Approximating Partition Functions of Bounded-Degree Boolean Counting Constraint Satisfaction Problems[*][†]

Andreas Galanis[1], Leslie Ann Goldberg[2], and Kuan Yang[3]

1 **University of Oxford, Oxford, UK**
andreas.galanis@cs.ox.ac.uk
2 **University of Oxford, Oxford, UK**
leslie.goldberg@cs.ox.ac.uk
3 **University of Oxford, Oxford, UK**
kuan.yang@cs.ox.ac.uk

Abstract

We study the complexity of approximate counting Constraint Satisfaction Problems (#CSPs) in a bounded degree setting. Specifically, given a Boolean constraint language Γ and a degree bound Δ, we study the complexity of $\#\mathsf{CSP}_\Delta(\Gamma)$, which is the problem of counting satisfying assignments to CSP instances with constraints from Γ and whose variables can appear at most Δ times. Our main result shows that: (i) if every function in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma)$ is in FP for all Δ, (ii) otherwise, if every function in Γ is in a class called IM_2, then for all sufficiently large Δ, $\#\mathsf{CSP}_\Delta(\Gamma)$ is equivalent under approximation-preserving (AP) reductions to the counting problem #BIS (the problem of counting independent sets in bipartite graphs) (iii) otherwise, for all sufficiently large Δ, it is NP-hard to approximate the number of satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$, even within an exponential factor. Our result extends previous results, which apply only in the so-called “conservative” case.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures

Keywords and phrases Constraint Satisfaction, Approximate Counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.27

1 Introduction

Constraint Satisfaction Problems (CSPs), which originated in Artificial Intelligence [18] provide a general framework for modelling decision, counting and approximate counting problems. The paradigm is sufficiently general that applications from diverse areas such as database theory, scheduling and graph theory can all be captured (see, for example, [14, 15, 17]). Moreover, all graph homomorphism decision and counting problems [12] can be re-cast in the CSP framework and partition function problems from statistical physics [21] can be represented as counting CSPs. Given the usefulness of CSPs, the study of the complexity

[*] A full version of the paper (with the same theorem-numbering) containing detailed proofs is available at http://arxiv.org/abs/1610.04055.

[†] The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors’ views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein. Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 27; pp. 27:1–27:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of CSPs is a an extremely active area in computational complexity (for example, see [2] and the references therein).

In this paper, we are concerned with Boolean counting CSPs. An instance $I = (V, \mathcal{C})$ of a Boolean counting CSP consists of a set V of *variables* and a set $\mathcal{C}$ of constraints. An assignment $\sigma : V \to \{0, 1\}$ assigns a Boolean value called a "spin" to each variable. Each constraint associates a tuple $(v_1, \ldots, v_k)$ of variables with a Boolean relation which constrains the spins that can be assigned to $v_1, \ldots, v_k$. The assignment σ is said to "satisfy" the constraint if the tuple $(\sigma(v_1), \ldots, \sigma(v_k))$ is in the corresponding relation. An assignment is said to be "satisfying" if it satisfies all constraints. A Constraint Satisfaction Problem comes with two important parameters – the constraint language Γ is the set of all relations that may be used in constraints and the degree Δ is the maximum number of times that any variable $v \in V$ may be used in constraints in any instance. The number of satisfying assignments is denoted Z_I. The computational problem $\#\mathsf{CSP}_\Delta(\Gamma)$ is the problem of computing Z_I, given a CSP instance I with constraints in Γ and degree at most Δ. We use $\#\mathsf{CSP}(\Gamma)$ to denote the version of the problem in which the degree of instances is unconstrained.

