,\dots,1) > 0$. Then $\mathsf{down}(x) := f'(x,\dots,x) \in \langle f, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^>$. Similarly, as $\bar{g}$ is not monotone on its support, there must be a pair $\mathbf{c}, \mathbf{d} \in R_g$ such that $\mathbf{c} \leq \mathbf{d}$ and $0 < g(\mathbf{c}) < g(\mathbf{d})$. By pinning, we obtain a function $g' \in \langle g, \delta_0, \delta_1 \rangle$ satisfying $0 < g'(0,\dots,0) < g'(1,\dots,1)$, and thus a unary function $\mathsf{up}(x) := g'(x,\dots,x)$, which is in $\langle g, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^<$. Hence, $\mathcal{F}$ satisfies Property 4.

Case 2. Suppose all functions in $\mathcal{F}$ are monotone on their support and there exists a function $f \in \mathcal{F}$ such that $\bar{f}$ is not monotone on its support.

One possibility is that the supports of all functions in $\mathcal{F}$ are all closed under $\vee$. In this case, $\mathcal{F}$ satisfies Property 2. Otherwise, there exists a function $g \in \mathcal{F}$ whose support is not closed under $\vee$, that is, there are $\mathbf{a}, \mathbf{b} \in R_g$ such that $\mathbf{a} \vee \mathbf{b} \notin R_g$. Again, g may be the same as f, or they may be distinct.

Since $\bar{f}$ is not monotone on its support, there exist $\mathbf{c}, \mathbf{d} \in R_f$ such that $\mathbf{c} \leq \mathbf{d}$ and $0 < f(\mathbf{c}) < f(\mathbf{d})$. By pinning and identification of variables, we can therefore realise a permissive unary strictly increasing function $\mathsf{up} \in \langle f, \delta_0, \delta_1 \rangle$ as in Case 1.

We will now show that we can also realise a permissive unary strictly decreasing function.

By pinning in all places where $\mathbf{a}$ and $\mathbf{b}$ agree, we may assume that $\mathbf{a} \vee \mathbf{b} = (1,\dots,1)$ and $\mathbf{a} \wedge \mathbf{b} = (0,\dots,0)$. Furthermore, by identifying all pairs i, j of variables such that $\mathbf{a}[i] = \mathbf{a}[j]$ (and thus $\mathbf{b}[i] = \mathbf{b}[j]$), we obtain a function $h(x,y) \in \langle g, \delta_0, \delta_1 \rangle$ such that $h(0,1), h(1,0) \neq 0$ but $h(1,1) = 0$. Let $a = h(0,0)$, $b = h(0,1)$, $c = h(1,0)$, where $b, c > 0$ and $a \geq 0$, and define

$$h'(x,y) := h(x,y)h(y,x) = \begin{pmatrix} a^2 & bc \\ bc & 0 \end{pmatrix}.$$

Let $u_0 := \mathsf{up}(0)$ and $u_1 := \mathsf{up}(1)$; these values satisfy $0 < u_0 < u_1$. Now,

$$\sum_{y \in \{0,1\}} h'(0,y)\mathsf{up}(y) = a^2 u_0 + bcu_1 > bcu_0 = \sum_{y \in \{0,1\}} h'(1,y)\mathsf{up}(y),$$

so $\mathsf{down}(x) := \sum_y h'(x,y)\mathsf{up}(y)$ is strictly decreasing. It is also permissive since $bc \neq 0$ and $u_0 > 0$, hence $\mathcal{F}$ satisfies Property 4.

Case 3. Suppose that for all functions $g \in \mathcal{F}$, $\bar{g}$ is monotone on its support and there exists a function $f \in \mathcal{F}$ such that f is not monotone on its support.

The first step in Case 3 is to note the following equivalent formulation: All functions in $\bar{\mathcal{F}}$ are monotone on their support and there exists a function $f \in \bar{\mathcal{F}}$ such that $\bar{f}$ is not monotone on its support.

