Log-supermodular functions, functional clones and counting CSPs*

Andrei A. Bulatov[1], Martin Dyer[2], Leslie Ann Goldberg[3], and Mark Jerrum[4]

1 **School of Computing Science, Simon Fraser University**
Burnaby, Canada
abulatov@cs.sfu.edu
2 **School of Computing, University of Leeds**
Leeds LS2 9JT, UK
dyer@comp.leeds.ac.uk
3 **Department of Computer Science, University of Liverpool**
Liverpool, L69 3BX, UK
L.A.Goldberg@liverpool.ac.uk
4 **School of Mathematical Sciences, Queen Mary, University of London**
London E1 4NS, UK
m.jerrum@qmul.ac.uk

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 nontrivial non-lsm function is computationally as hard as any problem in #P. Furthermore, any non-trivial functional clone (in a sense that will be made precise below) contains the binary function "implies". 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.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases counting constraint satisfaction problems, approximation, complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.302

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. An instance of $\mathsf{CSP}(\Gamma)$ is a set of n variables taking values in D, together with a set of

* 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".

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 302–313

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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. Thus, constraint satisfaction problems (CSPs) may be viewed as generalised satisfiability problems, among which usual satisfiability is a very special case.

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 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 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 [18]. However, pps_ω-definability is more liberal than the corresponding notion in [18], 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 9 is a precise statement of this connection.) Many approximation-preserving reductions in the literature (for example, [12]) are based not on a fixed "gadget" but on sequences of increasingly-large gadgets that come arbitrarily close to some property without actually attaining it. 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 [18]: 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 8) 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 8), if F is not log-supermodular (and is not in the clone generated by disequality), 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 10). 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 $\psi = \sum_{v_{n+1}, \ldots, v_{n+m}} \prod_{j=1}^{s} \varphi_j$, 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:

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

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are assignments $\boldsymbol{x} : \{v_1, \ldots, v_n\} \to D$ and $\boldsymbol{y} : \{v_{n+1}, \ldots, v_{n+m}\} \to D$. 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. We use the following lemma.

▶ **Lemma 1.** *If $G \in \langle \mathcal{F} \rangle$ then $\langle \mathcal{F}, G \rangle = \langle \mathcal{F} \rangle$.*

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 $\mathcal{F} \cup \{\mathrm{EQ}\}$ 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 $\mathcal{F}$. Note 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 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 2. Suppose $F \in \langle \mathcal{F} \rangle_{\omega,p}$. Then there is a finite $S_F \subseteq \mathcal{F}$ 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 3.** *If $G \in \langle \mathcal{F} \rangle_\omega$ [or $G \in \langle \mathcal{F} \rangle_{\omega,p}$] then $\langle \mathcal{F}, G \rangle_\omega = \langle \mathcal{F} \rangle_\omega$ [$\langle \mathcal{F}, G \rangle_{\omega,p} = \langle \mathcal{F} \rangle_{\omega,p}$].*

Lemma 3 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 3 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 existentially 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* [9] 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" [9, Definition 1] if every member of $\langle \mathcal{R} \rangle_{\mathrm{R}}$ is definable by a 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 underlying 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 4.** *Suppose $\mathcal{F} \subseteq \mathcal{B}$. Then $\langle \{R_F \mid F \in \mathcal{F}\} \rangle_{\mathrm{R}} = \{R_F \mid F \in \langle \mathcal{F} \rangle\}$.*

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 8) 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.

▶ **Lemma 5.** *If $\mathcal{F} \subseteq$ LSM is any set of lsm functions then $\langle\mathcal{F}\rangle_\omega \subseteq$ 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 $G'(\boldsymbol{a}') = G(A)$ and $G'(\boldsymbol{b}') = G(B)$. 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 $G'(\boldsymbol{a}' \vee \boldsymbol{b}') = G(A \vee B)$ and $G'(\boldsymbol{a}' \wedge \boldsymbol{b}') = G(A \wedge B)$. Since G is lsm, we know that $G(\boldsymbol{a})G(\boldsymbol{b}) \le G(\boldsymbol{a} \vee \boldsymbol{b})G(\boldsymbol{a} \wedge \boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^{n+1}$. Thus, applying the Ahlswede-Daykin "Four-functions Theorem" [1, Theorem 1] with $\alpha = \beta = \gamma = \delta = G$,

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

As $\boldsymbol{a}', \boldsymbol{b}' \in \{0,1\}^n$ were arbitrary, G' is lsm. More details are in the full version [4]. ◀

An important example of an lsm function is the 0,1-function "implies", with $\text{IMP}(1,0) = 0$ and $\text{IMP}(x,y) = 1$ for all other x and y. We also think of this as a binary relation $\text{IMP} = \{(0,0), (0,1), (1,1)\}$. 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 10) that #BIS and #CSP(IMP) are interreducible via approximation-preserving reductions, and hence are of equivalent difficulty.

