LOG-SUPERMODULAR FUNCTIONS, FUNCTIONAL CLONES AND COUNTING CSP'S

ANDREI BULATOV, MARTIN DYER, LESLIE ANN GOLDBERG, AND MARK JERRUM

Abstract. Motivated by a desire to understand the computational complexity of counting constraint satisfaction problems (counting CSPs), particularly the complexity of approximation, we study functional clones of functions on the Boolean domain, which are analogous to the familiar relational clones constituting Post's lattice. One of these clones is the collection of log-supermodular (lsm) functions, which turns out to play a significant role in classifying counting CSPs. In our study, we assume that non-negative unary functions (weights) are available. Given this, we prove that there are no functional clones lying strictly between the clone of lsm functions and the total clone (containing all functions). Thus, any counting CSP that contains a single non-lsm function is computationally as hard as it possibly could be. Furthermore, any non-trivial functional clone (in some precise sense) contains the binary function IMP (implies with false interpreted as 0 and true as 1). As a consequence, all non-trivial counting CSPs (with non-negative unary weights assumed to be available) are computationally at least as difficult as #BIS, the problem of counting independent sets in a bipartite graph. There is empirical evidence that #BIS is hard to solve even approximately, in the sense of FPRAS.

1. Introduction

In the classical setting, a constraint satisfaction problem $\mathsf{CSP}(\Gamma)$ is specified by a finite domain D and constraint language Γ, which is a set of relations of varying arities over D. A instance of $\mathsf{CSP}(\Gamma)$ is a set of n variables taking values in D, together with a set of constraints on those variables. Each constraint is an a-ary relation R from Γ applied to an a-tuple of variables, the scope of the constraint.

The relational clone $\langle\Gamma\rangle_{\mathrm{R}}$ generated by a set Γ of relations is the set of relations that are expressible, in some precise sense termed "pp-definability", in terms of the base relations Γ. It turns out that if two sets of relations Γ and Γ' generate the same relational clone $\langle\Gamma\rangle_{\mathrm{R}} = \langle\Gamma'\rangle_{\mathrm{R}}$, then the computational complexity of the corresponding CSPs, $\mathsf{CSP}(\Gamma)$ and $\mathsf{CSP}(\Gamma')$, are the same. Relational clones have played a key role in the development of the complexity theory of CSPs: instead of considering all sets of relations Γ, one only needs to consider the ones that are relational clones. For an introduction to the algebraic theory of relational clones, see, for example, the expository chapter of Cohen and Jeavons [7].

Recently, there has been considerable interest in the computational complexity of counting CSPs. Here, the goal is to count the number of solutions rather than merely

The work described in this paper was partly supported by EPSRC Research Grant (refs EP/I011528/1, EP/I011935/1 and EP/I012087/1) "Computational Counting", and by NSERC Discovery Grant. Part of the work was supported by a visit to the Isaac Newton Institute for Mathematical Sciences, as part of the programme "Discrete Analysis".

to decide if one exists. In fact, in order to encompass the computation of partition functions of models from statistical physics and other generating functions, it is reasonable to consider weighted sums, which can be expressed by replacing the relations in the constraint language by real- or complex-valued functions. Then the weight of an assignment is the product of the function values corresponding to that assignment, while the value of the CSP instance itself is the sum of the weights of all assignments. If I is an instance of such a counting CSP then we denote this weighted sum by $Z(I)$, and call it the "partition function of I" by analogy with the concept in statistical physics. For a finite set of functions Γ we are interested in the problem $\#\mathsf{CSP}(\Gamma)$: given an instance I using only functions from Γ, output $Z(I)$.

Our first goal (see §2) is to answer the question: what is the analogue of pp-definability, and hence of relational clones, in the context of (weighted) counting CSPs (#CSPs), and what insight does it provide into the computational complexity of these problems? At a high level, the answer to the first question is clear. View the relations in Γ as predicates. A relation is pp-definable over Γ in the classical sense if it can be expressed as the projection of a conjunction of predicates in Γ. (Projection is the operation of existential quantification over a certain subset of variables.) In order to adapt this concept to the algebraic or counting setting, we should replace a conjunction of relations by a product of functions, and replace existential quantification (projection) by summation. However, in defining a counting analogue of pp-definability, a number of detailed decisions have to be made, and a number of delicate issues faced.

We call our proposed analogue of pp-definability "pps_ω-definability", and our analogue of relational clone "functional clone". There is at least one proposal in the literature for extending pp-definability to the algebraic/functional setting, that of Yamakami [20]. However, pps_ω-definability is more liberal than the corresponding notion in [20], and leads to a more inclusive functional clone. Our notion of pps_ω-definability includes a limiting operation. Without this limit, a functional clone may contain arbitrarily close approximations to a function F of interest, without including F itself.

Aside from a desire for tidiness, there is a good empirical motivation for introducing limits. Just as pp-definability is closely related to polynomial-time reductions between classical CSPs, so is pps_ω-definability related to approximation-preserving reductions between counting CSPs. (Lemma 11 is a precise statement of this connection.) Now, many approximation-preserving reductions in the literature are based not on a fixed "gadget" but in sequences of gadgets of increasing size that come arbitrarily close to some property without actually attaining it [13]. Our notion of pps_ω-definability seems exactly to capture this phenomenon.

Our second, more concrete goal (see §3–§5) is to explore the role of log-supermodular functions in the classification of functional clones, and hence in the complexity of approximating #CSPs. We restrict attention to the Boolean situation; that is, the domain is $\{0,1\}$ and the allowed functions are of the form $\{0,1\}^k \to \mathbb{R}^{\geq 0}$ for some integer k. A function with Boolean domain is said to be log-supermodular if the logarithm of it is supermodular. It is a non-trivial fact (Lemma 5) that the set LSM of log-supermodular functions is in fact a functional clone. We examine the landscape of functional clones under the assumption that non-negative unary functions (weights) are available. (Such

an assumption is quite usual in related work, such as Cai, Lu and Xia's work on classifying "Holant*" problems [6].) Adding non-negative weights makes the classification of functional clones more tractable, though we are still unable to provide a complete inventory. On the other hand, adding all unary weights leads to a less rich (and more pessimistic) landscape [20]: negative weights introduce cancellation, which tends to drive approximate counting CSPs in the direction of intractability.

One particularly simple functional clone is the one generated by disequality. (Following convention, we allow equality for free, in addition to the non-negative weights mentioned earlier.) A counting CSP derived from this clone is trivial to solve exactly, as the partition function factorises. Let us say that functions from this clone are of "product form". Our main result (Theorem 10) is that any clone that contains a function F that is not of product form necessarily contains IMP, the binary (i.e., arity-2) function that takes the value 1, unless its first argument is 1 and its second is 0, when it takes the value 0. (The complexity-theoretic consequence of this will be discussed presently.) Furthermore (also Theorem 10), if F is not log-supermodular, then the clone contains all functions. Note that a large part of the functional clone landscape — below the clone generated by IMP and above LSM — is very simple. If there is a complex landscape of functional clones it must lie between the functional clone generated by IMP and the class of functions LSM.

We present also an efficient version of pps_ω-definability, and a corresponding notion of functional clone, that allows complexity-theoretical consequences to be deduced (Theorem 12). This is the third contribution of the paper (see §6). The last three authors, together with Greenhill [10], studied the complexity of counting problems expressible using IMP. They identified a class of natural problems of this form (which has since grown considerably) which are interreducible via approximation-preserving reduction, and for which no efficient approximation algorithm (FPRAS) is known. They conjectured that problems in this class do not admit an FPRAS. If this is so then $\#\mathsf{CSP}(\mathcal{F})$ is computationally intractable (in the presence of nonnegative weights) whenever $\mathcal{F}$ contains a function F that is not of product form. Furthermore, if F is not log-supermodular, then the counting problem $\#\mathsf{CSP}(\mathcal{F})$ is universal for Boolean counting CSPs and hence is provably NP-hard to approximate.

Although we focus on approximation of the partition functions of (weighted) #CSPs in this paper, there is of course an extensive literature on exact computation; see, e.g., Cai, Chen and Lu [5] and prior work.

2. Functional clones

Let $(R, +, \times)$ be any subsemiring of $(\mathbb{C}, +, \times)$, where $\mathbb{C}$ denotes the complex numbers, and D a finite domain. For $n \in \mathbb{N}$, denote by $\mathcal{U}_n$ the set of all functions $D^n \to R$; also denote by $\mathcal{U} = \mathcal{U}_0 \cup \mathcal{U}_1 \cup \mathcal{U}_2 \cup \cdots$ the set of functions of all arities. Suppose $\mathcal{F} \subseteq \mathcal{U}$ is some collection of functions, $V = \{v_1, \ldots, v_n\}$ is a set of variables and $\boldsymbol{x} : \{v_1, \ldots, v_n\} \to D$ is an assignment to those variables. An atomic formula has the form $\varphi = G(v_{i_1}, \ldots, v_{i_a})$ where $G \in \mathcal{F}$, $a = a(G)$ is the arity of G, and $(v_{i_1}, v_{i_2}, \ldots, v_{i_a}) \in V^a$ is a scope. Note that repeated variables are allowed. The function $F_\varphi : D^n \to R$ represented by the atomic formula $\varphi = G(v_{i_1}, \ldots, v_{i_a})$ is just

$$F_\varphi(\boldsymbol{x}) = G(\boldsymbol{x}(v_{i_1}), \ldots, \boldsymbol{x}(v_{i_a})) = G(x_{i_1}, \ldots, x_{i_a}),$$

where from now on we write $x_j = \boldsymbol{x}(v_j)$.

A pps-formula ("primitive product summation formula") is a summation of a product of atomic formulas. A pps-formula ψ over $\mathcal{F}$ in variables $V' = \{v_1, \ldots, v_{n+m}\}$ has the form

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

where φ_j are all atomic formulas over $\mathcal{F}$ in the variables V'. (The variables V are free, and the others, $V' \setminus V$, are bound.) The formula ψ specifies a function $F_\psi : D^n \to R$ in the following way:

$$F_\psi(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in D^m} \prod_{j=1}^{s} F_{\varphi_j}(\boldsymbol{x}, \boldsymbol{y}), \tag{2}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are assignments $\boldsymbol{x} : \{v_1, \ldots, v_n\} \to D$ and $\boldsymbol{y} : \{v_{n+1}, \ldots, v_{n+m}\} \to D$. To make the next step we suppose that R is dense-in-itself with respect to the usual topology on $\mathbb{C}$. Then we say that an a-ary function F is pps_ω-definable over $\mathcal{F}$ if there exists a finite subset S_F of $\mathcal{F} \cup \{\mathrm{EQ}\}$, where EQ is the binary equality function defined by $\mathrm{EQ}(x, x) = 1$ and $\mathrm{EQ}(x, y) = 0$ for $x \neq y$, such that, for every $\varepsilon > 0$, there is an a-ary function $\widehat{F}$ specified by a pps-formula over S_F with

$$\|\widehat{F} - F\|_\infty = \max_{\boldsymbol{x} \in D^a} |\widehat{F}(\boldsymbol{x}) - F(\boldsymbol{x})| < \varepsilon.$$

Denote the set of functions in $\mathcal{U}$ that are pps_ω-definable over $\mathcal{F} \cup \{\mathrm{EQ}\}$ by $\langle \mathcal{F} \rangle_\omega$; we call this the pps_ω-*definable functional clone* generated by $\mathcal{F}$. Observe that functions in $\langle \mathcal{F} \rangle_\omega$ are determined only by finite subsets of $\mathcal{F}$. Also, although some functions taking values outside R (including partial functions, which are undefined, or infinite, on some inputs) may be pps_ω-definable over $\mathcal{F} \cup \{\mathrm{EQ}\}$, $\langle \mathcal{F} \rangle_\omega$ is defined to include only functions in $\mathcal{U}$. The universal class of functions $\mathcal{U}$ in operation at any time will be clear from the context.

That completes the setup for expressibility. In order to deduce complexity results, we need an effective version of $\langle \mathcal{F} \rangle_\omega$. We say that a function F is *efficiently* pps_ω-definable over $\mathcal{F}$ if there is a finite subset S_F of $\mathcal{F}$, and a TM $\mathcal{M}_{F,S_F}$ 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 $\log \varepsilon^{-1}$. Denote the set of functions in $\mathcal{U}$ that are efficiently pps_ω-definable over $\mathcal{F} \cup \{\mathrm{EQ}\}$ by $\langle \mathcal{F} \rangle_{\omega,p}$; we call this the *efficient* pps_ω-*definable functional clone* generated by $\mathcal{F}$,

The following useful observation is immediate from the definition of $\langle \mathcal{F} \rangle_{\omega,p}$.