Then, we can apply the argument from Case 2 to $\bar{\mathcal{F}}$ to find that either $\mathcal{F}$ satisfies Property 3, or $\langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^<$ and $\langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^>$ are both non-empty. Note that the bit-flip operation is its own inverse and that furthermore $\bar{\delta_0} = \delta_1$ and $\bar{\delta_1} = \delta_0$. Thus, if there is a permissive strictly increasing function $\mathsf{up} \in \langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^<$, then $\overline{\mathsf{up}} \in \langle \mathcal{F}, \delta_0, \delta_1 \rangle$. But $0 < \mathsf{up}(0) < \mathsf{up}(1)$ implies $0 < \overline{\mathsf{up}}(1) < \overline{\mathsf{up}}(0)$, so $\overline{\mathsf{up}} \in \mathcal{B}_1^>$. Similarly, if $\mathsf{down} \in \langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^>$ then $\overline{\mathsf{down}} \in \langle \mathcal{F}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^<$. Hence, if $\langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^<$ and $\langle \bar{\mathcal{F}}, \delta_0, \delta_1 \rangle \cap \mathcal{B}_1^>$ are both non-empty, then $\mathcal{F}$ satisfies Property 4.

Case 4. For all functions $f \in \mathcal{F}$, both f and $\bar{f}$ are monotone on their support.

One possibility is that all functions in $\mathcal{F}$ are pure. In this case, Property 1 is satisfied. Otherwise, there exists a function $g \in \mathcal{F}$ whose range contains at least two non-zero values. In this case, we show that there are both a unary permissive strictly increasing and a unary permissive strictly decreasing function in $\langle g, \delta_0, \delta_1 \rangle$.

As g is not pure, we can find $\mathbf{a}, \mathbf{b} \in R_g$ such that $0 < g(\mathbf{a}) < g(\mathbf{b})$. These two tuples $\mathbf{a}, \mathbf{b}$ must then be incomparable, otherwise g and $\bar{g}$ could not both be monotone on their support. Pin in all places where $\mathbf{a}$ and $\mathbf{b}$ agree, then identify all pairs of inputs i, j such that $\mathbf{a}[i] = \mathbf{a}[j]$ and $\mathbf{b}[i] = \mathbf{b}[j]$ to obtain a binary function $h(x,y)$. Without loss of generality, we may suppose this function satisfies $h(0,0) = g(\mathbf{a} \wedge \mathbf{b})$, $h(0,1) = g(\mathbf{a})$, $h(1,0) = g(\mathbf{b})$, and $h(1,1) = g(\mathbf{a} \vee \mathbf{b})$ (otherwise replace $h(x,y)$ with $h(y,x)$).

Now, g and $\bar{g}$ both being monotone on their support also requires that there be no $\mathbf{c} \in R_g$ which satisfies $\mathbf{c} \leq \mathbf{a} \wedge \mathbf{b}$ or $\mathbf{c} \geq \mathbf{a} \vee \mathbf{b}$. In particular, $\mathbf{a} \wedge \mathbf{b}$ and $\mathbf{a} \vee \mathbf{b}$ cannot be in the support of g. Hence $h(0,0) = h(1,1) = 0$ and g is a weighted disequality function. We assumed $0 < g(\mathbf{a}) < g(\mathbf{b})$, which implies $0 < h(0,1) < h(1,0)$.

Let $\mathsf{down}(x) := \sum_y h(x,y)h^2(y,x)$ and $\mathsf{up}(x) := \sum_y h^2(x,y)h(y,x)$, then

$$\mathsf{down}(0) = h(0,1)h^2(1,0) = g(\mathbf{a})g^2(\mathbf{b}) > g(\mathbf{b})g^2(\mathbf{a}) = h(1,0)h^2(0,1) = \mathsf{down}(1)$$
$$\mathsf{up}(0) = h^2(0,1)h(1,0) = g^2(\mathbf{a})g(\mathbf{b}) < g^2(\mathbf{b})g(\mathbf{a}) = h^2(1,0)h(0,1) = \mathsf{up}(1).$$

Both functions are permissive and in $\langle g, \delta_0, \delta_1 \rangle$. Therefore, Property 4 holds.