We know from Lemma 5 that $\langle\text{IMP}, \mathcal{B}_1\rangle_\omega \subseteq$ LSM. It is an open question whether the inclusion is strict. A related question is whether $\text{LSM} = \langle\mathcal{F}\rangle_\omega$ for any finite set $\mathcal{F}$ of lsm functions. A similar question has been investigated by Živný et al. [19] in this context of optimisation problems, where summation is replaced by maximisation or minimisation.

5 The main result

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 $\mathcal{B}^p_n$ the set of all functions $\{0,1\}^n \to \mathbb{R}^p$; also denote by $\mathcal{B}^p = \mathcal{B}^p_0 \cup \mathcal{B}^p_1 \cup \mathcal{B}^p_2 \cup \cdots$ the set of functions of all arities.

▶ Remark. 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 [13], but it results in a more uniform treatment of limits when we discuss efficient pps_ω-definability using these functions.

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 $i, j \in \{1, \ldots, n\}$, $i \neq j$, 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 [15, Theorem 44.1] for the supermodular case, but Schrijver's proof also applies to the modular case. The lemma is originally due to Topkis [16].

▶ **Lemma 6** (Topkis). *Let F be a function from $\{0, 1\}^n$ to $\mathbb{R}^{>0}$. 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 as well.) 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. From Lemma 7, we see that IMP does not really occupy a special position in $\langle \mathrm{IMP}, \mathcal{B}_1^p \rangle_{\omega,p}$, in the sense that there are other function F with $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \langle \mathrm{IMP}, \mathcal{B}_1^p \rangle_{\omega,p}$. Similarly, OR does not occupy a special position 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. From the proof of Lemma 7, we have the following inclusions between the four classes involved. $\langle \mathcal{B}_1^p \rangle_{\omega,p} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle_{\omega,p} \subseteq \langle \mathrm{OR}, \mathcal{B}_1^p \rangle_{\omega,p}$ and $\langle \mathcal{B}_1^p \rangle_{\omega,p} \subseteq \langle \mathrm{IMP}, \mathcal{B}_1^p \rangle_{\omega,p} \subseteq \langle \mathrm{OR}, \mathcal{B}_1^p \rangle_{\omega,p}$. 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.

5.3 Functional clones on 2-element set

▶ **Theorem 8.** *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 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 4, $C \subseteq \{R_{F'} \mid F' \in \langle F, \mathcal{B}_1^p\rangle\}$.

First, suppose 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 at the end of Section 5.2). Similarly, if OR $\in C$ or NAND $\in C$ then IMP $\in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$.

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

In fact, the set of all relational clones (also called "co-clones") is well understood. These are listed in [9, Table 2], which gives a plain basis for each relational clone. A graph illustrating the subset inclusions between the relational clones, called Post's lattice, is depicted in [2, Figure 2]. This graph is reproduced in the full version [4, Figure 1]. The relational clones are the vertices of the graph. 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 Post's lattice, 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 is the set of all Boolean linear equations) or C is some subset of IL_2, in which case it is below IL_2 in [4, 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 [4, 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 $V' = \{v_i \in V \mid \psi_i = \{(0),(1)\}\}$. 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. As is easily seen, of the 16 possible binary relations, the only ones that can occur are EQ, NEQ and $\{0,1\}^2$. (See the full version [4] for details.) 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 $V'' = \{v_1, \ldots, v_k\}$.