Observation 1. Suppose $F \in \langle \mathcal{F} \rangle_{\omega,p}$. Then there is a finite subset S_F of $\mathcal{F}$ such that $F \in \langle S_F \rangle_{\omega,p}$.

Since pps-formulas are defined using sums of products (with just one level of each), we need to check that functions that are pps_ω-definable in terms of functions that are themselves pps_ω-definable over $\mathcal{F}$ are actually directly pps_ω-definable over $\mathcal{F}$. The following lemma ensures that this is the case.

Lemma 2. *If $G \in \langle \mathcal{F} \rangle_\omega$ [or $G \in \langle \mathcal{F} \rangle_{\omega,p}$] then $\langle \mathcal{F}, G \rangle_\omega = \langle \mathcal{F} \rangle_{\omega,p}$ [resp., $\langle \mathcal{F}, G \rangle_{\omega,p} = \langle \mathcal{F} \rangle_{\omega,p}$].*

Lemma 2 may have wider applications in the study of approximate counting problems. Often, approximation-preserving reductions between counting problems are complicated to describe and difficult to analyse, owing to the need to track error estimates. Lemma 2 suggests breaking the reduction into smaller steps, and analysing each of them independently. This assumes, of course, that the reductions are pps_ω-definable, but that often seems to be the case in practice.

3. Relational Clones and Non-negative functions

A function $F \in \mathcal{U}$ is a *Boolean function* if its range is contained in $\{0,1\}$. F encodes a relation R as follows: $\boldsymbol{x}$ is in the relation R iff $F(\boldsymbol{x}) = 1$. We will not distinguish between relations and the Boolean functions that define them. Suppose that $\mathcal{R} \subseteq \mathcal{U}$ is a set of Boolean functions/relations. A pp-formula over $\mathcal{R}$ is an existantially quantified product of atomic formulas. More precisely, a pp-formula ψ over $\mathcal{R}$ in variables $V' = \{v_1, \ldots, v_{n+m}\}$ has the form

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

where φ_j are all atomic formulas over $\mathcal{R}$ in the variables V'. As before, the variables $V = \{v_1, \ldots, v_n\}$ are called "free", and the others, $V' \setminus V$, are called "bound". The formula ψ specifies a Boolean function $R_\psi : D^n \to \{0,1\}$ in the following way. $R_\psi(\boldsymbol{x}) = 1$ if there is a vector $\boldsymbol{y} \in D^m$ such that $\bigwedge_{j=1}^{s} R_{\varphi_j}(\boldsymbol{x}, \boldsymbol{y})$ evaluates to "1", where $\boldsymbol{x}$ and $\boldsymbol{y}$ are assignments $\boldsymbol{x} : \{v_1, \ldots, v_n\} \to D$ and $\boldsymbol{y} : \{v_{n+1}, \ldots, v_{n+m}\} \to D$; $R_\psi(\boldsymbol{x}) = 0$ otherwise. We refer to the pp-formula as an "implementation" of R_ψ.

A *relational clone* (often called a "co-clone") is a set of Boolean relations containing the equality relation and closed under finite Cartesian products, projections, and identification of variables. A *basis* [8] for the relational clone I is a set $\mathcal{R}$ of Boolean relations such that the relations in I are exactly the relations that can be implemented with a pp-formula over $\mathcal{R}$. Every relational clone has such a basis.

For every set $\mathcal{R}$ of Boolean relations, let $\langle\mathcal{R}\rangle_{\mathrm{R}}$ denote the set of relations that can be represented by a pp-formula over $\mathcal{R} \cup \{\mathrm{EQ}\}$. It is well-known that if $R \in \langle\mathcal{R}\rangle_{\mathrm{R}}$ then $\langle\mathcal{R} \cup \{R\}\rangle_{\mathrm{R}} = \langle\mathcal{R}\rangle_{\mathrm{R}}$ Thus, $\langle\mathcal{R}\rangle_{\mathrm{R}}$ is in fact a relational clone with basis $\mathcal{R}$.

A basis $\mathcal{R}$ for a relational clone $\langle\mathcal{R}\rangle_{\mathrm{R}}$ is called a "plain basis" [8, Definition 1] if every member of $\langle\mathcal{R}\rangle_{\mathrm{R}}$ is definable by a $\mathrm{CNF}(\mathcal{R})$-formula (a pp-formula over $\mathcal{R}$ with no $\exists$).

For most of this paper, we restrict attention to the Boolean domain $D = \{0,1\}$ and to the codomain $R = \mathbb{R}^{\geq 0}$ of non-negative real numbers. For $n \in \mathbb{N}$, denote by $\mathcal{B}_n$ the set of all functions $\{0,1\}^n \to \mathbb{R}^{\geq 0}$; also denote by $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots$ the set of functions of all arities. The advantage of working with the Boolean domain is (i) that it comes with a well-developed theory of relational clones, and (ii) the concept of log-supermodular function makes sense (see §4). As explained in the introduction, the advantage of working with non-negative real numbers is that we thereby forbid cancellation, and potentially obtain a more nuanced expressibility/complexity landscape.

Given a function $F \in \mathcal{B}$, let R_F be the function corresponding to the relation underying F. That is, $R_F(\boldsymbol{x}) = 0$ if $F(\boldsymbol{x}) = 0$ and $R_F(\boldsymbol{x}) = 1$ if $F(\boldsymbol{x}) > 0$. The following straightforward lemma will be useful.

Lemma 3. *Suppose $\mathcal{F} \subseteq \mathcal{B}$. Then*

$$\langle \{R_F \mid F \in \mathcal{F}\} \rangle_{\mathrm{R}} = \{R_F \mid F \in \langle \mathcal{F} \rangle\}.$$

Since we want to be able to derive computational results, we now restrict attention to functions whose co-domains are restricted to efficiently-computable real numbers. A real number is polynomial-time computable if the first n bits of its binary expansion can be computed in time polynomial in n. Let $\mathbb{R}^p$ denote the set of non-negative real numbers that are polynomial-time computable. For $n \in \mathbb{N}$, denote by $\mathcal{B}_n^p$ the set of all functions $\{0,1\}^n \to \mathbb{R}^p$; also denote by $\mathcal{B}^p = \mathcal{B}_0^p \cup \mathcal{B}_1^p \cup \mathcal{B}_2^p \cup \cdots$ the set of functions of all arities.

Remark 4. If $\mathcal{F} \subseteq \mathcal{B}^p$ then real numbers appearing as function values must be polynomial-time computable. This is a stronger requirement than the *efficiently approximable* real numbers defined in [14], but it results in a more uniform treatment of limits when we discuss efficient pps_ω-definability using these functions.

4. Log-supermodular functions

A function $F \in \mathcal{B}_n$ is log-supermodular (lsm) if $F(\boldsymbol{x} \vee \boldsymbol{y})F(\boldsymbol{x} \wedge \boldsymbol{y}) \geq F(\boldsymbol{x})F(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$. The terminology is justified by the observation that F is lsm if and only if $f = \ln F$ is supermodular, where $\ln 0$ is treated as $-\infty$, a formal entity that is operated on in the obvious way. We denote by $\mathsf{LSM} \subset \mathcal{B}$ the class of all lsm functions. The second part of our main result (Theorem 10) in some sense says that, in terms of expressivity, everything of interest takes place in the class LSM.

The class LSM fits naturally into our study of expressibility because of the following closure property: functions that are pps_ω-definable from lsm functions are lsm. The non-trivial step in showing this is encapsulated in the following lemma.

Lemma 5. *If $\mathcal{F} \subseteq \mathsf{LSM}$ is any set of lsm functions then $\langle \mathcal{F} \rangle_\omega \subseteq \mathsf{LSM}$.*

Proof. The only nontrivial step is to show that if $G \in \mathcal{B}_{n+m}$ is lsm then so is the function $G' \in \mathcal{B}_n$ defined by $G'(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \{0,1\}^m} G(\boldsymbol{x}, \boldsymbol{y})$. It is enough to prove the claim for $m = 1$, as the result for general m follows by induction. Suppose $\boldsymbol{a}', \boldsymbol{b}' \in \{0,1\}^n$, and let $A = \{(\boldsymbol{a}', 0), (\boldsymbol{a}', 1)\}$ and $B = \{(\boldsymbol{b}', 0), (\boldsymbol{b}', 1)\}$. We extend G to subsets of $\{0,1\}^{n+1}$ by letting $G(Z) = \sum_{\boldsymbol{z} \in Z} G(\boldsymbol{z})$ for all $Z \subseteq \{0,1\}^{n+1}$. Note that

$$G'(\boldsymbol{a}') = G(A) \quad \text{and} \quad G'(\boldsymbol{b}') = G(B).$$

Denote by $A \vee B$ and $A \wedge B$ the sets $A \vee B = \{\boldsymbol{a} \vee \boldsymbol{b} : \boldsymbol{a} \in A \text{ and } \boldsymbol{b} \in B\}$ and $A \wedge B = \{\boldsymbol{a} \wedge \boldsymbol{b} : \boldsymbol{a} \in A \text{ and } \boldsymbol{b} \in B\}$. Note that

$$G'(\boldsymbol{a}' \vee \boldsymbol{b}') = G(A \vee B) \quad \text{and} \quad G'(\boldsymbol{a}' \wedge \boldsymbol{b}') = G(A \wedge B).$$

Since G is lsm, we know that $G(\boldsymbol{a})G(\boldsymbol{b}) \leq G(\boldsymbol{a} \vee \boldsymbol{b})G(\boldsymbol{a} \wedge \boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^{n+1}$. Thus, applying the Ahlswede-Daykin "Four-functions Theorem" [1, Theorem 1] with $\alpha = \beta = \gamma = \delta = G$,

$$G'(\boldsymbol{a}')G'(\boldsymbol{b}') = G(A)G(B) \leq G(A \vee B)G(A \wedge B) = G'(\boldsymbol{a}' \vee \boldsymbol{b}')G'(\boldsymbol{a}' \wedge \boldsymbol{b}').$$

As $\boldsymbol{a}', \boldsymbol{b}' \in \{0,1\}^n$ were arbitrary, we see that G' is lsm. □

An important example of an lsm function is the 0,1-function "implies",

$$\mathrm{IMP}(x,y) = \begin{cases} 0, & \text{if } (x,y) = (1,0); \\ 1, & \text{otherwise.} \end{cases}$$

We also think of this as a binary relation $\mathrm{IMP} = \{(0,0),(0,1),(1,1)\}$. Complexity-theoretic issues will be treated in detail in §6. However, it may be helpful to give a pointer here to the importance of IMP in the study of approximate counting problems.

The problem #BIS is that of counting independent sets in a bipartite graph. Dyer et al. [10] exhibited a class of counting problems, including #BIS, which are interreducible via approximation-preserving reductions. Further natural problems have been shown to lie in this class, which appears to be of intermediate complexity between counting problems that are tractable (i.e., admitting a polynomial-time approximation algorithm) and those that are NP-hard to approximate. We will see in due course (Theorem 12) that #BIS and #CSP(IMP) are interreducible via approximation-preserving reductions, and hence are of equivalent difficulty.

We know from Lemma 5 that $\langle \mathrm{IMP}, \mathcal{B}_1 \rangle_\omega \subseteq \mathsf{LSM}$. It is an open question whether the inclusion is strict. A related open question is whether $\mathsf{LSM} = \langle \mathcal{F} \rangle_\omega$ for any finite set $\mathcal{F}$ of lsm functions. A similar question has been investigated by Živný et al. [21] in this context of optimisation problems, where summation is replaced by maximisation or minimisation.

5. The main result

5.1. **Pinnings and modular functions.** Let δ_0 be the unary function with $\delta_0(0) = 1$ and $\delta_0(1) = 0$ and let δ_1 be the unary function with $\delta_1(0) = 0$ and $\delta_1(1) = 1$.

If $n \geq 2$ then a 2-*pinning* of a function $F \in \mathcal{B}_n$ is a function

$$G_{i,j}(x_1, x_2) = F(c_1, \ldots, c_{i-1}, x_1, c_{i+1}, \ldots, c_{j-1}, x_2, c_j, \ldots, c_n),$$

where i and j are distinct indices in $\{1, \ldots, n\}$ and each c_k is in $\{0,1\}$. Clearly, every 2-pinning of F is in $\langle F, \mathcal{B}_1^p \rangle$, since the constants c_k can be implemented using the functions δ_0 and δ_1.

We say that a function $F \in \mathcal{B}_n$ is log-modular if $f = \ln F$ is modular, ie., $F(\boldsymbol{x} \vee \boldsymbol{y})F(\boldsymbol{x} \wedge \boldsymbol{y}) = F(\boldsymbol{x})F(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$.