This concludes the analysis of all cases. □

Remark. Properties 1, 2, and 3 of Theorem 58 are disjoint.

Properties 2 and 3 being disjoint is immediate. For Properties 1 and 2, note that if f is a pure function, then both f and $\bar{f}$ are monotone on their support. Thus, Property 1 is disjoint from Property 2. If all functions in $\mathcal{F}$ are pure, then all functions in $\bar{\mathcal{F}}$ are pure, so an analogous argument applies for Property 1 and Property 3.

Remark. If $\mathcal{F}$ satisfies Property 1 of Theorem 58, then the complexity of $\#\mathrm{CSP}(\mathcal{F}, \delta_0, \delta_1)$ can be determined via the approximation trichotomy for Boolean #CSP in Theorem 29 from [12]. If $\mathcal{F}$ satisfies Property 4, then the complexity of $\#\mathrm{CSP}(\mathcal{F}, \delta_0, \delta_1)$ is determined by Lemma 25 and Theorem 1. So if the complexity of $\#\mathrm{CSP}(\mathcal{F}, \delta_0, \delta_1)$ is unresolved (up to the granularity of Theorem 1) then $\mathcal{F}$ or $\bar{\mathcal{F}}$ satisfies Property 2.

5.2 Condition for the absence of permissive strictly decreasing functions

In this section the goal is to identify a large functional clone that does not contain both types of permissive unary functions (the two types being strictly increasing and strictly decreasing).

We start by defining the clone MON, using definitions from [1, Section 10]. Given a k-ary function f, let $\sim_f$ be the equivalence relation on $[k]$ given by $i \sim_f j$ if, for every $\mathbf{a} \in \{0,1\}^k$, $f(\mathbf{a}) \neq 0$ implies $\mathbf{a}[i] = \mathbf{a}[j]$. If $\sim_f$ is the equality relation, f is said to be *irredundant*. By identifying variables in the equivalence classes of $\sim_f$, any function f can be transformed into an irredundant function $f^\dagger$. Note that any subset of variables of a function f can be identified using the closure operations of a functional clone (cf. Lemma 14), whether or not they are in the same equivalence class.

Recall from Section 2 that a function $g \in \mathcal{B}_k$ is *monotone* if for any $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ with $\mathbf{a} \leq \mathbf{b}$ we have $g(\mathbf{a}) \leq g(\mathbf{b})$. The function f is in MON if $f^\dagger$ is monotone. Bulatov et al. [1, Theorem 62] have shown that MON is a functional clone.[2] It does not contain a strictly decreasing permissive unary function, so in some sense it is a solution to the problem stated in the previous paragraph. However, it also does not contain δ_0, so, in some sense, it is a trivial solution.

Theorem 59 below identifies a clone that is strictly larger than MON and contains δ_0 and δ_1 but still does not contain both types of permissive unary functions.

Theorem 59. *There is a functional clone* Pin-MON *containing* δ_0 *and* δ_1 *such that* MON *is a strict subset of* Pin-MON *and* Pin-MON *contains no strictly decreasing permissive unary functions.*

In order to define Pin-MON, we use the following definition.

Definition 60. A function $f \in \mathcal{B}$ is *pin-monotone* if the following condition holds for $f^\dagger$. Let $k = \mathrm{arity}(f^\dagger)$ and suppose that $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ satisfy

- $\mathbf{a}[j] = 0$ and $\mathbf{b}[j] = 1$ for some $j \in [k]$,

[2] Bulatov et al. [1] define MON in a slightly different but equivalent way.