Although constraints are supported by Boolean relations, they can be used to code up weighted interactions such as those that arise in statistical physics. For example, let R be the "not-all-equal" relation of arity 3. Then consider the conjunction of $R(x, a, b)$ and $R(y, a, b)$. There are two satisfying assignments with $\sigma(x) = 0$ and $\sigma(y) = 1$ since $\sigma(a)$ and $\sigma(b)$ must differ. Similarly, there are two satisfying assignments with $\sigma(x) = 1$ and $\sigma(y) = 0$. On the other hand, there are three satisfying assignments with $\sigma(x) = \sigma(y) = 1$ and there are three satisfying assignments with $\sigma(x) = \sigma(y) = 0$. Thus, the induced interaction on the variables x and y is the same as the interaction of the ferromagnetic Ising model (at an appropriate temperature) – an assignment in which x and y have the same spin has weight 3, whereas an assignment where they have different spins has weight 2.

For every $\Delta \geq 3$, the work of Cai, Lu and Xia [5] completely classifies the complexity of exactly solving $\#\mathsf{CSP}_\Delta(\Gamma)$, depending on the parameter Γ. If every relation in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma)$ is solvable in polynomial time (so the problem in the complexity class FP). Otherwise, it is $\#\mathsf{P}$-complete. The term "affine" will be defined in § 2. Roughly, it means that the tuples in the relation are solutions to a linear system, so Gaussian elimination gives an appropriate polynomial-time algorithm. The characterisation of Cai, Lu and Xia is exactly the same classification that was obtained for the unbounded problem $\#\mathsf{CSP}(\Gamma)$ by Creignou and Hermann [6]. Thus, as far as exact counting is concerned, the degree-bound Δ does not affect the complexity as long as $\Delta \geq 3$. As Cai, Lu and Xia point out, the dichotomy is false for $\Delta = 2$, where $\#\mathsf{CSP}_2(\Gamma)$ is equivalent to the Holant problem $\mathsf{Holant}(\Gamma)$ – see the references in [5] for more information about Holant problems.

Much less is known about the complexity of *approximately* solving $\#\mathsf{CSP}_\Delta(\Gamma)$. In fact, even the *decision problem* is still open. While Schaefer [19] completely classified the complexity of the decision problem $\mathsf{CSP}(\Gamma)$ – where the goal is to determine whether or not Z_I is 0 for an instance of $\#\mathsf{CSP}(\Gamma)$ – the complexity of the corresponding decision problem $\mathsf{CSP}_\Delta(\Gamma)$, where the instance has degree at most Δ, is still not completely resolved. For $\Delta \geq 3$, Dalmau and Ford [7] have solved the special case where Γ includes both of the relations $R_{\delta_0} = \{0\}$ and $R_{\delta_1} = \{1\}$. This special case is known as the "conservative case" in the CSP literature. For $\Delta \geq 6$, Dyer et al. [9] have classified the difficulty of the approximation problem:

- If every relation in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is in FP.
- Otherwise, if every relation in Γ is in a class called IM_2 (a class defined in § 2) then $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is equivalent under approximation-preserving (AP) reductions to the counting problem $\#\mathsf{BIS}$ (the problem of counting independent sets in bipartite graphs).
- Otherwise, there is no FPRAS for $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ unless $\mathsf{NP} = \mathsf{RP}$.

Dyer et al. made only partial progress on the cases where $\Delta \in \{3, 4, 5\}$. We refer the reader to [9, 16] for a discussion of the partial classification. However, it is worth noting here that the complexity of $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is closely related to the complexity of counting satisfying assignments of so-called read-d Monotone CNF Formulas. Crucial progress was made by Liu and Lu [16], who completely resolved the complexity of the latter problem. Given the work of Liu and Lu, a complete classification of $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ for $\Delta \in \{3, 4, 5\}$ may be in reach.