We will use the following fact about 2-pinnings of lsm and log-modular functions. This follows directly from [17, Theorem 44.1] for the supermodular case, but Schrijver's proof also applies to the modular case. The lemma is originally due to Topkis [18].

Lemma 6 (Topkis)**.** *F is lsm iff every 2-pinning of F is lsm. F is log-modular iff every 2-pinning of F is log-modular.*

5.2. **Binary functions.** Recall that EQ is the binary relation $\mathrm{EQ} = \{(0,0),(1,1)\}$. (We used the name "EQ" to denote the equivalent binary function, but it will do no harm to use the same symbol for the relation and the function.) Denote by OR, NEQ, and NAND the binary relations $\mathrm{OR} = \{(0,1),(1,0),(1,1)\}$, $\mathrm{NEQ} = \{(0,1),(1,0)\}$, and $\mathrm{NAND} = \{(0,0),(0,1),(1,0)\}$.

Lemma 7. *Let $F \in \mathcal{B}_2^p$. Assuming $F(0,1) \geq F(1,0)$,*

(i) *if* $F(0,0)F(1,1) = F(0,1)F(1,0)$, *then* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \langle \mathcal{B}_1^p\rangle_{\omega,p}$;
(ii) *if* $R_F = \mathrm{EQ}$, *then* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \langle \mathcal{B}_1^p\rangle_{\omega,p}$;
(iii) *if* $R_F = \mathrm{NEQ}$, *then* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle_{\omega,p}$;
(iv) *if* $\mathrm{IMP} \subseteq R_F$ *and* $F(0,0)F(1,1) > F(0,1)F(1,0)$, *then* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}$;
(v) *otherwise,* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p} == \mathcal{B}^p$.

Remark 8. From Lemma 7, we see that IMP does not really occupy a special position in $\langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}$, nor does OR in $\langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Nevertheless, it is useful to label the classes this way, and we will do so.

Remark 9. From the proof of Lemma 7, we have the following inclusions between the four classes involved.

$$\langle \mathcal{B}_1^p\rangle_{\omega,p} \subseteq \begin{matrix}\langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle_{\omega,p} \\ \langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}\end{matrix} \subseteq \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}.$$

In fact, $\langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle_{\omega,p}$ and $\langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}$ are incomparable, and hence all the inclusions are actually strict. For one non-inclusion, note the clone $\langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}$ contains only lsm functions, and hence does not contain NEQ. For the other, we claim that arity-2 functions in the clone $\langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle_{\omega,p}$ are of one of three forms — $U_1(x)U_2(y)$, $U(x)\mathrm{EQ}(x,y)$ or $U(x)\mathrm{NEQ}(x,y)$ — and then observe that IMP matches none of these. The claim is a special case of a more general one, namely that any function in $\langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle_{\omega,p}$ can be expressed without summation, i.e., is of the form F_φ, where φ is a simple product of atomic formulas. This general claim can be established by induction on the number m of bound variables in (2).

5.3. **Boolean functional clones.**

Theorem 10. *Suppose* $F \in \mathcal{B}^p$.

- *If* $F \notin \langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle$ *then* $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$, *and hence* $\langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p} \subseteq \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$
- *If, in addition,* $F \notin \mathsf{LSM}$ *then* $\langle F, \mathcal{B}_1^p\rangle_{\omega,p} = \mathcal{B}^p$.

The non-effective version of the theorem — with $\mathcal{B}, \mathcal{B}_1$ *replacing* $\mathcal{B}^p, \mathcal{B}_1^p$, *and* $\langle\cdot\rangle_\omega$ *replacing* $\langle\cdot\rangle_{\omega,p}$ *— also holds.*

Proof. We start with the first part of the theorem, for which the aim is to show that either $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$ or $F \in \langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle$. Let C be the relational clone $\langle R_F, \delta_0, \delta_1\rangle_{\mathrm{R}}$. Since $\{R_F, \delta_0, \delta_1\} \subseteq \{R_{F'} \mid F' \in \{F\} \cup \mathcal{B}_1^p\}$, $C \subseteq \langle R_{F'} \mid F' \in \{F\} \cup \mathcal{B}_1^p\rangle_{\mathrm{R}}$, so by Lemma 3, $C \subseteq \{R_{F'} \mid F' \in \langle F, \mathcal{B}_1^p\rangle\}$.

First, suppose $\mathrm{IMP} \in C$. Then $\langle F, \mathcal{B}_1^p\rangle_{\omega,p}$ contains a function F' with $R_{F'} = \mathrm{IMP}$. The function F' falls into parts (iv) or (v) of Lemma 7, so by this lemma, $\langle F, \mathcal{B}_1^p\rangle_{\omega,p}$ is either $\langle \mathrm{IMP}, \mathcal{B}_1^p\rangle_{\omega,p}$ or $\langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Either way, $\langle F, \mathcal{B}_1^p\rangle_{\omega,p}$ contains IMP (as noted in Remark 9). Similarly, if $\mathrm{OR} \in C$ or $\mathrm{NAND} \in C$ then $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$.

We now consider the possibilities. If R_F is not affine, then Creignou, Khanna and Sudan [8, Lemma 5.30] have shown that one of IMP, OR and NAND is in C. This is also stated and proved as [11, Lemma 15].

In fact, the set of all relational clones (also called "co-clones") is finite, and is well understood. These are listed in [9, Table 2], which gives a plain basis for each relational clone. There is a similar table in [3] (though the bases given there are not plain). A graph illustrating the subset inclusions between the relational clones is depicted in [2,

Figure 2]. This graph is reproduced here as Figure 1. The relational clones are the vertices of the graph in Figure 1. A downwards edge from one clone to another indicates that the lower clone is a subset of the higher one. For example, since there is a path (in this case, an edge) from ID_1 down to IR_2 in Figure 1, we deduce that $\mathrm{IR}_2 \subset \mathrm{ID}_1$. We will require bases for only 3 relational clones: IR_2, ID_1, and IL_2; their plain bases are $\{\mathrm{EQ}, \delta_0, \delta_1\}$, $\{\mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1\}$, and $\{(x_1 \oplus \ldots \oplus x_k = c) \mid k \in \mathbb{N}, c \in \{0,1\}\}$, respectively.

If R_F is affine then the relations in C are given by linear equations, so C is either the relational clone IL_2 (whose plain basis the set of all Boolean linear equations) or C is some subset of IL_2, in which case it is below IL_2 in Figure 1.

Now, EQ, δ_0 and δ_1 are in C. The relational clone containing these relations (and nothing else) is IR_2, so C is a (not necessarily proper) superset of IR_2. Thus, C is (not necessarily strictly) above IR_2 in Figure 1. From the figure, it is clear that the only possibilities are that C is one of the relational clones IL_2, ID_1 and IR_2.

Now $\mathrm{IR}_2 \subset \mathrm{ID}_1$ and the plain basis of ID_1 is $\{\mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1\}$. Therefore if $C = \mathrm{IR}_2$ or $C = \mathrm{ID}_1$, then R_F is definable by a CNF formula over $\{\mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1\}$.

Suppose that $F(\boldsymbol{x})$ has arity n. To avoid trivialities, suppose that R_F is not the empty relation. Suppose that $\psi(v_1, \ldots, v_n)$ is a CNF formula over $\{\mathrm{EQ}, \mathrm{NEQ}, \delta_0, \delta_1\}$ implementing the relation $R_\psi = R_F$.

Let $V = \{v_1, \ldots, v_n\}$. Let ψ_i be the projection of ψ onto variable v_i. ψ_i is one of the three unary relations $\{(0)\}$, $\{(1)\}$, and $\{(0),(1)\}$. Let $V' = \{v_i \in V \mid \psi_i = \{(0),(1)\}\}$. ($V'$ is the set of variables that are not "pinned" in R_F.) For $v_i \in V'$ and $v_j \in V'$, let $\psi_{i,j}$ be the projection of ψ onto variables v_i and v_j. $\psi_{i,j}$ is a binary relation. Of the 16 possible binary relations, the only ones that can occur are EQ, NEQ and $\{0,1\}^2$. (The empty relation is ruled out since R_F is not empty. The four single-tuple binary relations are ruled out since v_i and v_j are in V'. For the same reason, the other four two-tuple binary relations are ruled out. The three-tuple binary relations are ruled out since $\psi_{i,j} \in \mathrm{ID}_1$.) We define an equivalence relation $\sim$ on V' in which $v_i \sim v_j$ iff $\psi_{i,j} \in \{\mathrm{EQ}, \mathrm{NEQ}\}$. Let V'' contain exactly one variable from each equivalence class in V'. Let $k = |V''|$. For convenience, we will assume $V'' = \{v_1, \ldots, v_k\}$. (This can be achieved by renaming variables.)

Now, for every assignment $\boldsymbol{x} : \{v_1, \ldots, v_k\} \to \{0,1\}$ there is exactly one assignment $\boldsymbol{y} : \{v_{k+1}, \ldots, v_n\} \to \{0,1\}$ such that $R_F(\boldsymbol{x}, \boldsymbol{y}) = 1$. Let $\sigma(\boldsymbol{x})$ be this assignment $\boldsymbol{y}$. Now, define the arity-k function G by $G(\boldsymbol{x}) = F(\boldsymbol{x}, \sigma(\boldsymbol{x}))$. Note that

$$G(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \{0,1\}^{n-k}} F(\boldsymbol{x}, \boldsymbol{y}), \tag{3}$$

where $\boldsymbol{y}$ is an assignment $\boldsymbol{y} : \{v_{k+1}, \ldots, v_n\} \to \{0,1\}$. By construction, $G(\boldsymbol{x})$ is a strictly positive function. Also, from (3), $G \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$. We finish with two cases.

Case 1. Every 2-pinning of G is log-modular. Then G is also log-modular, by Lemma 6. This means (see, for example, [4, Proposition 24]) that $g = \ln G$ is a linear function of $x_1, \ldots, x_k$ and $\neg x_1, \ldots, \neg x_k$ so $G \in \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$. Since $F(\boldsymbol{x}, \boldsymbol{y}) = R_F(\boldsymbol{x}, \boldsymbol{y}) G(\boldsymbol{x})$, we conclude that $F \in \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$.

Case 2. There is a 2-pinning G' of G that is not log-modular. Since G is strictly positive, so is G'. Since $G \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$, so is G'. By Lemma 7, (parts (iv) or (v)), $\mathrm{IMP} \in \langle G', \mathcal{B}_1^p \rangle_{\omega,p}$. By Lemma 2, $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$.

Finally, we consider the case in which $C = \mathrm{IL}_2$. Let $\oplus_3$ be the relation $\{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$ containing all triples whose Boolean sums are 0. From the plain basis of IL_2 (Table **??**), we see that the relation $\oplus_3$ is in C, so $\langle F, \mathcal{B}_1^p\rangle$ contains a function F' with $R_{F'} = \oplus_3$. Let F'' be the symmetrisation of F' implemented by

$$F''(x,y,z) = F'(x,y,z)F'(x,z,y)F'(y,x,z)F'(y,z,x)F'(z,x,y)F'(z,y,z).$$

Now let $\mu_0 = F''(0,0,0)$ and $\mu_2 = F''(0,1,1)$. Let U be the unary function with $U(0) = \mu_0^{-1/3}$ and $U(1) = \mu_0^{1/6}\mu_2^{-1/2}$. Note that since $F \in \mathcal{B}^p$, the appropriate roots of μ_0 and μ_2 are efficiently computable, so $U \in \mathcal{B}_1^p$. Now $\oplus_3(x,y,z) = U(x)U(y)U(z)F''(x,y,z)$, so $\oplus_3 \in \langle F, \mathcal{B}_1^p\rangle$. Finally, let U' be the unary function defined by $U'(0) = 1$ and $U'(1) = 2$ and let $G(x,z) = \sum_y \oplus_3(x,y,z)U'(y)$. Note that $G(0,0) = G(1,1) = 1$ and $G(0,1) = G(1,0) = 2$. By Lemma 15, G is in $\langle F, \mathcal{B}_1^p\rangle$. But by Lemma 7, $\mathrm{IMP} \in \langle G, \mathcal{B}_1^p\rangle_{\omega,p}$ so by Lemma 2, $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$.

We now prove Part 2 of the theorem. Suppose that F is not lsm and that $F \notin \langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle$ so, by Part 1 of the theorem, we have $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$.