- $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k] \setminus \{j\}$, and
- $f^\dagger(\mathbf{a}) > f^\dagger(\mathbf{b})$,

then

1. $f^\dagger(\mathbf{b}) = 0$, and furthermore
2. $f^\dagger(\mathbf{c}) = 0$ for every $\mathbf{c} \in \{0,1\}^k$ such that $\mathbf{c}[j] = 1$.

Let Pin-MON be the set of all pin-monotone functions.

In other words, f is pin-monotone if, for every pair of bit strings $\mathbf{a}$ and $\mathbf{b}$ that differ only in one bit, with $\mathbf{a} \leq \mathbf{b}$ and $f^\dagger(\mathbf{a}) > f^\dagger(\mathbf{b})$, this implies that $f^\dagger(\mathbf{b}) = 0$ and furthermore implies that $f^\dagger(\mathbf{c}) = 0$ for all input bit strings $\mathbf{c}$ that agree with $\mathbf{b}$ on the bit where it differs from $\mathbf{a}$.

Lemma 61. *The pinning functions δ_0 and δ_1 are pin-monotone.*

Proof. Since they have arity 1, both δ_0 and δ_1 are irredundant. Then, δ_1 is pin-monotone because for any $\mathbf{a}, \mathbf{b} \in \{0,1\}$, $\mathbf{a} \leq \mathbf{b}$ implies $\delta_1(\mathbf{a}) \leq \delta_1(\mathbf{b})$.

For δ_0, there are bit strings $\mathbf{a}, \mathbf{b}$ satisfying the three conditions of Definition 60, namely $\mathbf{a} = 0$ and $\mathbf{b} = 1$. But then the two implications required by that definition are trivially satisfied, since $\delta_0(1) = 0$. Thus, δ_0 is also pin-monotone. □

Lemma 62. MON $\subsetneq$ Pin-MON.

Proof. To show MON $\subseteq$ Pin-MON, consider $f \in$ MON and let $k = \text{arity}(f^\dagger)$. By the definition of MON, $f^\dagger$ is monotone. We will show that $f^\dagger$ is pin-monotone, which implies that f is also pin-monotone.

To this end, suppose that there are $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ such that $\mathbf{a}[j] = 0$ and $\mathbf{b}[j] = 1$ for some $j \in [k]$, and $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k] \setminus \{j\}$. Then by monotonicity of $f^\dagger$, $f^\dagger(\mathbf{a}) \leq f^\dagger(\mathbf{b})$. Hence $f^\dagger$ is trivially pin-monotone. We conclude that f is pin-monotone, so, since f was an arbitrary function in MON, we conclude that MON $\subseteq$ Pin-MON.

To show that the inclusion is strict, note that δ_0 is not monotone because $\delta_0(0) > \delta_0(1)$, yet it is pin-monotone by Lemma 61. Therefore, MON $\subsetneq$ Pin-MON. □

Lemma 63. *The set* Pin-MON *is a functional clone.*

Proof. The irredundant version of EQ is the constant function $\text{EQ}^\dagger(x) = 1$ for $x \in \{0,1\}$. Hence EQ is pin-monotone, as there is no pair of bit strings $\mathbf{a}, \mathbf{b}$ differing in one bit such that $\text{EQ}^\dagger(\mathbf{a}) > \text{EQ}^\dagger(\mathbf{b})$.

To show that Pin-MON is a functional clone, it remains to show that the set is closed under permutation of arguments, introduction of fictitious arguments, product, and summation (see Lemma 14).

- It is straightforward to see that permuting arguments does not destroy the property of being pin-monotone, so Pin-MON is closed under permutation of arguments.
- Consider the effect of introducing a fictitious argument: let $h(\mathbf{x}, y) = f(\mathbf{x})$, where $f \in$ Pin-MON. The fictitious argument is in an equivalence class of its own since $h(\mathbf{x}, 0) = h(\mathbf{x}, 1)$ for all $\mathbf{x} \in \{0,1\}^{\text{arity}(f)}$. Hence $h^\dagger(\mathbf{x}, y) = f^\dagger(\mathbf{x})$. Let $k = \text{arity}(f^\dagger)$. We look at input bit strings to $h^\dagger$ that differ in exactly one bit and distinguish cases according to whether that single bit is the fictitious bit or not.