The restriction that R_{δ_0} and R_{δ_1} are contained in Γ is a severe one because it does not apply to many natural applications. On the other hand, we are a long way from a precise understanding of the complexity of $\#\mathsf{CSP}_\Delta(\Gamma)$ without this restriction because there are specific, relevant parameters that we do not understand. For example, for a positive integer k, let Γ be the singleton set containing only the arity-k "not-all-spin-1" relation. Then satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$ correspond to independent sets of a k-uniform hypergraph with maximum degree Δ. It is known that there is an FPRAS for $\Delta = O(2^{k/2})$ [13] and that the problem is NP-hard to approximate for $\Delta = \Omega(2^{k/2})$ [1]; the implicit constants in these bounds do not currently match and thus, for big k, there is a large range of Δ's where we do not yet know the complexity of approximating $\#\mathsf{CSP}_\Delta(\Gamma)$. If Γ instead contains (only) the arity-k "at-least-one-spin-0" relation then satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$ correspond to the so-called "strong" independent sets of a k-uniform hypergraph. Song, Yin and Zhao [20] have presented a barrier for hardness results, showing why current technology is unsuitable for resolving the cases where $\Delta \in \{4, 5\}$ (roughly, these cases are in "non-uniqueness", but this is not realisable by finite gadgets).

The purpose of the present paper is to remove the severe restriction that R_{δ_0} and R_{δ_1} are contained in Γ in the approximate counting classification of $\#\mathsf{CSP}_\Delta(\Gamma)$ from [9]. Since pinning down precise thresholds seems a long way out of reach, we instead focus on whether there is a "barrier" value Δ_0 such that, for all $\Delta \geq \Delta_0$, approximation is intractable. Since we wish to get the strongest possible inapproximability results (showing the hardness of approximating Z_I even within an exponential factor), we define the following computational problem, which has an extra parameter $c > 1$ that captures the desired accuracy.

Name $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$.
Instance An n-variable instance I of a CSP with constraint language Γ and degree at most Δ.
Output A number $\widehat{Z}$ such that $c^{-n} Z_I \leq \widehat{Z} \leq c^n Z_I$.

Although we have not yet defined all of the terms, we can now at least state (a weak version of) our result.

▶ **Theorem 1.** *Let Γ be a Boolean constraint language. Then,*
1. *If every relation in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ is in FP.*
2. *Otherwise, if every relation in Γ is in the class IM_2, then there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, $\#\mathsf{CSP}_\Delta(\Gamma)$ is $\#\mathsf{BIS}$-equivalent under AP-reductions.*
3. *Otherwise, there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ is NP-hard.*

After defining all of the terms, we will state a stronger theorem, Theorem 6, which immediately implies Theorem 1. The stronger version applies to the #CSP problems that we have already introduced, but it also applies to other restrictions of these problems, which have even more applications.

We now explain the restriction. Note that in the CSP framework, as we have defined it, the variables that are constrained by a given constraint need not be distinct. Thus, if the arity-4 relation R is present in a constraint language Γ, then an instance of $\#\mathsf{CSP}(\Gamma)$

with variables x and y may contain a constraint such as $R(x, x, y, x)$. This ability to repeat variables is equivalent to assuming that equality relations of all arities are present in Γ. This feature of the CSP definition is inconvenient for two reasons: (1) It does not fit well with some spin-system applications, and (2) In many settings, it obscures the nuanced complexity classification that arise.

As an example of (1), recall the application where Γ is the singleton set containing only the arity-k "not-all-spin-1" relation. As we noted earlier, satisfying assignments of a $\#\mathsf{CSP}(\Gamma)$ instance correspond to independent sets of a k-uniform hypergraph. Here, hyperedges are size-k subsets of vertices and it does not make sense to allow repeated vertices!

The point (2) is well-known. In fact, the "equality is always present" assumption is the main feature that separates #CSPs from the more general Holant framework [3].

In our current setting, it turns out that adding equality functions to Γ does not change the complexity classification, but this is a result of our theorems rather than an a priori assumption – indeed, determining which constraint languages Γ can appropriately simulate equality functions is one of the difficulties – thus, throwing equalities in "for free" would substantially weaken our results! Our main result, Theorem 6, which will be presented in § 2, applies both to the #CSPs that we have already defined, and to more refined versions, in which constraints may not repeat variables.