By Lemma 6 there is a binary function $F_1 \in \langle F, \mathcal{B}_1^p\rangle$ that is not lsm so $F_1(0,0)F_1(1,1) < F_1(0,1)F_1(1,0)$. By Parts (iii) and (v) of Lemma 7, we either have $\mathrm{NEQ} \in \langle F, \mathcal{B}_1^p\rangle$ or $\mathrm{OR} \in \langle F, \mathcal{B}_1^p\rangle$. In the latter case, we are finished by Part (v) of Lemma **??**. In the former case, we are also finished since as is easily seen $\mathrm{OR} \in \langle \mathrm{IMP}, \mathrm{NEQ}\rangle$. □

6. Complexity-theoretic consequences

In order to explore the computational consequences of Theorem 10, we need to recall some definitions from computational complexity, specifically relating to approximate counting problems. For contextual material and proofs of any unsubstantiated claims made below, please refer to [10].

For our purposes, a counting problem is a function Π from instances w (encoded as a word over some alphabet Σ) to a number $\Pi(w) \in \mathbb{R}^{\geq 0}$. For example, w might encode an instance I of a counting CSP problem $\#\mathsf{CSP}(\Gamma)$, in which case $\Pi(w)$ would be the partition function $Z(I)$ associated with I. A *randomised approximation scheme* for Π is a randomised algorithm that takes an instance w and returns an approximation Y to $\Pi(w)$. The approximation scheme has a parameter $\varepsilon > 0$ which specifies the error tolerance. Since the algorithm is randomised, the output Y is a random variable depending on the "coin tosses" made by the algorithm. We require that, for every instance w and every $\varepsilon > 0$,

$$\Pr\left[e^{-\varepsilon}\Pi(w) \leq Y \leq e^{\varepsilon}\Pi(w)\right] \geq 3/4\,. \tag{4}$$

The randomised approximation scheme is said to be a *fully polynomial randomised approximation scheme*, or *FPRAS*, if it runs in time bounded by a polynomial in $|w|$ (the length of the word w) and ε^{-1}. See Mitzenmacher and Upfal [16, Definition 10.2]. Note that the quantity $3/4$ in Equation (4) could be changed to any value in the open interval $(1/2, 1)$ without changing the set of problems that have randomised approximation schemes [15, Lemma 6.1].

Suppose that Π_1 and Π_2 are functions from Σ^* to $\mathbb{R}^{\geq 0}$. An "approximation-preserving reduction" (AP-reduction) [10] from Π_1 to Π_2 gives a way to turn an FPRAS for Π_2 into an FPRAS for Π_1. Specifically, an *AP-reduction from Π_1 to Π_2* is a randomised

algorithm $\mathcal{A}$ for computing Π_1 using an oracle[1] for Π_2. The algorithm $\mathcal{A}$ takes as input a pair $(w, \varepsilon) \in \Sigma^* \times (0,1)$, and satisfies the following three conditions: (i) every oracle call made by $\mathcal{A}$ is of the form (v, δ), where $v \in \Sigma^*$ is an instance of Π_2, and $0 < \delta < 1$ is an error bound satisfying $\delta^{-1} \leq \text{poly}(|w|, \varepsilon^{-1})$; (ii) the algorithm $\mathcal{A}$ meets the specification for being a randomised approximation scheme for Π_1 (as described above) whenever the oracle meets the specification for being a randomised approximation scheme for Π_2; and (iii) the run-time of $\mathcal{A}$ is polynomial in $|w|$ and ε^{-1}. Note that the class of functions computable by an FPRAS is closed under AP-reducibility. Informally, AP-reducibility most liberal notion of reduction meeting this requirement. If an AP-reduction from Π_1 to Π_2 exists we write $\Pi_1 \leq_{\text{AP}} \Pi_2$. If $\Pi_1 \leq_{\text{AP}} \Pi_2$ and $\Pi_2 \leq_{\text{AP}} \Pi_1$ then we say that Π *and* Π_2 *are AP-interreducible*, and write $\Pi_1 =_{\text{AP}} \Pi_2$.

A word of warning about terminology. Subsequent to [10] the notation $\leq_{\text{AP}}$ has been used to denote a different time of approximation-preserving reduction which applies to optimisation problems. We will not study optimisation problems in this paper, so hopefully this will not cause confusion.

The complexity of approximating Boolean #CSPs in the unweighted case (i.e., where the functions in Γ have codomain $\{0,1\}$) was earlier studied by the final three authors [11]. Two counting problems played a special role there, and in earlier work in the complexity of approximate counting [10]. They also play a key role here.

Name: #SAT
Instance: A Boolean formula φ in conjunctive normal form.
Output: The number of satisfying assignments of φ.

Name: #BIS.
Instance: A bipartite graph B.
Output: The number of independent sets in B.

An FPRAS for #SAT would, in particular, have to decide with high probability between a formula having some satisfying assignments or having none. Thus #SAT cannot have an FPRAS unless NP = RP.[2] The same is true of any problem to which #SAT is AP-reducible. As far as we are aware, the complexity of approximating #BIS does not relate to any of the standard complexity theoretic assumptions, such as NP $\neq$ RP. Nevertheless, there is increasing empirical evidence that no FPRAS for #BIS exists, and we adopt this as a working hypothesis. Of course, this hypothesis implies that no #BIS-hard problem (problem to which #BIS is AP-reducible) admits an FPRAS.

Finally, a precise statement of the computational task we are interested in. A (weighted) #CSP problem is parameterised by a finite subset $\mathcal{F}$ of $\mathcal{B}^p$ and defined as follows.

Name: #CSP($\mathcal{F}$)
Instance: A pps-formula ψ consisting of a product of m atomic $\mathcal{F}$-formulas over n free variables $\boldsymbol{x}$. (Thus, ψ has no bound variables.)
Output: The value $\sum_{\boldsymbol{x} \in \{0,1\}^n} F_\psi(\boldsymbol{x})$ where F_ψ is the function defined by that formula.

[1]The reader who is not familiar with oracle Turing machines can just think of this as an imaginary (unwritten) subroutine for computing Π_2.

[2]The supposed FPRAS would provide a polynomial-time decision procedure for satisfiability this two-sided error; however, there is a standard trick for converting two-sided error to the one-sided error demanded by the definition of RP [19, Thm 10.5.9].

Officially, the input size $|w|$ is the length of the encoding of the instance. However, we shall take the size of a $\#\mathsf{CSP}(\mathcal{F})$ instance to be $n+m$, where n is the number of (free) variables and m is the number of constraints (atomic formulas). This is acceptable, as we are only concerned to measure the input size within a polynomial factor; moreover, we have restricted Γ to be finite, thereby avoiding the issue of how to the encode constraint functions $\mathcal{F}$. We typically denote an instance of $\#\mathsf{CSP}(\mathcal{F})$ by I and the output by $Z(I)$; by analogy with systems in statistical physics we refer to $Z(I)$ as the partition function.

Aside from simplifying the representation of problem instances, there is another, more important reason for decreeing that $\mathcal{F}$ is finite, namely, that it allows us to prove the following basic lemma relating functional clones and computational complexity. It is, of course, based on a similar result for classical decision CSPs.

Lemma 11. *Suppose that $\mathcal{F}$ is a finite subset of $\mathcal{B}^p$. If $F \in \langle\mathcal{F}\rangle_{\omega,p}$ then $\#\mathsf{CSP}(F,\mathcal{F}) \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F})$*

Proof. Let k be the arity of F. Let $\mathcal{M}$ be a TM which, on input $\varepsilon' > 0$, computes a k-ary pps-formula ψ over $\mathcal{F} \cup \mathrm{EQ}$ such that $\|F_\psi - F\|_\infty < \varepsilon'$. Consider an input (I,ε) where I is an instance of $\#\mathsf{CSP}(F,\mathcal{F})$ and ε is an accuracy parameter. The key idea of the proof is to construct an instance I' of $\#\mathsf{CSP}(\mathcal{F})$ by replacing each F-constraint in I with the set of constraints and extra (bound) variables in the formula ψ that is output by $\mathcal{M}$ with input ε'. After choosing a proper ε' the proof can be completed by a fairly straightforward computation. □

Theorem 12. *Suppose $\mathcal{F}$ is a finite subset of $\mathcal{B}^p$.*

- *If $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p\rangle$ then, for any finite subset S of $\mathcal{B}_1^p$, there is an FPRAS for $\#\mathsf{CSP}(\mathcal{F},S)$.*
- *Otherwise,*
  - *There is a finite subset S of $\mathcal{B}_1^p$ such that $\#\mathsf{BIS} \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F},S)$.*
  - *If there is a function $F \in \mathcal{F}$ such that $F \notin \mathsf{LSM}$ then there is a finite subset S of $\mathcal{B}_1^p$ such that $\#\mathsf{SAT} =_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F},S)$.*

Example 13. Let $F \in \mathcal{B}_2^p$ be the function defined by $F(0,0) = F(1,1) = \lambda$ and and $F(0,1) = F(1,0) = 1$, where $\lambda > 1$. Then, from Theorem 12, $\#\mathsf{CSP}(F,S)$ is $\#\mathsf{BIS}$-hard, for some set S of unary weights. (In fact, this counting CSP is also $\#\mathsf{BIS}$-easy.) Note that $\#\mathsf{CSP}(F,S)$ is nothing other than the ferromagnetic Ising model with an applied field. So we recover, with no effort, the main result of Goldberg and Jerrum's investigation of this model [13].

Example 14. If F is as before, but $\lambda \in (0,1)$, then $F \notin \mathsf{LSM}$ and Theorem 12 tells us that $\#\mathsf{CSP}(F,S)$ is $\#\mathsf{SAT}$-hard, for some set S of unary weights. This is a restatement of the well-known fact that the partition function of an antiferromagnetic Ising model is hard to compute, even approximately. (In fact, one could even dispense with the weights, but this fact cannot be read off directly from Theorem 12.)

References

[1] Rudolf Ahlswede and David E. Daykin. An inequality for the weights of two families of sets, their unions and intersections. *Z. Wahrsch. Verw. Gebiete*, 43(3):183–185, 1978.

[2] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35:22–35, 2004.

[3] Elmar Böhler, Steffen Reith, Henning Schnoor, and Heribert Vollmer. Bases for Boolean co-clones. *Inf. Process. Lett.*, 96(2):59–66, 2005.
[4] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.
[5] Jin-Yi Cai, Xi Chen, and Pinyan Lu. Non-negative weighted #CSPs: An effective complexity dichotomy. *CoRR*, abs/1012.5659, 2010.
[6] Jin-Yi Cai, Pinyan Lu, and Xia Mingji. Dichotomy for Holant* problems of Boolean domain. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1714–1728, 2011.
[7] David Cohen and Peter Jeavons. Chapter 8: The complexity of constraint languages. In Peter van Beek Francesca Rossi and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 245 – 280. Elsevier, 2006.
[8] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.
[9] Nadia Creignou, Phokion 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.
[10] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004. Approximation algorithms.
[11] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for Boolean #CSP. *Journal of Computer and System Sciences*, 76(3-4):267 – 277, 2010.
[12] Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. The complexity of weighted Boolean #CSP. *SIAM Journal on Computing*, 38:1970–1986, 2009.
[13] Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic Ising with local fields. *Combin. Probab. Comput.*, 16(1):43–61, 2007.
[14] Leslie Ann Goldberg and Mark Jerrum. Approximating the partition function of the ferromagnetic Potts model. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2010.
[15] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43(2-3):169–188, 1986.
[16] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, Cambridge, 2005. Randomized algorithms and probabilistic analysis.
[17] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.
[18] Donald M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26:305–321, 1978.
[19] Ingo Wegener. *Complexity Theory*. Springer-Verlag, Berlin, 2005. Exploring the limits of efficient algorithms, Translated from the German by Randall Pruim.
[20] Tomoyuki Yamakami. Approximate counting for complex-weighted Boolean constraint satisfaction problems. *CoRR*, abs/1007.0391, 2010.
[21] Stanislav Živný, David A. Cohen, and Peter G. Jeavons. The expressive power of binary submodular functions. *Discrete Appl. Math.*, 157(15):3347–3358, 2009.

Appendix A. Proof of Lemma 2

We start with proving two intermediate lemmas.

The functional clone $\langle\mathcal{F}\rangle$ generated by $\mathcal{F}$ is the set of all functions in $\mathcal{U}$ that can be represented by a pps-formula over $\mathcal{F}\cup\{\mathrm{EQ}\}$ where EQ is the binary equality function defined by $\mathrm{EQ}(x,x)=1$ and $\mathrm{EQ}(x,y)=0$ for $x\neq y$. We refer to the pps-formula as an "implementation" of the function.

Lemma 15. *If $G\in\langle\mathcal{F}\rangle$ then $\langle\mathcal{F},G\rangle=\langle\mathcal{F}\rangle$.*

Note that, to simplify notation, we write $\langle\mathcal{F},G\rangle$ in place of the more correct $\langle\mathcal{F}\cup\{G\}\rangle$. More generally, we shall drop set-brackets, replace the union symbol $\cup$ by a comma, and confuse a singleton set with the element it contains.