  Case 1. Suppose there exist $\mathbf{a}, \mathbf{b} \in \{0,1\}^{k+1}$ such that $\mathbf{a}[j] = 0$ and $\mathbf{b}[j] = 1$ for some $j \in [k]$, $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k+1] \setminus \{j\}$, and $h^\dagger(\mathbf{a}) > h^\dagger(\mathbf{b})$. Let $\mathbf{a}', \mathbf{b}'$ be the first k bits of $\mathbf{a}, \mathbf{b}$, respectively. Then $f^\dagger(\mathbf{a}') > f^\dagger(\mathbf{b}')$. Thus, by pin-monotonicity of f, $f^\dagger(\mathbf{b}') = 0$ and

$f^\dagger(\mathbf{c}') = 0$ for all $\mathbf{c}' \in \{0,1\}^k$ such that $\mathbf{c}'[j] = 1$. This implies $h^\dagger(\mathbf{b}) = 0$ and $h^\dagger(\mathbf{c}) = 0$ for all $\mathbf{c} \in \{0,1\}^{k+1}$ such that $\mathbf{c}[j] = 1$.

Case 2. If $\mathbf{a}, \mathbf{b} \in \{0,1\}^{k+1}$ such that $\mathbf{a}[k+1] = 0$ and $\mathbf{b}[k+1] = 1$ and $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k]$, then $h^\dagger(\mathbf{a}) = h^\dagger(\mathbf{b})$ by definition.

Hence h is pin-monotone, which implies that the set of pin-monotone functions is closed under introduction of fictitious arguments.

- To show closure under taking products, let $h(\mathbf{x}) = f(\mathbf{x})g(\mathbf{x})$, where $f, g \in$ Pin-MON.

  If there exist $i, j \in [\text{arity}(f)]$ with $i \neq j$ such that $f(\mathbf{x}) \neq 0$ implies $\mathbf{x}[i] = \mathbf{x}[j]$, then $h(\mathbf{x}) \neq 0$ implies $\mathbf{x}[i] = \mathbf{x}[j]$, so i and j need to be identified to make the irredundant function $h^\dagger$. The same holds for any pair of variables that are in the same equivalence class for g. Hence, $h^\dagger(\mathbf{x}) = f'(\mathbf{x})g'(\mathbf{x})$, where f' arises from $f^\dagger$ by identifying some subset of the variables (namely, variables that are in the same equivalence class for g but not f), and similarly for g' and $g^\dagger$.

  Let $k = \text{arity}(h^\dagger)$ and suppose $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ satisfy $\mathbf{a}[j] = 0$ and $\mathbf{b}[j] = 1$ for some $j \in [k]$, $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k] \setminus \{j\}$, and $h^\dagger(\mathbf{a}) > h^\dagger(\mathbf{b})$. Then, $f'(\mathbf{a}) > f'(\mathbf{b})$ or $g'(\mathbf{a}) > g'(\mathbf{b})$ (or both). Assume the former; in the other case, the argument is analogous with g in place of f.

  The function f' arises from $f^\dagger$ by identifying some variables. In particular, suppose the j-th variable of f' results from identifying variables $\ell_1, \ldots, \ell_m$ of $f^\dagger$ for some positive integer m. Let $\mathbf{a}'$ be the extension of $\mathbf{a}$ that satisfies $\mathbf{a}'[p] = \mathbf{a}'[q]$ for all pairs $p, q \in [\text{arity}(f^\dagger)]$ such that variable x_p was identified with variable x_q in going from $f^\dagger$ to f', and let $\mathbf{b}'$ be the analogous extension of $\mathbf{b}$. Then $f^\dagger(\mathbf{a}') = f'(\mathbf{a}) > f'(\mathbf{b}) = f^\dagger(\mathbf{b}')$. Furthermore, for all $s \in [m]$, $\mathbf{a}'[\ell_s] = 0$ and $\mathbf{b}'[\ell_s] = 1$. Finally, for all $i \in [\text{arity}(f^\dagger)] \setminus \{\ell_1, \ldots, \ell_m\}$, $\mathbf{a}'[i] = \mathbf{b}'[i]$.