We wish now to discuss an important special case in which both the #CSPs and the refined versions have already been studied. This is the special case in which Γ consists of a single relation which is symmetric in its arguments. A symmetric relation that is not affine is not in IM_2. Therefore, Item 2 in the statement of Theorem 1 never arises in this special case. Our earlier paper [11] shows that, in this case (where Γ consists of a single, symmetric, non-affine relation) there is an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ is NP-hard.

While the work of [11] is important for this paper, note that the special case is far from general – in particular, it is easy to induce asymmetric constraints using symmetric ones. For example, suppose that R_1 is the (symmetric) arity-2 "not-all-spin-1" constraint, R_2 is the (symmetric) arity-2 "not the same spin" constraint and $R_3 = \{(0,0), (0,1), (1,1)\}$ is the (asymmetric) arity-2 "Implies" constraint. Then the conjunction of $R_1(x, a)$ and $R_2(a, y)$ induces $R_3(x, y)$.

It is interesting that Theorem 1 is exactly the same classification that was obtained for the *unbounded* problem $\#\mathsf{CSP}(\Gamma)$ by Dyer et al. [10]. In particular, they showed

1. If every relation in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ is in FP.
2. Otherwise, if every relation in Γ is in the class IM_2, then $\#\mathsf{CSP}(\Gamma)$ is $\#\mathsf{BIS}$-equivalent under AP-reductions.
3. Otherwise, $\#\mathsf{CSP}(\Gamma)$ is $\#\mathsf{SAT}$-equivalent under AP-reductions, where $\#\mathsf{SAT}$ is the problem of counting the satisfying assignments of a Boolean formula.

The inapproximability that we demonstrate in Item 3 of Theorem 1 is stronger than what was known in the unbounded case, both (obviously) because of the degree bound, but also because we show that it is hard to get within an exponential factor. (This strong kind of inapproximability was also missing from the results of [9]).

2 Definitions and Statement of Main Result

Before giving formal definitions of the problems that we study, we introduce some notation. We use boldface letters to denote Boolean vectors. A *pseudo-Boolean* function is a function of the form $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ for some positive integer k, which is called the *arity* of f.

▶ **Definition 2.** Given a pseudo-Boolean function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$, we use the notation R_f to denote the relation $R_f = \{\mathbf{x} \in \{0,1\}^k \mid f(\mathbf{x}) > 0\}$, which is the relation underlying f.

If the range of f is $\{0,1\}$ then f is said to be a *Boolean function* and of course in that case $R_f = \{\mathbf{x} \in \{0,1\}^k \mid f(\mathbf{x}) = 1\}$.

In order to allow consistency with obvious generalisations, our formal definition of the Boolean Constraint Satisfaction Problem is in terms of Boolean functions (rather than, equivalently, using the underlying relations).

A *Constraint language* Γ is a set of pseudo-Boolean functions. It is a *Boolean constraint language* if all of the functions in it are Boolean functions. An instance $I = (V, \mathcal{C})$ of a CSP with constraint language Γ consists of a set V of variables and a set $\mathcal{C}$ of constraints. Each constraint $C_i \in \mathcal{C}$ is of the form $f_i(v_{i,1}, \ldots, v_{i,k_i})$ where f_i is an arity-k_i function in Γ and $(v_{i,1}, \ldots, v_{i,k_i})$ is a tuple of (not necessarily distinct) variables in V. The constraint C_i is said to be "Repeat-Free" if all of the variables are distinct. Each *assignment* $\sigma : V \to \{0,1\}$ of Boolean values to the variables in V has a weight $w_I(\sigma) := \prod_{f_i(v_{i,1},\ldots,v_{i,k_i}) \in \mathcal{C}} f_i(\sigma(v_{i,1}), \ldots, \sigma(v_{i,k_i}))$. The *partition function* maps the instance I to the quantity $Z_I := \sum_{\sigma : V \to \{0,1\}} w_I(\sigma) = \sum_{\sigma : V \to \{0,1\}} \prod_{f_i(v_{i,1},\ldots,v_{i,k_i}) \in \mathcal{C}} f_i(\sigma(v_{i,1}), \ldots, \sigma(v_{i,k_i}))$.