Proof of Lemma 15. Let $\mathcal{F}'=\mathcal{F}\cup\{\mathrm{EQ}\}$. Suppose that ψ is a pps-formula over $\mathcal{F}'\cup\{G\}$ given by

$$\psi=\sum_{v_{n+1},\ldots,v_{n+m}}\prod_{i=1}^{r}\varphi_i\prod_{j=1}^{s}\psi_j, \tag{5}$$

where $\{\varphi_i\}$ are atomic $\mathcal{F}'$-formulas and $\{\psi_j\}$ are atomic G-formulas in the variables V'. Then

$$F_\psi(\boldsymbol{x})=\sum_{\boldsymbol{y}\in D^m}\prod_{i=1}^{r}F_{\varphi_i}(\boldsymbol{x},\boldsymbol{y})\prod_{j=1}^{s}F_{\psi_j}(\boldsymbol{x},\boldsymbol{y}), \tag{6}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are assignments $\boldsymbol{x}:\{v_1,\ldots,v_n\}\to D$ and $\boldsymbol{y}:\{v_{n+1},\ldots,v_{n+m}\}\to D$. Now, since G is pps-definable over $\mathcal{F}'$, and each ψ_j is an atomic G-formula in the variables V', we can write each ψ_j as

$$\psi_j=\sum_{v_{\nu_j+1},\ldots,v_{\nu_j+\ell}}\prod_{k=1}^{t}\varphi_{j,k},$$

where ℓ is the number of bound variables used in the definition of ψ_j (ℓ is independent of j), $\nu_j=n+m+(j-1)\ell$ is the number of free variables plus the number of bound variables that are "used up" by $\psi_1,\ldots,\psi_{j-1}$, and each $\varphi_{j,k}$ is an atomic $\mathcal{F}'$-formula over the variables $V'\cup\{v_{\nu_j+1},\ldots,v_{\nu_j+\ell}\}$. We then get

$$\begin{aligned}F_\psi(\boldsymbol{x})&=\sum_{\boldsymbol{y}\in D^m}\prod_{i=1}^{r}F_{\varphi_i}(\boldsymbol{x},\boldsymbol{y})\prod_{j=1}^{s}\sum_{\boldsymbol{z}^j\in D^\ell}\prod_{k=1}^{t}F_{\varphi_{j,k}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}^j)\\&=\sum_{\boldsymbol{y}\in D^m}\sum_{\boldsymbol{z}^1\in D^\ell}\cdots\sum_{\boldsymbol{z}^s\in D^\ell}\prod_{i=1}^{r}F_{\varphi_i}(\boldsymbol{x},\boldsymbol{y})\prod_{j=1}^{s}\prod_{k=1}^{t}F_{\varphi_{j,k}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}^j),\end{aligned}$$

where each $\boldsymbol{z}^j$ is an assignment $\boldsymbol{z}^j:\{v_{\nu_j+1},\ldots,v_{\nu_j+\ell}\}\to D$. So

$$\varphi=\sum_{v_{n+1},\ldots,v_{\nu_s+\ell}}\prod_{i=1}^{r}\varphi_i\prod_{j=1}^{s}\prod_{k=1}^{t}\varphi_{j,k}$$

is a pps-formula over $\mathcal{F}'$ for the function F_ψ. □

The following lemma is an analogue of Lemma 15.

Lemma 16. *If $G \in \langle \mathcal{F} \rangle_\omega$ then $\langle \mathcal{F}, G \rangle_\omega = \langle \mathcal{F} \rangle_\omega$.*

Proof. Let $\mathcal{F}' = \mathcal{F} \cup \{\mathrm{EQ}\}$. Suppose that H is an a-ary function in $\langle \mathcal{F}, G \rangle_\omega$. Let S_H be a finite subset of $\mathcal{F}' \cup \{G\}$ such that the following is true: Given $\varepsilon > 0$, there exists an a-ary function $\widehat{H} \in \langle S_H \rangle$ such that $\|\widehat{H} - H\|_\infty < \varepsilon/2$. Let ψ be a pps-formula over S_H representing $\widehat{H}$. For any function $\widehat{G}$ with the same arity as G, denote by $\psi[G{:=}\widehat{G}]$ the formula obtained from ψ by replacing all occurrences of G by $\widehat{G}$. By continuity of the operators of pps-formulas, we know there exists $\delta > 0$ such that, for every function $\widehat{G}$ of the same arity as G, $\|\widehat{G} - G\|_\infty < \delta$ implies

$$\|F_{\psi[G:=\widehat{G}]} - F_\psi\|_\infty < \varepsilon/2.$$

This claim will be explicitly quantified in the proof of Lemma 2, but we don't need so much detail here. Of course, $\widehat{H} = F_\psi$ so for each such $\widehat{G}$ we have $\|F_{\psi[G:=\widehat{G}]} - \widehat{H}\|_\infty < \varepsilon/2$. Now let S_G be the finite subset of $\mathcal{F}'$ used to show that G is pps_ω-definable over $\mathcal{F}'$. Let $S = S_G \cup S_H \setminus \{G\} \subseteq \mathcal{F}'$. Choose a function $\widehat{G} \in \langle S_G \rangle$ satisfying $\|\widehat{G} - G\|_\infty < \delta$. Notice that $F_{\psi[G:=\widehat{G}]} \in \langle S \rangle$ (by Lemma 15), and

$$\|F_{\psi[G:=\widehat{G}]} - H\|_\infty \leq \|F_{\psi[G:=\widehat{G}]} - \widehat{H}\|_\infty + \|\widehat{H} - H\|_\infty < \varepsilon.$$

Since $\varepsilon > 0$ was arbitrary and S is a finite subset of $\mathcal{F}'$, we conclude that $H \in \langle \mathcal{F} \rangle_\omega$. □

Finally, we are ready to prove Lemma 2.

Lemma 17. *If $G \in \langle \mathcal{F} \rangle_\omega$ [or $G \in \langle \mathcal{F} \rangle_{\omega,p}$] then $\langle \mathcal{F}, G \rangle_\omega = \langle \mathcal{F} \rangle_{\omega,p}$ [resp., $\langle \mathcal{F}, G \rangle_{\omega,p} = \langle \mathcal{F} \rangle_{\omega,p}$].*

Proof. Let $\mathcal{F}' = \mathcal{F} \cup \{\mathrm{EQ}\}$. Suppose H is an a-ary function in $\langle \mathcal{F}', G \rangle_{\omega,p}$. Our goal is to specify a finite subset S of $\mathcal{F}'$ and to construct a TM $M_{H,S}$ with the following property: on input $\varepsilon > 0$, $M_{H,S}$ should compute an a-ary pps-formula φ over S such that $\|F_\varphi - H\|_\infty < \varepsilon$. The running time of $M_{H,S}$ should be at most a polynomial in $\log \varepsilon^{-1}$.

Let S_H be the finite subset of $\mathcal{F}' \cup \{G\}$ from the efficient pps_ω-definition of H over $\mathcal{F}' \cup \{G\}$. Given an input $\varepsilon/2$, the TM M_{H,S_H} computes an a-ary pps-formula ψ over S_H such that $\|F_\psi - H\|_\infty < \varepsilon/2$. Write ψ as in Equation (5) so F_ψ is writen as in Equation (6). Suppose that, for $j \in [s]$ and $\boldsymbol{y} \in \{0,1\}^m$, $\delta_{j,\boldsymbol{y}}(\boldsymbol{x})$ is a function of $\boldsymbol{x}$. Consider the expression

$$\Upsilon(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in D^m} \prod_{i=1}^{r} F_{\varphi_i}(\boldsymbol{x}, \boldsymbol{y}) \prod_{j=1}^{s} (F_{\psi_j}(\boldsymbol{x}, \boldsymbol{y}) + \delta_{j,\boldsymbol{y}}(\boldsymbol{x})) - \sum_{\boldsymbol{y} \in D^m} \prod_{i=1}^{r} F_{\varphi_i}(\boldsymbol{x}, \boldsymbol{y}) \prod_{j=1}^{s} F_{\psi_j}(\boldsymbol{x}, \boldsymbol{y}),$$

which can be expanded as

$$\Upsilon(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in D^m} \sum_{\emptyset \subset T \subseteq [s]} C_{\boldsymbol{y},T}(\boldsymbol{x}) \prod_{j \in T} \delta_{j,\boldsymbol{y}}(\boldsymbol{x}),$$

where

$$C_{\boldsymbol{y},T}(\boldsymbol{x}) = \prod_{i=1}^{r} F_{\varphi_i}(\boldsymbol{x},\boldsymbol{y}) \prod_{j\in[s]\setminus T} F_{\psi_j}(\boldsymbol{x},\boldsymbol{y}).$$

Let $C = \max_{\boldsymbol{x},\boldsymbol{y},T} |C_{\boldsymbol{y},T}(\boldsymbol{x})|$ and let $\delta = \varepsilon 2^{-(s+1)}|D|^{-m}C^{-1} < 1$.

Now let S_G be the finite subset of $\mathcal{F}'$ used to show that G is efficiently pps_ω-definable over $\mathcal{F}'$. Given the input δ, the TM M_{G,S_G} computes a pps-formula $\widehat{\psi}$ over S_G representing a function $F_{\widehat{\psi}}$ with the same arity as G such that $\|F_{\widehat{\psi}} - G\|_\infty < \delta$. Since each ψ_j is an atomic G-formula in the variables V', we may appropriately name the variables of $\widehat{\psi}$ to obtain a pps-formula $\widehat{\psi}_j$ over S_G such that $\|F_{\widehat{\psi}_j} - F_{\psi_j}\|_\infty < \delta$.

For $\boldsymbol{y} \in D^m$, let $\delta_{j,\boldsymbol{y}}(\boldsymbol{x}) = F_{\widehat{\psi}_j}(\boldsymbol{x},\boldsymbol{y}) - F_{\psi_j}(\boldsymbol{x},\boldsymbol{y})$ and note that $|\delta_{j,\boldsymbol{y}}(\boldsymbol{x})| \le \delta$. Let $S = S_G \cup S_H \setminus \{G\} \subseteq \mathcal{F}'$. Let ψ' be the formula over S formed from ψ by substituting each occurrence of ψ_j with $\widehat{\psi}_j$. Let φ be the pps-formula over S for the function $F_{\psi'}$ which is constructed as in the proof of Lemma 15. From the calculation above,

$$\begin{aligned}\|F_\varphi - F_\psi\|_\infty &= \|F_{\psi'} - F_\psi\|_\infty \\ &= \max_{\boldsymbol{x}} |F_{\psi'}(\boldsymbol{x}) - F_\psi(\boldsymbol{x})| \\ &= \max_{\boldsymbol{x}} |\Upsilon(\boldsymbol{x})| \\ &\le 2^s |D|^m C\delta = \varepsilon/2.\end{aligned}$$

Note that

$$\|F_\varphi - H\|_\infty \le \|F_\varphi - F_\psi\|_\infty + \|F_\psi - H\|_\infty < \varepsilon.$$

Thus, the formula φ is an appropriate output for our TM $M_{H,S}$.

Finally, let's check how long the computation takes. The running time of M_{H,S_H} is at most $\mathrm{poly}(\log \varepsilon^{-1})$. Since this machine outputs the formula ψ, we conclude that m and r and s are bounded from above by polynomials in $\log \varepsilon^{-1}$. Let Δ be the ceiling of the maximum absolute value of any function in S_H. Note that $C \le \Delta^{r+s}$. The running time of M_{G,S_G} is at most $\mathrm{poly}(\log(\delta^{-1}))$, which is at most a polynomial in $m + s + \log(C) + \log(\varepsilon^{-1})$ which is at most a polynomial in $\log(\varepsilon^{-1})$. Finally, the direct manipulation of the formulas that we did (renaming variables from $\widehat{\psi}$ to obtain $\widehat{\psi}_j$ and producing the pps-formula φ from ψ and the $\widehat{\psi}_j$ formulas) takes time at most polynomial in the size of ψ and $\widehat{\psi}$, which is at most a polynomial in $\log(\varepsilon^{-1})$. □

Appendix B. Proof of Lemma 3

Lemma 18. *Suppose $\mathcal{F} \subseteq \mathcal{B}$. Then*

$$\langle \{R_F \mid F \in \mathcal{F}\}\rangle_{\mathrm{R}} = \{R_F \mid F \in \langle\mathcal{F}\rangle\}.$$

Proof. Let $\mathcal{F}$ be a subset of $\mathcal{B}$. First, we must show that, for any $R \in \langle \{R_F \mid F \in \mathcal{F}\}\rangle_{\mathrm{R}}$, R is in $\{R_F \mid F \in \langle\mathcal{F}\rangle\}$.