  To use the pin-monotonicity property, we need two input bit strings for $f^\dagger$ that differ in exactly one bit. If $m = 1$, then $\mathbf{a}'$ and $\mathbf{b}'$ are such a pair. Otherwise, we will identify two suitable bit strings from a sequence of $(m+1)$ bit strings, whose first member is $\mathbf{a}'$, whose last member is $\mathbf{b}'$, and where neighbouring members differ in exactly one bit. Formally, define the sequence $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_m$ as follows. Let $\mathbf{a}_0 = \mathbf{a}'$ and for $s \in [m]$, let $\mathbf{a}_s$ be the bit string that agrees with $\mathbf{a}'$ except on the bits with indices $\ell_1, \ldots, \ell_s$. Then, $\mathbf{a}_m = \mathbf{b}'$ and furthermore, for each $s \in [m]$, $\mathbf{a}_{s-1}$ and $\mathbf{a}_s$ differ in exactly the bit with index ℓ_s. In other words, $\mathbf{a}_{s-1}[\ell_s] = 0$, $\mathbf{a}_s[\ell_s] = 1$ and $\mathbf{a}_{s-1}[i] = \mathbf{a}_s[i]$ for $i \in [\text{arity}(f^\dagger)] \setminus \{\ell_s\}$.

  Since $f^\dagger(\mathbf{a}_0) > f^\dagger(\mathbf{a}_m)$, there is an integer $t \in [m]$ such that $f^\dagger(\mathbf{a}_{t-1}) > f^\dagger(\mathbf{a}_t)$. Then, by pin-monotonicity of f, we have $f^\dagger(\mathbf{a}_t) = 0$ and $f^\dagger(\mathbf{c}) = 0$ for all $\mathbf{c} \in \{0,1\}^{\text{arity}(f^\dagger)}$ with $\mathbf{c}[\ell_t] = 1$. In particular, $\mathbf{b}'[\ell_t] = 1$, so we have $f^\dagger(\mathbf{b}') = 0$ and thus $f'(\mathbf{b}) = 0$ and $h^\dagger(\mathbf{b}) = f'(\mathbf{b})g'(\mathbf{b}) = 0$.

  It remains to show that $h^\dagger(\mathbf{d}) = 0$ for all $\mathbf{d} \in \{0,1\}^k$ that agree with $\mathbf{b}$ on the j-th bit. Suppose $\mathbf{d} \in \{0,1\}^k$ with $\mathbf{d}[j] = 1$. Let $\mathbf{d}'$ be the extension of $\mathbf{d}$ that satisfies $\mathbf{d}'[p] = \mathbf{d}'[q]$ for all pairs $p, q \in [\text{arity}(f^\dagger)]$ such that variable x_p was identified with variable x_q in going from $f^\dagger$ to f'. Then $\mathbf{d}'[\ell_t] = 1$ since the variables with indices $\ell_1, \ldots, \ell_m$ are identified to form x_j in going from $f^\dagger$ to f'. Thus $f^\dagger(\mathbf{d}') = 0$, which implies that $f'(\mathbf{d}) = 0$ and $h^\dagger(\mathbf{d}) = f'(\mathbf{d})g'(\mathbf{d}) = 0$. Hence, h is pin-monotone and Pin-MON is closed under products.