If Γ is a Boolean constraint language then it is easy to see that $w_I(\sigma) = 1$ if the assignment is satisfying and $w_I(\sigma) = 0$, otherwise. Thus, Z_I is the number of satisfying assignments of I.

When $Z_I > 0$, we will use $\mu_I(\cdot)$ to denote the Gibbs distribution corresponding to Z_I. This is the probability distribution on the set of assignments $\sigma : V \to \{0,1\}$ such that $\mu_I(\sigma) = w_I(\sigma)/Z_I$ for all $\sigma : V \to \{0,1\}$.

The *degree* $d_v(C)$ of a variable v in a constraint C is the number of times that the variable v appears in the tuple corresponding to C and the degree d_v of the variable is $d_v = \sum_{C \in \mathcal{C}} d_v(C)$. Finally, the degree of the instance I is $\max_{v \in V} d_v$.

▶ **Definition 3.** $\#\mathsf{CSP}_\Delta(\Gamma)$ is the problem of computing Z_I, given a CSP instance I with constraints in Γ and degree at most Δ. $\#\mathsf{CSP}(\Gamma)$ is the version of the problem in which the degree of instances is unconstrained. $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ has an extra parameter $c > 1$ that captures the desired accuracy. The problem is to compute a number $\widehat{Z}$ such that $c^{-n} Z_I \leq \widehat{Z} \leq c^n Z_I$, where n is the number of variables in the instance I. The problems $\#\mathsf{NoRepeatCSP}_\Delta(\Gamma)$, $\#\mathsf{NoRepeatCSP}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\Gamma)$ are defined similarly, except that inputs are restricted so that all constraints are Repeat-Free.

▶ **Definition 4.** A Boolean function $f : \{0,1\}^k \to \{0,1\}$ is *affine* if there is a $k \times k$ Boolean matrix $\mathbf{A}$ and a length-k Boolean vector $\mathbf{b}$ such that R_f is equal to the set of solutions $\mathbf{x}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$ over GF(2).

▶ **Definition 5** (The set of functions IM_2). A Boolean function $f : \{0,1\}^k \to \{0,1\}$ is in IM_2 if $f(x_1, \ldots, x_k)$ is logically equivalent to a conjuction of (any number of) predicates of the form x_i, $\neg x_i$ or $x_i \Rightarrow x_j$.

We have now defined all of the terms in our main theorem apart from some well-known concepts from complexity theory, which we discuss next. FP is the class of computational problems (with numerical output) that can be solved in polynomial time. An FPRAS is a randomised algorithm that produces approximate solutions within specified relative error with high probability in polynomial time. For two counting problems $\#\mathsf{A}$ and $\#\mathsf{B}$, we say that $\#\mathsf{A}$ is $\#\mathsf{B}$-easy if there is an approximation-preserving (AP)-reduction from $\#\mathsf{A}$ to $\#\mathsf{B}$. The formal definition of an AP-reduction can be found in [8]. It is a randomised Turing reduction that yields close approximations to $\#\mathsf{A}$ when provided with close approximations to $\#\mathsf{B}$. The definition of AP-reduction meshes with the definition of FPRAS in the sense that the existence of an FPRAS for #B implies the existence of an FPRAS for #A. We say that #A is #B-hard if there is an AP-reduction from #B to #A. Finally, we say that #A is #B-equivalent if #A is both #B-easy and #B-hard.