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

$$G(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \{0,1\}^{n-k}} F(\boldsymbol{x}, \boldsymbol{y}), \tag{2}$$

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 (2), $G \in \langle F, \mathcal{B}_1^p\rangle_{\omega,p}$. 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, [3, 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 $G \in \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$. (For example, if $g = a_1 x_1 + a_2 x_2 + a_3 \neg x_3$ then G can be written as $G(x_1, x_2, x_3) = \sum_{y_3} f_1(x_1) f_2(x_2) f_3(y_3) \mathrm{NEQ}(x_3, y_3)$, where $f_i(x) = \exp(a_i x)$ is a function in $\mathcal{B}_1^p$.) 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 3, $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$.

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, 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 1, 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 3, $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$.

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 $\mathrm{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$. Let $H(x_1, x_2) = \sum_{y_1, y_2} \mathrm{IMP}(y_1, x_1)\mathrm{IMP}(y_1, x_2)\mathrm{IMP}(x_1, y_2)\mathrm{IMP}(x_2, y_2)$. Note that $H(0,0) = H(1,1) = 2$ and $H(0,1) = H(1,0) = 1$. Now for any integer k, let

$$H_k(x_1, \ldots, x_n) = \sum_{y_1, \ldots, y_n} F(y_1, \ldots, y_n) \prod_{i=1}^{n} H(x_i, y_i)^k.$$

By construction, H_k is strictly positive. Also, as k gets large, $H_k(x_1, \ldots, x_n)$ gets closer and closer to $2^{kn} F(x_1, \ldots, x_n)$. Thus, for sufficiently large k, H_k is not lsm. By Lemma 1, $H \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$ so $H_k \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$. Applying Lemma 6 to H_k, 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 (v) of Lemma 7. In the former case, we are also finished since $\mathrm{OR} \in \langle \mathrm{IMP}, \mathrm{NEQ} \rangle$— details are given in the full version [4]. ◀

6 Complexity-theoretic consequences

In order to explore the computational consequences of Theorem 8, 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* (RAS) 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 $\varepsilon > 0$, $\Pr\left[e^{-\varepsilon}\Pi(w) \leq Y \leq e^{\varepsilon}\Pi(w)\right] \geq 3/4$. The RAS 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 [14, Definition 10.2].

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 is the 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_1 *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 type 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 ≠ 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, here is a precise statement of the

[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 with 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 [17, Thm 10.5.9].

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 $\#\mathsf{CSP}(\mathcal{F})$
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.

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 9.** *Suppose $\mathcal{F}\subseteq\mathcal{B}^p$ is finite. 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 an appropriate ε' the proof can be completed by a fairly straightforward computation (see the full version [4]). ◀

▶ **Theorem 10.** *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 $S\subseteq\mathcal{B}_1^p$, 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 11.** Let $F\in\mathcal{B}_2^p$ be the function defined by $F(0,0)=F(1,1)=\lambda$ and $F(0,1)=F(1,0)=1$, where $\lambda>1$. Then, by Theorem 10, $\#\mathsf{CSP}(F,S)$ is $\#\mathsf{BIS}$-hard, for some set S of unary weights. (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 [12].

▶ **Example 12.** If F is as before, but $\lambda\in(0,1)$, then $F\notin\mathsf{LSM}$ and Theorem 10 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.

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 Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discr. Appl. Math.*, 123(1-3):155–225, 2002.

4 A. A. Bulatov, M. Dyer, L. A. Goldberg, and M. Jerrum. Log-supermodular functions, functional clones and counting CSPs. *ArXiv e-prints*, August 2011.

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 *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*. SIAM, 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 Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic Ising with local fields. *Combin. Probab. Comput.*, 16(1):43–61, 2007.

13 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 *LNCS*, pages 396–407. Springer, 2010.

14 Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, Cambridge, 2005. Randomized algorithms and probabilistic analysis.

15 Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

16 Donald M. Topkis. Minimizing a submodular function on a lattice. *Operat. Res.*, 26:305–321, 1978.

17 Ingo Wegener. *Complexity Theory*. Springer-Verlag, Berlin, 2005. Exploring the limits of efficient algorithms, Translated from the German by Randall Pruim.

18 Tomoyuki Yamakami. Approximate counting for complex-weighted Boolean constraint satisfaction problems. *CoRR*, abs/1007.0391, 2010.

19 Stanislav Živný, David A. Cohen, and Peter G. Jeavons. The expressive power of binary submodular functions. *Discrete Appl. Math.*, 157(15):3347–3358, 2009.