- Finally, consider the effect of summation over a variable: let $h(\mathbf{x}) = f(\mathbf{x}, 0) + f(\mathbf{x}, 1)$, where $f \in$ Pin-MON. We distinguish cases according to whether the final input variable of f is equivalent to another input variable.

  Case 1. The final input variable of f is in an equivalence class of its own. Then $h^\dagger(\mathbf{x}) = f^\dagger(\mathbf{x}, 0) + f^\dagger(\mathbf{x}, 1)$. Let $k = \text{arity}(h^\dagger)$ and suppose there exist $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$ such that $\mathbf{a}[j] = 0$ and $\mathbf{b}[j] = 1$ for some $j \in [k]$, $\mathbf{a}[i] = \mathbf{b}[i]$ for all $i \in [k] \setminus \{j\}$, and $h^\dagger(\mathbf{a}) > h^\dagger(\mathbf{b})$.

Then either $f^\dagger(\mathbf{a},0) > f^\dagger(\mathbf{b},0)$ or $f^\dagger(\mathbf{a},1) > f^\dagger(\mathbf{b},1)$ (or both). In either case by pin-monotonicity of f we have $f^\dagger(\mathbf{c},d) = 0$ for all $d \in \{0,1\}$ and $\mathbf{c} \in \{0,1\}^k$ with $\mathbf{c}[j] = 1$. Therefore, $h^\dagger(\mathbf{c}) = f^\dagger(\mathbf{c},0) + f^\dagger(\mathbf{c},1) = 0$ for any such $\mathbf{c}$: that is, h is pin-monotone.

Case 2. The final input variable of f is equivalent to the ℓ-th input variable of f, where $\ell \in [\text{arity}(f) - 1]$. Then $h(\mathbf{x}) = f(\mathbf{x},0) + f(\mathbf{x},1) = f(\mathbf{x},\mathbf{x}[\ell])$ for all $\mathbf{x} \in \{0,1\}^{\text{arity}(h)}$, since $f(\mathbf{x},y)$ is 0 unless $y = \mathbf{x}[\ell]$. Furthermore, $h^\dagger(\mathbf{x}) = f'(\mathbf{x},\mathbf{x}[\ell])$, where f' arises from f by identifying all variables in the same equivalence classes, except the ℓ-th and the final variable. But $f'(\mathbf{x},\mathbf{x}[\ell]) = f^\dagger(\mathbf{x})$ since identification of two variables that are always equal does not change the value of the function. Therefore $h^\dagger(\mathbf{x}) = f^\dagger(\mathbf{x})$, and pin-monotonicity of f immediately implies pin-monotonicity of h.

Hence, Pin-MON is closed under summation.

This concludes the proof. □

We can now prove the central theorem of this section.

Theorem 59 (restated). *There is a functional clone* Pin-MON *containing* δ_0 *and* δ_1 *such that* MON *is a strict subset of* Pin-MON *and* Pin-MON *contains no strictly decreasing permissive unary functions.*

Proof. Recall that Pin-MON is the set of all pin-monotone functions. This is shown to be a functional clone in Lemma 63. The pinning functions δ_0, δ_1 are in Pin-MON by Lemma 61. Furthermore, by Lemma 62, we have MON $\subsetneq$ Pin-MON.

Now let f be an arbitrary strictly decreasing permissive unary function, i.e. $f(0) > f(1) > 0$; then f is irredundant. The bit strings $\mathbf{a} = 0$, $\mathbf{b} = 1$ satisfy the three conditions of Definition 60 for f. Yet $f(\mathbf{b}) \neq 0$, so f is not pin-monotone. Since f was arbitrary, this establishes that Pin-MON does not contain any strictly decreasing permissive unary functions. □

As the pinning functions are contained in Pin-MON, we also have the following corollary.

Corollary 64. *For any* $\mathcal{F} \subseteq$ Pin-MON, $\langle \mathcal{F}, \delta_0, \delta_1 \rangle$ *does not contain any strictly decreasing permissive unary functions.*

References

[1] Andrei Bulatov, Leslie Ann Goldberg, Mark Jerrum, David Richerby, and Stanislav Živný. Functional clones and expressibility of partition functions. *Theoretical Computer Science*, 687:11–39, July 2017.