Let ψ be the pp-formula over $\{R_F \mid F \in \mathcal{F}\} \cup \{\mathrm{EQ}\}$ that is used to represent R. Write ψ as

$$\psi = \exists\, v_{n+1}, \ldots, v_{n+m} \bigwedge_{j=1}^{s} R_{F_j}(v_{i(j,1)}, \ldots, v_{i(j,a_j)}),$$

where F_j is an arity-a_j function in $\mathcal{F} \cup \{\mathrm{EQ}\}$, and the index function $i(\cdot,\cdot)$ picks out an index in the range $[1, n+m]$, and hence a variable from $V' = \{v_1, \ldots, v_{n+m}\}$. Let ψ' be the pps-formula over $\mathcal{F} \cup \{\mathrm{EQ}\}$ given by

$$\psi' = \sum_{v_{n+1},\ldots,v_{n+m}} \prod_{j=1}^{s} F_j(v_{i(j,1)}, \ldots, v_{i(j,a_j)}).$$

Let $F' = F_{\psi'}$. Note that $F' \in \langle \mathcal{F} \rangle$ and that $R_{F'} = R$.

By reversing this construction, we can show that, for any $R \in \{R_F \mid F \in \langle \mathcal{F} \rangle\}$, R is in $\langle \{R_F \mid F \in \mathcal{F}\} \rangle_{\mathrm{R}}$. □

Appendix C. Proof of Lemma 5

Lemma 19. *If $\mathcal{F} \subseteq \mathsf{LSM}$ is any set of lsm functions then $\langle \mathcal{F} \rangle_\omega \subseteq \mathsf{LSM}$.*

Proof. We just need to show that each level in the definition of pps_ω-definable function preserves lsm: first that every atomic formula over $\mathcal{F} \cup \{\mathrm{EQ}\}$ defines an lsm function, then that a product of lsm functions is lsm, then that a summation of an lsm function is lsm, and finally that a limit of lsm functions is lsm. As we shall see below, only the third step is non-trivial, and it is covered by Lemma **??**.

First, note that the EQ is lsm, so every function in $\mathcal{F} \cup \{\mathrm{EQ}\}$ is lsm, An atomic formula $\varphi = G(v_{i_1}, \ldots, v_{i_a})$ defines a function $F_\varphi(\boldsymbol{x}) = G(x_{i_1}, \ldots, x_{i_a})$ which is lsm:

$$\begin{aligned} F_\varphi(\boldsymbol{x} \vee \boldsymbol{y}) F_\varphi(\boldsymbol{x} \wedge \boldsymbol{y}) &= G(x_{i_1} \vee y_{i_1}, \ldots, x_{i_a} \vee y_{i_a})\, G(x_{i_1} \wedge y_{i_1}, \ldots, x_{i_a} \wedge y_{i_a}) \\ &\geq G(x_{i_1}, \ldots, x_{i_a})\, G(y_{i_1}, \ldots, y_{i_a}) \\ &= F_\varphi(\boldsymbol{x}) F_\varphi(\boldsymbol{y}). \end{aligned}$$

Note that we do not need to assume that $i_1, \ldots, i_a$ are all distinct.

It is immediate that the product of two lsm functions (and hence the product of any number) is lsm. Thus the product $\prod_{j=1}^{s} F_{\varphi_j}$ appearing in (2) is lsm. Then, by Lemma **??**, the pps-definable function F_ψ itself in (2) is lsm.

Finally, we will show that any function that is approximated by lsm functions is lsm. Suppose that a function $F \in \mathcal{B}_n$ has the property that, for every $\varepsilon > 0$, there is an arity-n lsm function $\widehat{F}$ satisfying

$$\|\widehat{F} - F\|_\infty = \max_{\boldsymbol{x} \in \{0,1\}^a} |\widehat{F}(\boldsymbol{x}) - F(\boldsymbol{x})| < \varepsilon.$$

We wish to show that F is lsm. Let $F_{\max} = \max_{\boldsymbol{x}} F(\boldsymbol{x})$. Suppose for contradiction that F is not lsm, so there is a $\delta > 0$ and $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ such that

$$F(\boldsymbol{x} \vee \boldsymbol{y}) F(\boldsymbol{x} \wedge \boldsymbol{y}) \leq F(\boldsymbol{x}) F(\boldsymbol{y}) - \delta.$$

Let $\varepsilon > 0$ be sufficiently small that $\varepsilon \max(F_{\max}, 1)$ is tiny compared to δ. Then

$$\begin{aligned}
\widehat{F}(\boldsymbol{x} \vee \boldsymbol{y})\widehat{F}(\boldsymbol{x} \wedge \boldsymbol{y}) &\leq (F(\boldsymbol{x} \vee \boldsymbol{y}) + \varepsilon)(F(\boldsymbol{x} \wedge \boldsymbol{y}) + \varepsilon) \\
&\leq F(\boldsymbol{x} \vee \boldsymbol{y})F(\boldsymbol{x} \wedge \boldsymbol{y}) + 2\varepsilon F_{\max} + \varepsilon^2 \\
&\leq F(\boldsymbol{x})F(\boldsymbol{y}) - \delta + 2\varepsilon F_{\max} + \varepsilon^2 \\
&\leq (\widehat{F}(\boldsymbol{x}) + \varepsilon)(\widehat{F}(\boldsymbol{y}) + \varepsilon) - \delta + 2\varepsilon F_{\max} + \varepsilon^2 \\
&\leq \widehat{F}(\boldsymbol{x})\widehat{F}(\boldsymbol{y}) + 2\varepsilon(F_{\max} + \varepsilon) - \delta + 2\varepsilon F_{\max} + 2\varepsilon^2 \\
&< \widehat{F}(\boldsymbol{x})\widehat{F}(\boldsymbol{y}),
\end{aligned}$$

so $\widehat{F}$ is not lsm, giving a contradiction. □

Appendix D. Proof of Lemma 7

When we write a function $F \in \mathcal{B}_2$, we will identify the arguments by writing $F(x_1, x_2)$. We may represent F by a 2×2 matrix

$$M(F) \;=\; \begin{bmatrix} F(0,0) & F(0,1) \\ F(1,0) & F(1,1) \end{bmatrix} \;=\; \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix},$$

say, with rows indexed by $x_1 \in \{0,1\}$ and columns by $x_2 \in \{0,1\}$. We will assume $f_{01} \geq f_{10}$, since otherwise we may consider the function F^T, such that $F^T(x_1, x_2) = F(x_2, x_1)$, represented by the matrix $M(F)^T$. Clearly $\langle F^T \rangle = \langle F \rangle$.

If U is a unary function, we will write $U = (U(0), U(1)) = (u_0, u_1)$, say. Then we have

$$M\big(U(x_1)F(x_1,x_2)\big) = \begin{bmatrix} u_0 f_{00} & u_0 f_{01} \\ u_1 f_{10} & u_1 f_{11} \end{bmatrix}, \quad M\big(U(x_2)F(x_1,x_2)\big) = \begin{bmatrix} u_0 f_{00} & u_1 f_{01} \\ u_0 f_{10} & u_1 f_{11} \end{bmatrix},$$

where both $U(x_1)F(x_1, x_2)$ and $U(x_2)F(x_1, x_2)$ are clearly in $\langle F, U \rangle$.

If $F_1, F_2 \in \mathcal{B}_2$, then $M(F_1)M(F_2) = M(F)$, where $F \in \langle F_1, F_2 \rangle$ is such that

$$F(x_1, x_2) \;=\; \sum_{y=0}^{1} F_1(x_1, y)F_2(y, x_2).$$

Lemma 20. *Let $F \in \mathcal{B}_2^p$. Assuming $f_{01} \geq f_{10}$,*

(i) if $f_{00}f_{11} = f_{01}f_{10}$, then $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathcal{B}_1^p \rangle_{\omega,p}$;
(ii) if $f_{01}, f_{10} = 0$ and $f_{00}, f_{11} > 0$, then $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathcal{B}_1^p \rangle_{\omega,p}$;
(iii) if $f_{00}, f_{11} = 0$ and $f_{01}, f_{10} > 0$, then $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle_{\omega,p}$;
(iv) if $f_{00}, f_{01}, f_{11} > 0$ and $f_{00}f_{11} > f_{01}f_{10}$, then $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{IMP}, \mathcal{B}_1^p \rangle_{\omega,p}$;
(v) otherwise, $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{OR}, \mathcal{B}_1^p \rangle_{\omega,p}$.

Proof. To prove $\langle F_1, \mathcal{B}_1^p \rangle_{\omega,p} = \langle F_2, \mathcal{B}_1^p \rangle_{\omega,p}$, it is sufficient to show that $F_2 \in \langle F_1, \mathcal{B}_1^p \rangle_{\omega,p}$ and $F_1 \in \langle F_2, \mathcal{B}_1^p \rangle_{\omega,p}$. We will verify this in each of the five cases.

(i) Suppose $f_{00}f_{11} = f_{01}f_{10}$. If $f_{00}, f_{01} = 0$, then

$$F(x_1, x_2) = U_1(x_1)U_2(x_2)$$

with $U_1 = (0, 1)$ and $U_2 = (f_{10}, f_{11})$. Similarly if $f_{00}, f_{10} = 0$, $f_{01}, f_{11} = 0$, or $f_{10}, f_{11} = 0$. In the remaining case $f_{00}, f_{01}, f_{10}, f_{11} > 0$. Then choose $U_1 = (1, f_{10}/f_{00})$, $U_2 = (f_{00}, f_{01})$. In all cases $F \in \langle U_1, U_2 \rangle$, so $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathcal{B}_1^p \rangle_{\omega,p}$.

(ii) if $f_{01}, f_{10} = 0$ and $f_{00}, f_{11} > 0$, then $F(x_1, x_2) = U(x_1)\mathrm{EQ}(x_1, x_2)$, where $U = (f_{00}, f_{11})$, so $F \in \langle U \rangle$. Hence $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathcal{B}_1^p \rangle_{\omega,p}$.

(iii) If $f_{00}, f_{11} = 0$ and $f_{01}, f_{10} > 0$, then $F(x_1, x_2) = U(x_1)\mathrm{NEQ}(x_1, x_2)$, where $U = (f_{01}, f_{10})$, so $F \in \langle \mathrm{NEQ}, U \rangle$. Similarly $\mathrm{NEQ}(x_1, x_2) = U'(x_1)F(x_1, x_2)$, where $U' = (1/f_{01}, 1/f_{10})$, so $\mathrm{NEQ} \in \langle F, U' \rangle$. Hence $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle_{\omega,p}$.

(iv) If $f_{00}, f_{01}, f_{11} > 0$, $f_{00}f_{11} > f_{01}f_{10}$, we can apply unary weights U_1, U_2, where $U_1 = (1/f_{00}, f_{01}/f_{00}f_{11})$, $U_2 = (1, f_{00}/f_{01})$, to implement $\mathrm{IMP}_\alpha(x_1, x_2) = U_1(x_1) \times U_2(x_2)F(x_1, x_2)$, where

$$M(\mathrm{IMP}_\alpha) \;=\; \begin{bmatrix} 1 & 1 \\ \alpha & 1 \end{bmatrix},$$

where $\alpha = f_{01}f_{10}/f_{00}f_{11} < 1$. Then we have $\mathrm{IMP}_\alpha \in \langle F, U_1, U_2 \rangle$. Note that $\mathrm{IMP}_0 = \mathrm{IMP}$. If $\alpha > 0$, consider the function IMP_α^k, with matrix

$$M(\mathrm{IMP}_\alpha^k) \;=\; \begin{bmatrix} 1 & 1 \\ \alpha^k & 1 \end{bmatrix}.$$

Note that IMP_α^k can be implemented as $\mathrm{IMP}_\alpha^k(x_1, x_2) = U_1^k(x_1)U_2^k(x_2)F^k(x_1, x_2)$, by taking k copies of U_1, U_2 and F. Since $\alpha < 1$, we see that $\lim_{k\to\infty} \mathrm{IMP}_\alpha^k = \mathrm{IMP}_0 = \mathrm{IMP}$. Moreover, the limit is efficient, since $\|\mathrm{IMP} - \mathrm{IMP}_\alpha^k\| < \varepsilon$ if $k = O(\log \varepsilon^{-1})$, and so an ε-approximation to IMP can be computed in $O(\log \varepsilon^{-1})$ time. Hence $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$.

Note that "powering" limits like that used here will be employed below without further discussion of their efficiency.