The problem of counting satisfying assignments of a Boolean formula is denoted by #SAT. Every counting problem in #P is AP-reducible to #SAT, so #SAT is said to be complete for #P with respect to AP-reductions. It is known that there is no FPRAS for #SAT unless RP = NP. The problem of counting independent sets in a bipartite graph is denoted by #BIS. The problem #BIS appears to be of intermediate complexity: there is no known FPRAS for #BIS (and it is generally believed that none exists) but there is no known AP-reduction from #SAT to #BIS. Indeed, #BIS is complete with respect to AP-reductions for a complexity class $\#\mathsf{RH}\Pi_1$.

Given all of these definitions, we now formally state the stronger version of Theorem 1 promised in the introduction. The proof can be found in full version.

▶ **Theorem 6.** *Let Γ be a Boolean constraint language. Then,*

1. *If every function in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}(\Gamma)$ are both in* FP.
2. *Otherwise, if $\Gamma \subseteq IM_2$, then there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, $\#\mathsf{CSP}_\Delta(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_\Delta(\Gamma)$ are both #BIS-equivalent under AP-reductions, and*
3. *Otherwise, there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\Gamma)$ are both NP-hard.*

3 Overview of the Proof of Theorem 6

In this section, we give a non-technical overview of the proof of Theorem 6. Our objective is to illustrate the main ideas and obstacles without delving into the more detailed definitions. A more technical overview can be found in § 5. Our focus in this section will be on the case where Γ consists of a single Boolean function $f : \{0,1\}^k \to \{0,1\}$. This case is the main ingredient in the proof of the theorem.

A typical approach for showing that a counting CSP is intractable is to use an instance of the CSP to build a "gadget" which simulates an intractable binary 2-spin constraint. This was the approach used in [11], which proved the intractability of $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ for any *symmetric* non-affine Boolean function f. There, an instance I of $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ was constructed, along with variables x and y, such that for all spins $s_x \in \{0,1\}$ and $s_y \in \{0,1\}$ the marginal distribution $\mu_I(x,y)$ satisfies

$$\mu_I(\sigma(x) = s_x, \sigma(y) = s_y) = g(s_x, s_y)/(g(0,0) + g(0,1) + g(1,0) + g(1,1)), \qquad (1)$$

where g is a binary function that codes up the interaction of an intractable anti-ferromagnetic 2-spin system. We will not need to give detailed definitions of 2-spin systems in this paper. Instead, we give a sufficient condition for intractability.

▶ **Definition 7.** A binary function $g : \{0,1\}^2 \to \mathbb{R}_{\geq 0}$ is said to be "*hard*" if all of the following hold: $g(0,0)+g(1,1) > 0$, $\min\{g(0,0), g(1,1)\} < \sqrt{g(0,1)g(1,0)}$, and $\max\{g(0,0), g(1,1)\} \leq \sqrt{g(0,1)g(1,0)}$.

It was established in [11] that the ability to "simulate" a hard function g in the sense of (1) ensures that $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ is NP-hard to approximate, even within an exponential factor.

A key feature of *symmetric* Boolean functions f which facilitated such simulation in [11] was the fact that the class of relevant hard functions g is well-behaved, and it turned out that it suffices to encode such a hard binary function with only ϵ-accuracy, for some sufficiently small $\epsilon > 0$, and this was enough to ensure the NP-hardness of $\#\mathsf{CSP}_{\Delta,c}(\{f\})$.

The main obstacle in adapting the approach of [11] to the case where f need not be symmetric in its arguments arises when f is in IM_2. It is unlikely that such a function f can simulate a hard function g in the sense of (1) – indeed such a simulation would prove the (very surprising) result that #BIS does not have an FPRAS (unless NP = RP). Thus, for $f \in IM_2$, we need instead to encode a binary function which will allow us to connect the problem $\#\mathsf{NoRepeatCSP}_{\Delta}(\{f\})$ to #BIS.