[2] Andrei A. Bulatov, Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, and Colin McQuillan. The Expressibility of Functions on the Boolean Domain, with Applications to Counting CSPs. *Journal of the ACM*, 60(5):32:1–32:36, October 2013.

[3] Jin-Yi Cai and Xi Chen. *Dichotomies for Counting Problems: Volume 1, Boolean Domain.* Cambridge University Press, 2018.

[4] Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and Back: Dichotomy for Holant^c Problems. *Algorithmica*, 64(3):511–533, March 2012.

[5] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *Journal of Computer and System Sciences*, 80(1):217–236, February 2014.

[6] Xi Chen, Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, Pinyan Lu, Colin McQuillan, and David Richerby. The complexity of approximating conservative counting CSPs. *Journal of Computer and System Sciences*, 81(1):311–329, February 2015.

[7] Nadia Creignou, Phokion G. Kolaitis, and Bruno Zanuttini. Structure identification of boolean relations and plain bases for co-clones. *Journal of Computer and System Sciences*, 74(7):1103–1115, 2008.

[8] Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity.* PhD thesis, Universität des Saarlandes, 2015.

[9] Ronald de Wolf. A Brief Introduction to Fourier Analysis on the Boolean Cube. *Theory of Computing Graduate Surveys*, 1:1–20, 2008.

[10] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The Relative Complexity of Approximate Counting Problems. *Algorithmica*, 38(3):471–500, March 2004.

[11] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. The Complexity of Weighted Boolean #CSP. *SIAM Journal on Computing*, 38(5):1970–1986, January 2009.

[12] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for Boolean #CSP. *Journal of Computer and System Sciences*, 76(3):267–277, May 2010.

[13] Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the Partition Function for the Antiferromagnetic Ising and Hard-Core Models. *Combinatorics, Probability and Computing*, 25(4):500–559, July 2016.

[14] Leslie Ann Goldberg and Mark Jerrum. The Complexity of Ferromagnetic Ising with Local Fields. *Combinatorics, Probability and Computing*, 16(1):43–61, January 2007.

[15] Leslie Ann Goldberg and Mark Jerrum. Approximating the Tutte polynomial of a binary matroid and other related combinatorial polynomials. *Journal of Computer and System Sciences*, 79(1):68–78, February 2013.

[16] Leslie Ann Goldberg, Mark Jerrum, and Mike Paterson. The computational complexity of two-state spin systems. *Random Structures & Algorithms*, 23(2):133–154, September 2003.

[17] Mark Jerrum and Alistair Sinclair. Approximating the Permanent. *SIAM Journal on Computing*, 18(6):1149–1178, December 1989.

[18] Liang Li, Pinyan Lu, and Yitong Yin. Correlation Decay up to Uniqueness in Spin Systems. In Sanjeev Khanna, editor, *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 67–84, Philadelphia, PA, January 2013. Society for Industrial and Applied Mathematics. Full version at arXiv:1111.7064.

[19] Jingcheng Liu, Pinyan Lu, and Chihao Zhang. The Complexity of Ferromagnetic Two-spin Systems with External Fields. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 843–856, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[20] Reinhard Pöschel and Lev A. Kalužnin. *Funktionen- und Relationenalgebren.* DVW, Berlin, 1979.

[21] Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation Algorithms for Two-State Anti-Ferromagnetic Spin Systems on Bounded Degree Graphs. *Journal of Statistical Physics*, 155(4):666–686, May 2014.

[22] Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 361–369, 2012.

[23] Donald M. Topkis. Minimizing a Submodular Function on a Lattice. *Operations Research*, 26(2):305–321, April 1978.

[24] Leslie G. Valiant. Holographic Algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, January 2008.