Conversely, from IMP, we first implement IMP_α. If $\alpha = 0$, we do nothing. Otherwise, we use unary weights U_1, U_2 such that $U_1 = (1/\alpha - 1, 1)$, $U_2 = (\alpha, 1)$, to implement $F_1(x_1, x_2) = U_1(x_1)U_2(x_2)\mathrm{IMP}(x_2, x_1)$, where

$$M(F_1) \;=\; \begin{bmatrix} 1-\alpha & 0 \\ \alpha & 1 \end{bmatrix}.$$

Then we have $M(\mathrm{IMP}_\alpha) = M(\mathrm{IMP})M(F_1)$, so $\mathrm{IMP}_\alpha \in \langle \mathrm{IMP}, U_1, U_2 \rangle$. Now we can recover $F(x_1, x_2) = U_3(x_1)U_4(x_2)\mathrm{IMP}_\alpha(x_1, x_2)$, where $U_3 = (f_{00}, f_{00}f_{11}/f_{01})$, $U_4 = (1, f_{01}/f_{00})$, so we have $F \in \langle \mathrm{IMP}, U_1, U_2, U_3, U_4 \rangle$. Hence $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{IMP}, \mathcal{B}_1^p \rangle_{\omega,p}$.

(v) The remaining cases are (a) $f_{01}, f_{10}, f_{11} > 0$, $f_{00}f_{11} < f_{01}f_{10}$ and (b) $f_{00}, f_{01}, f_{10} > 0$, $f_{11} = 0$.

First, we deal with part (a): If $f_{01}, f_{10}, f_{11} > 0$ and $f_{00}f_{11} < f_{01}f_{10}$, we apply unary weights U_1, U_2, where $U_1 = (f_{11}/f_{01}, 1)$, $U_2 = (1/f_{10}, 1/f_{11})$, to implement $\mathrm{OR}_\alpha(x_1, x_2) = U_1(x_1)U_2(x_2)F(x_1, x_2)$, where $\alpha = f_{00}f_{11}/f_{01}f_{10} < 1$, and

$$M(\mathrm{OR}_\alpha) = \begin{bmatrix} \alpha & 1 \\ 1 & 1 \end{bmatrix}$$

If $\alpha = 0$, $\mathrm{OR}_0 = \mathrm{OR}$, so we have $\mathrm{OR} \in \langle F, \mathcal{B}_1^p \rangle$. Otherwise $\lim_{k\to\infty} \mathrm{OR}_\alpha^k = \mathrm{OR}_0 = \mathrm{OR}$, so we have $\mathrm{OR} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$.

Conversely, from OR, we first express NEQ. Use the unary function $U = (2, 1/2)$ to implement $F_1 = U(x_1)U(x_2)\mathrm{OR}(x_1, x_2)$, where

$$M(F_1) = \begin{bmatrix} 0 & 1 \\ 1 & 1/4 \end{bmatrix}.$$

Then $\lim_{k\to\infty} F_1^k = \mathrm{NEQ}$. so $\mathrm{NEQ} \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Now we observe that $M(\mathrm{IMP}) = M(\mathrm{NEQ})M(\mathrm{OR})$, so $\mathrm{IMP} \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Now we have $\mathrm{IMP}_\alpha \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$, as in (iv) above. Then $M(\mathrm{OR}_\alpha) = M(\mathrm{NEQ})M(\mathrm{IMP}_\alpha)$, so $\mathrm{OR}_\alpha \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Now we can reverse the transformation from F to OR_α to recover F.

Now, we consider part (b): If $f_{00}, f_{01}, f_{10} > 0$ and $f_{11} = 0$, we apply unary weights U_1, U_2, where $U_1 = (1/f_{00}, 1/f_{10})$, $U_2 = (1, f_{00}/f_{01})$, to implement $\mathrm{NAND}(x_1, x_2) = U_1(x_1)U_2(x_2)F(x_1, x_2)$, where

$$M(\mathrm{NAND}) = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix},$$

so we have $\mathrm{NAND} \in \langle F, \mathcal{B}_1^p\rangle$. We now use unary weight $U = (1/2, 2)$ to implement $F_1(x_1) = U(x_1)U(x_1)\mathrm{NAND}(x_1, x_2)$ with

$$M(F_1) = \begin{bmatrix} 1/4 & 1 \\ 1 & 0 \end{bmatrix}.$$

Then we have $\lim_{k\to\infty} F_1^k = \mathrm{NEQ}$, so again $\mathrm{NEQ} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$. Then we observe that $M(\mathrm{OR}) = M(\mathrm{NEQ})M(\mathrm{NAND})M(\mathrm{NEQ})$, so $\mathrm{OR} \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$.

Conversely, from OR, we have $\mathrm{NEQ} \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$ from the above. Then we have $M(\mathrm{NAND}) = M(\mathrm{NEQ})M(\mathrm{OR})M(\mathrm{NEQ})$, so $\mathrm{NAND} \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$. Now we reverse the transformation above from F to NAND to recover F. Thus $F \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$.

Finally we prove that $\langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p} = \mathcal{B}^p$. Suppose $F \in \mathcal{B}_n^p$. Suppose $x_1, \ldots, x_n$ are variables. For each $A \subseteq [n]$, let $\boldsymbol{x}(A)$ be the assignment to $x_1, \ldots, x_n$ in which $x_i = 1$ if $i \in A$ and $x_i = 0$ otherwise. Let $\mathcal{A} = \{A \mid F(\boldsymbol{x}(A)) > 0\}$, and let $\mu = \min_{A\in\mathcal{A}} F(\boldsymbol{x}(A))$. For any $A \subseteq [n]$, let $u_A \in \mathcal{B}_1^p$ be the function such that $u_A(0) = 1$ and $u_A(1) = 2F(\boldsymbol{x}(A))/\mu - 1 \geq 1$. Note that every function u_A is in $\mathcal{B}_1^p$ and we have $\mathrm{IMP}, \mathrm{NAND} \in \langle \mathrm{OR}, \mathcal{B}_1^p\rangle_{\omega,p}$, from the proof of Lemma 7.

Our goal will be to show that there is a finite subset S_F of $\{\mathrm{IMP}, \mathrm{NAND}\} \cup \mathcal{B}_1^p$ and a TM M_{F,S_F} with the following property: on input $\varepsilon > 0$, M_{F,S_F} computes an arity-n pps-formula ψ over S_F such that $\|F_\psi - F\|_\infty < \varepsilon$. The running time of M_{F,S_F} should be at most a polynomial in $\log \varepsilon^{-1}$. To define S_F, we will use two unary functions U_1 and U_2 (both of which are actually constant functions). We define these by $U_1(0) = U_1(1) = 1/2$ and $U_2(0) = U_2(1) = \mu/2$. Then $S_F = \{\mathrm{IMP}, \mathrm{NAND}, U_1, U_2\} \cup \bigcup_{A\in\mathcal{A}} \{u_A\}$.

Let $V = \{v_1, \ldots, v_n\}$. For $A \in \mathcal{A}$, introduce a new variable z_A. Let $V'' = V \cup \{z_A \mid A \in \mathcal{A}\}$. Let

$$\psi_1 = \sum_{V''} \prod_{A\in\mathcal{A}} u_A(z_A) \prod_{i\in A} \mathrm{IMP}(z_A, x_i) \prod_{i\notin A} \mathrm{NAND}(z_A, x_i).$$

For every $A \in \mathcal{A}$ the assignment $\boldsymbol{x}(A)$ can be extended in two ways (both with $z_A = 0$ and with $z_A = 1$) to satisfy

$$\prod_{i \in A} \mathrm{IMP}(z_A, x_i) \prod_{i \notin A} \mathrm{NAND}(z_A, x_i). \tag{7}$$

Any other assignment $\boldsymbol{x}$ can be extended in only one way ($z_A = 0$) to satisfy (7). So if $A \in \mathcal{A}$ then

$$F_{\psi_1}(\boldsymbol{x}(A)) = (2F(\boldsymbol{x}(A))/\mu - 1) + 1 = 2F(\boldsymbol{x}(A))/\mu.$$

On the other hand, if $A \notin \mathcal{A}$ then

$$F_{\psi_1}(\boldsymbol{x}(A)) = 1.$$

We have shown that $F_{\psi_1} \in \langle S_F \rangle$. Let us now define

$$\psi_2 = \sum_{V''} \prod_{A \in \mathcal{A}} \prod_{i \in A} \mathrm{IMP}(z_A, x_i) \prod_{i \notin A} \mathrm{NAND}(z_A, x_i).$$

As before, for every $A \in \mathcal{A}$ the assignment $\boldsymbol{x}(A)$ can be extended in two ways ($z_A = 0$ and $z_A = 1$) to satisfy (7), and any other assignment $\boldsymbol{x}$ can be extended in only one way ($z_A = 0$) to satisfy it. So

$$F_{\psi_2}(\boldsymbol{x}(A)) = 2 \quad (A \in \mathcal{A}), \qquad F_{\psi_2}(\boldsymbol{x}(A)) = 1 \quad (A \notin \mathcal{A}).$$

Thus $F_{\psi_2} \in \langle S_F \rangle$. Now define F_3 by $F_3(\boldsymbol{x}(A)) = U_1(x_1) F_{\psi_2}(\boldsymbol{x}(A))$, so $F_3 \in \langle S_F \rangle$, where

$$F_3(\boldsymbol{x}(A)) = 1 \quad (A \in \mathcal{A}), \qquad F_{\psi_2}(\boldsymbol{x}(A)) = 1/2 \quad (A \notin \mathcal{A}).$$

Now $\lim_{k \to \infty} F_3^k = F_0$, where

$$F_0(\boldsymbol{x}(A)) = 1 \quad (A \in \mathcal{A}), \qquad F_{\psi_2}(\boldsymbol{x}(A)) = 0 \quad (A \notin \mathcal{A}),$$

and thus $F_0 \in \langle S_F \rangle_{\omega,p}$.

Note that $F_0 = R_F$, the underlying relation of F. Now define $F_4 = F_{\psi_1} F_0$, so that

$$F_4(\boldsymbol{x}(A)) = 2F(\boldsymbol{x}(A))/\mu \quad (A \in \mathcal{A}), \qquad F_4(\boldsymbol{x}(A)) = 0 \quad (A \notin \mathcal{A}),$$

Thus, by Lemma 2, $F_4 \in \langle S_F \rangle_{\omega,p}$. Now define F_5 by $F_5(\boldsymbol{x}(A)) = U_2(x_1) F_4(\boldsymbol{x}(A))$, so $F_5 \in \langle S_F \rangle_{\omega,p}$, where

$$F_5(\boldsymbol{x}(A)) = F(\boldsymbol{x}(A)) \quad (A \in \mathcal{A}), \qquad F_5(\boldsymbol{x}(A)) = 0 \quad (A \notin \mathcal{A}).$$

Since $F_5 = F$, the proof is complete. □

Appendix E. Post's lattice

Appendix F. Proof of Lemma 11

Lemma 21. *Suppose that $\mathcal{F}$ is a finite subset of $\mathcal{B}^p$. If $F \in \langle \mathcal{F} \rangle_{\omega,p}$ then $\#\mathsf{CSP}(F, \mathcal{F}) \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F})$*