Now consider the binary Boolean function Implies whose underlying relation $R_{\mathsf{Implies}} = \{(0,0),(0,1),(1,1)\}$ contains all (x,y) satisfying $x \Rightarrow y$. Obviously, Implies is not symmetric, and it is not hard according to Definition 7. On bipartite instances, however, the symmetry can be restored by interpreting differently the spins 0 and 1 on the two parts of the graph, and this leads to a connection with #BIS. In particular, it is well-known [10] that $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ is equivalent to #BIS under AP-reductions. This connection was extended to the bounded-degree setting by [4].

Unfortunately, the symmetrisation which connects $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ to #BIS is not very robust. For example, suppose that a (non-symmetric) Boolean function f can be used to simulate, in the sense of (1), a binary function g which is very close to Implies. In particular, suppose that for some $\epsilon > 0$ and $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ satisfying $|\epsilon_i| \leq \epsilon$ for $i = 1,2,3,4$, we have $g(0,0) = 1+\epsilon_1$, $g(0,1) = 1+\epsilon_2$, $g(1,0) = \epsilon_3$, and $g(1,1) = 1+\epsilon_4$. Such a close approximation is about the best that can be expected using the kind of approximate encodings that are available. However, the complexity of asymmetric 2-spin systems is not sufficiently well understood to exploit such a simulation. Surprisingly, for *any* arbitrarily small constant $\epsilon > 0$, it is not known even whether the unbounded degree version $\#\mathsf{CSP}(\{g\})$ is #BIS-hard, and certainly nothing is known in our bounded-degree setting! The trouble is that the symmetrisation that works for Implies (i.e., when $\epsilon_i = 0$ for $i = 1,2,3,4$) is no longer guaranteed to symmetrise the imperfect version with the ϵ_i's, so the swapping of spin-0 and spin-1 values on one side of the bipartite graph leads to an *asymmetric* 2-spin system on bipartite graphs and this does not fall into the scope of known results [4] concerning bounded-degree bipartite 2-spin systems.

Our approach to handle this problem for $f \in IM_2$ is to carefully ensure that there is no accuracy error ϵ in encoding the function Implies. In other words, we show that, using $f \in IM_2$, we can encode Implies *perfectly*, a task which is surprisingly intricate in the repeat-free setting. Our main technical theorem, Theorem 17, achieves this goal. Namely, it shows that, for every non-affine Boolean function f, either f simulates a hard function (with arbitrarily small accuracy-error ϵ, which leads to the desired intractability of $\#\mathsf{NoRepeatCSP}_{\Delta}(\{f\})$) or else f "supports perfect equality" – a concept which will be defined later, but essentially means that f can be used to perfectly simulate the binary function EQ with underlying relation $R_{\mathsf{EQ}} = \{(0,0),(1,1)\}$. Using EQ, it is possible to simulate repeated variables in constraints, so the #BIS-hardness of $\#\mathsf{CSP}_{\Delta}(\{f\})$ follows from [10]. When $f \notin IM_2$ but f supports perfect equality, instead of reducing to the work in [10], we work somewhat harder to make sure that we also get the strong (exponential factor) inapproximability given in Theorem 6.

4 Pinning, equality and simulating functions

An important case in our proof is the case where Γ contains a single function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. In this case, we can we simplify the notation because the constraints in an instance I are in one-to-one correspondence with k-tuples of variables (there is no need to repeat the name of the function f in each constraint). So, for convenience, we make the following definitions.

A *k-tuple hypergraph* $H = (V, \mathcal{F})$ consists of a set V of vertices, together with a set $\mathcal{F}$ of *hyperarcs*, where every hyperarc in $\mathcal{F}$ is a k-tuple of distinct vertices in V. The degree of H is the maximum, over all vertices $v \in V$, of the number of hyperarcs that contain v. Given a function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$, we let $I_f(H)$ denote the instance of $\#\mathsf{NoRepeatCSP}(\{f\})$ whose constraints correspond to the hyperarcs of H. Given an assignment $\sigma\colon V \to \{0,1\}$ we define $w_{f;H}(\sigma) := \prod_{(v_1,\ldots,v_k)\in\mathcal{F}} f(\sigma(v_1),\ldots,\sigma(v_k))$ and $Z_{f;H} := \sum_{\sigma:V\to\{0,1\}} w_{f;H}(\sigma)$, so $Z_{I_f(H)} = Z_{f;I_f(H)}$. By analogy to the Gibbs distribution on satisfying assignments, when $Z_{f;H} > 0$, we use $\mu_{f;H}(\cdot)$ to denote the probability distribution in which, for all assignments $\sigma : V \to \{0,1\}$, $\mu_{f;H}(\sigma) = w_{f;H}(\sigma)/Z_{f;H}$.