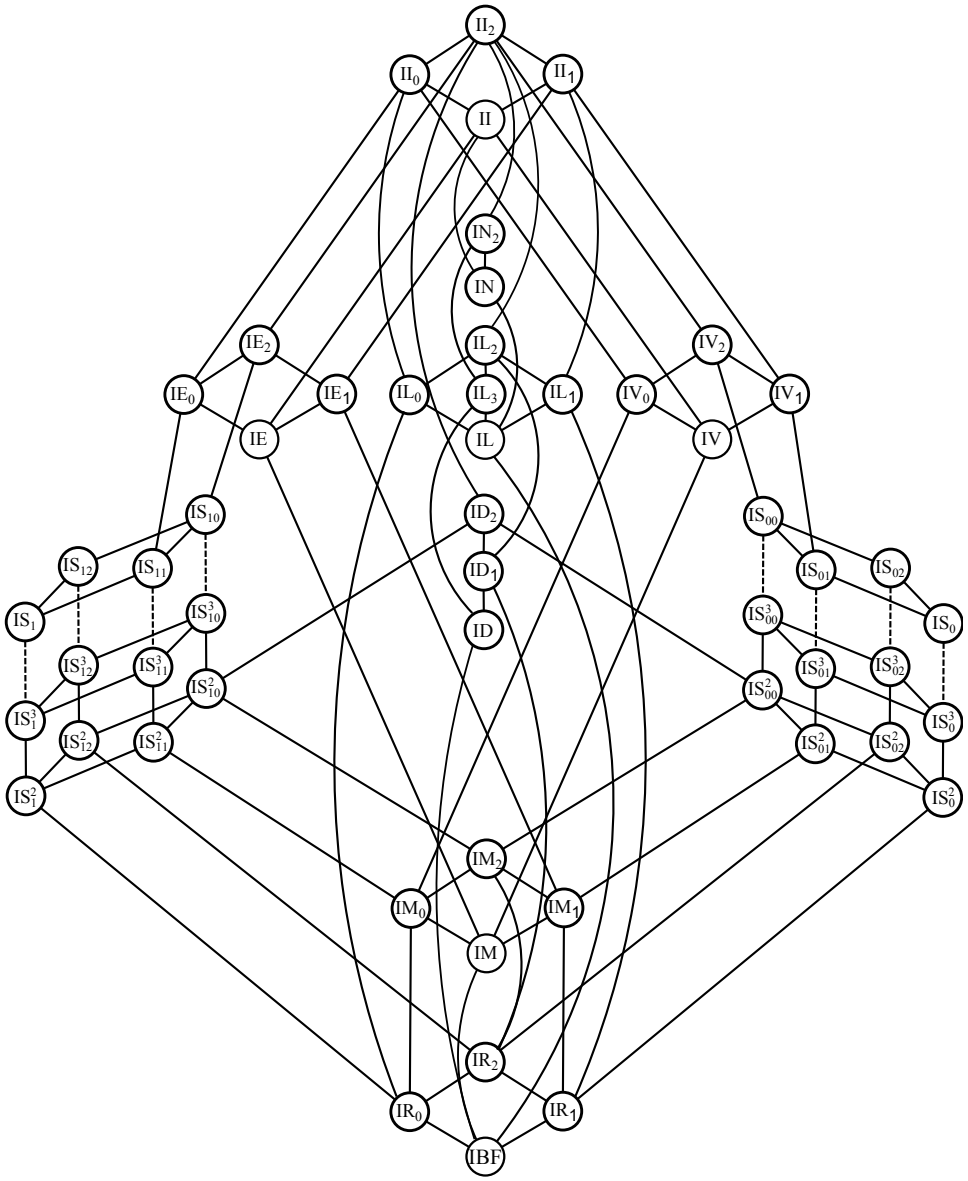

FIGURE 1. Post's lattice from [2, Figure 2].

Proof. Let k be the arity of F. Let $\mathcal{M}$ be a TM which, on input $\varepsilon' > 0$, computes a k-ary pps-formula ψ over $\mathcal{F} \cup \mathrm{EQ}$ such that $\|F_\psi - F\|_\infty < \varepsilon'$. We can assume without loss of generality that no function in $\{F\} \cup \mathcal{F}$ is identically zero (otherwise every #CSP instance using this function has partition function 0). Let $\mu_{\max}$ be the maximum value in the range of F and let $\mu_{\min}$ be the minimum of 1 and the minimum nonzero value in the range of F. Similarly, let S be the set of nonzero values in the range of functions in $\mathcal{F} \cup \{\mathrm{EQ}\}$. Let $\nu_{\max}$ be the maximum value in S and let $\nu_{\min}$ be the minimum of 1 and the minimum value in S.

Consider an input (I, ε) where I is an instance of $\#\mathsf{CSP}(F, \mathcal{F})$ and ε is an accuracy parameter. Suppose that I has n variables, m F-constraints, and m' other constraints. We can assume without loss of generality that $m > 0$ (otherwise, I is an instance of $\#\mathsf{CSP}(\mathcal{F})$).

The key idea of the proof is to construct an instance I' of $\#\mathsf{CSP}(\mathcal{F})$ by replacing each F-constraint in I with the set of constraints and extra (bound) variables in the formula ψ that is output by $\mathcal{M}$ with input ε'. We determine how small to make ε' in terms of the following quantities. Let

$$\begin{aligned} A &= \frac{4m}{\mu_{\min}} 2^n \mu_{\max}^m \nu_{\max}^{m'} \\ B &= 2^n (\mu_{\max} + 1)^{m-1} \nu_{\max}^{m'} \\ C &= \mu_{\min}^m \nu_{\min}^{m'}. \end{aligned}$$

Let $\varepsilon' = \frac{\varepsilon}{4} \frac{C}{A+B}$. The time needed to construct ψ for a given $\varepsilon' > 0$ is at most $\mathrm{poly}(\log(\varepsilon'^{-1}))$, which is at most a polynomial in n, m, m' and ε^{-1}, as required by the definition of AP-reduction. We shall see that $(I', \varepsilon/2)$ is the sought-for instance/tolerance pair required by our reduction.

Let I_ψ be the instance formed from I by replacing every F-constraint with an F_ψ-constraint. Note that $Z(I_\psi) = Z(I')$, since I', an instance of $\#\mathsf{CSP}(\mathcal{F})$, is an implementation of I_ψ. We want to show that if an oracle produces a sufficiently accurate approximation to $Z(I')$ (and hence to $Z(I_\psi)$) then we can deduce a sufficiently accurate approximation to $Z(I)$. Observe that the definition of FPRAS allows no margin of error when $Z(I) = 0$, and our reduction must give the correct result, namely 0, in this case. Therefore we need to treat separately the cases $Z(I) = 0$ and $Z(I) > 0$. We will show that

$$Z(I) = 0 \quad \text{implies} \quad Z(I_\psi) < C/3, \tag{8}$$

and

$$Z(I) > 0 \quad \text{implies} \quad Z(I_\psi) > 2C/3; \tag{9}$$

moreover, in the latter case,

$$e^{-\varepsilon/2} Z(I) \le Z(I_\psi) \le e^{\varepsilon/2} Z(I). \tag{10}$$

These estimates are enough to ensure correctness of the reduction. For a call to an oracle for $\#\mathsf{CSP}(\mathcal{F})$ with instance I' and accuracy parameter $\varepsilon/2$ would return a result in the range $[e^{-\varepsilon/2} Z(I_\psi), e^{\varepsilon/2} Z(I_\psi)]$ with high probability. Observe that this estimate is sufficient to distinguish between cases (8) and (9). In the former case, we are able to return the exact result, namely 0. In the latter case, we return the result given by the oracle, which by (10) satisfies the conditions for an FPRAS.

To establish (8–10), let Y' be the set of assignments to the variables of instance I which make a non-zero contribution to $Z(I)$ and let Y'' be the remaining assignments to the variables of instance I. Let $Z'(I_\psi)$ be the contribution to $Z(I_\psi)$ due to assignments in Y' and $Z''(I_\psi)$ be the contribution to $Z(I_\psi)$ due to assignments in Y'' (so $Z(I_\psi) = Z'(I_\psi) + Z''(I_\psi)$). We can similarly write $Z(I) = Z'(I) + Z''(I)$, though of course $Z''(I) = 0$.

First, note that if $|F_\psi(\boldsymbol{x}) - F(\boldsymbol{x})| \leq \varepsilon'$ and $F(\boldsymbol{x}) > 0$ then

$$|F_\psi(\boldsymbol{x})/F(\boldsymbol{x}) - 1| \leq \varepsilon'/F(\boldsymbol{x}) \leq \varepsilon'/\mu_{\min},$$

so

$$e^{-2\varepsilon'/\mu_{\min}} \leq \frac{F_\psi(\boldsymbol{x})}{F(\boldsymbol{x})} \leq e^{2\varepsilon'/\mu_{\min}}.$$

We conclude that

$$e^{-2\varepsilon' m/\mu_{\min}} Z'(I) \leq Z'(I_\psi) \leq e^{2\varepsilon' m/\mu_{\min}} Z'(I),$$

so

$$|Z'(I) - Z'(I_\psi)| \leq \frac{4\varepsilon' m}{\mu_{\min}} Z(I) \leq \varepsilon' A.$$

Furthermore,

$$|Z''(I) - Z''(I_\psi)| = Z''(I_\psi) \leq \varepsilon' B.$$

Here we use $\|F_\psi - F\|_\infty < \varepsilon' < 1$; the "+1" in the definition of B absorbs the discrepancy between F_ψ and F. Combining these two inequalities yields

$$|Z(I) - Z(I_\psi)| \leq \varepsilon'(A + B) \leq \frac{\varepsilon C}{4}. \tag{11}$$

Now, $Z(I) > 0$ implies $Z(I) \geq C$, and hence (8) and (9) follow directly from (11). If $Z(I) > 0$ we further have

$$\left| \frac{Z(I_\psi)}{Z(I)} - 1 \right| \leq \frac{\varepsilon'(A+B)}{Z(I)} \leq \frac{\varepsilon'(A+B)}{C} \leq \varepsilon/3.$$

This establishes (10) and completes the verification of the reduction. □

Appendix G. Proof of Theorem 12

Theorem 22. *Suppose $\mathcal{F}$ is a finite subset of $\mathcal{B}^p$.*

- *If $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$ then, for any finite subset S of $\mathcal{B}_1^p$, there is an FPRAS for $\#\mathsf{CSP}(\mathcal{F}, S)$.*
- *Otherwise,*
  - *There is a finite subset S of $\mathcal{B}_1^p$ such that $\#\mathsf{BIS} \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F}, S)$.*
  - *If there is a function $F \in \mathcal{F}$ such that $F \notin \mathsf{LSM}$ then there is a finite subset S of $\mathcal{B}_1^p$ such that $\#\mathsf{SAT} =_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F}, S)$.*

Proof. First, suppose that $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$. Let S be a finite subset of $\mathcal{B}_1^p$, Given an m-constraint input I of $\#\mathsf{CSP}(\mathcal{F}, S)$ and an accuracy parameter ε, we first approximate each arity-k function $F \in \mathcal{F}$ used in I with a function $\widehat{F} : \{0,1\}^k \to \mathbb{Q}^{\geq 0}$ such that $R_{\widehat{F}} = R_F$, and for every $\boldsymbol{x}$ for which $F(\boldsymbol{x}) > 0$,

$$e^{-\varepsilon/m} \leq \frac{\widehat{F}(\boldsymbol{x})}{F(\boldsymbol{x})} \leq e^{\varepsilon/m}. \tag{12}$$

Let $\widehat{\mathcal{F}} = \{\widehat{F} \mid F \in \mathcal{F}\}$ and let $\widehat{I}$ be the instance of $\#\mathsf{CSP}(\widehat{\mathcal{F}}, S)$ formed from I by replacing each F-constraint with $\widehat{F}$. [12, Theorem 4] gives a polynomial-time algorithm for computing the partition function $Z(\widehat{I})$, which satisfies

$$e^{-\varepsilon} Z(I) \leq Z(\widehat{I}) \leq e^{\varepsilon} Z(I). \tag{13}$$

Second, suppose that $\mathcal{F} \not\subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$. By Theorem 10, IMP $\in \langle \mathcal{F}, \mathcal{B}_1^p \rangle_{\omega,p}$. By Observation 1, there is a finite subset S of $\mathcal{B}_1^p$ such that IMP $\in \langle \mathcal{F}, S \rangle_{\omega,p}$. Thus, $\#\mathsf{CSP}(\mathrm{IMP}) \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathcal{F}, S)$, by Lemma 11 However, $\#\mathsf{BIS} =_{\mathrm{AP}} \#\mathsf{CSP}(\mathrm{IMP})$ by [11, Theorem 3].

Finally, suppose that there is a function $F \in \mathcal{F}$ such that $F \notin \mathsf{LSM}$. By Theorem 10, $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \mathcal{B}^p$ so OR $\in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$. By Observation 1, there is a finite subset S of $\mathcal{B}_1^p$ such that OR $\in \langle F, S \rangle_{\omega,p}$, so by Lemma 11, $\#\mathsf{CSP}(\mathrm{OR}) \leq_{\mathrm{AP}} \#\mathsf{CSP}(F, S)$. However, by [11, Lemma 7] $\#\mathsf{SAT} \leq_{\mathrm{AP}} \#\mathsf{CSP}(\mathrm{OR})$. To see that $\#\mathsf{CSP}(\mathcal{F}, S) \leq_{\mathrm{AP}} \#\mathsf{SAT}$, let I be an m-constraint instance of $\#\mathsf{CSP}(\mathcal{F}, S)$. For each function $G \in \mathcal{F} \cup S$, define $\widehat{G}$ as in (12). Let $\widehat{I}$ be the instance of $\#\mathsf{CSP}(\{\widehat{G} \mid G \in \mathcal{F} \cup S\})$ formed from I by replacing each G-constraint with a $\widehat{G}$-constraint. Equation (13) holds, as above. Furthermore, [12, Theorem 4] shows that $\#\mathsf{CSP}(\{\widehat{G} \mid G \in \mathcal{F} \cup S\})$ is in $\mathsf{FP}^{\#\mathsf{P}}$, so can be AP-reduced to $\#\mathsf{SAT}$. □