Given a function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and a positive integer Δ, we define $\#\mathsf{Multi2Spin}_\Delta(f)$ to be the problem of computing $Z_{f;H}$, given as input a k-tuple hypergraph H with degree at most Δ. The name $\#\mathsf{Multi2Spin}_\Delta(f)$ indicates that the problem is to compute the partition function of a 2-spin system with multi-body interactions specified by f and degree-bound Δ. Given a real number $c > 1$, the problem $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ has the same input, and the goal, when the input has n vertices, is to compute a number $\widehat{Z}$ such that $c^{-n}Z_{f;H} \leq \widehat{Z} \leq c^n Z_{f;H}$. Clearly, $\#\mathsf{Multi2Spin}_\Delta(f)$ is equivalent to $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ and $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is equivalent to $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\{f\})$.

4.1 Supporting pinning and equality

Let k be a positive integer and let $H = (V, \mathcal{F})$ be a k-tuple hypergraph. Given a configuration $\sigma : V \to \{0,1\}$ and a subset $T \subseteq V$, we will use σ_T to denote the restriction of σ to vertices in T. For a vertex $v \in V$, we will also use σ_v to denote the spin $\sigma(v)$ of vertex v in σ. The following definitions are generalisations of definitions from [11].

▶ **Definition 8.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. Suppose that $\epsilon \geq 0$ and $s \in \{0,1\}$. The k-tuple hypergraph H is an *ϵ-realisation of pinning-to-s* if there exists a vertex v of H such that $\mu_{f;H}(\sigma_v = s) \geq 1 - \epsilon$.

▶ **Definition 9.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $s \in \{0,1\}$. We say that f supports *pinning-to-s* if, for every $\epsilon > 0$, there is a k-tuple hypergraph which is an ϵ-realisation of pinning-to-s. We say that f supports *perfect* pinning-to-s if there is a k-tuple hypergraph which is a 0-realisation of pinning-to-s.

We now define what it means for a function f to support (perfect) equality (cf. § 3).

▶ **Definition 10.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $\epsilon \geq 0$. The k-tuple hypergraph H is an *ϵ-realisation of equality* if there exist distinct vertices v_1 and v_2 of H such that, for each $s \in \{0,1\}$, $\mu_{f;H}(\sigma_{v_1} = \sigma_{v_2} = s) \geq (1-\epsilon)/2$.

▶ **Definition 11.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. The function f *supports equality* if, for every $\epsilon > 0$, there is a k-tuple hypergraph which is an ϵ-realisation of equality. The function f *supports perfect equality* if there is a k-tuple hypergraph which is a 0-realisation of equality.

4.2 Realising conditional distributions induced by pinning and equality

Given a set S of vertices, we write $\sigma_S = \mathbf{0}$ to denote the event that all vertices in S are assigned the spin 0 under the assignment σ. We similarly write $\sigma_S = \mathbf{1}$ to denote the event that all vertices in S are assigned the spin 1 under the assignment σ. Finally, we use use σ_S^{eq} to denote the event that all vertices in S have the same spin under σ (the spin could be 0 or 1). The following definition is a generalisation of Definition 16 of [11] except that we have changed the notation slightly for convenience.

▶ **Definition 12** ([11, Definition 16]). Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. Let $H = (V, \mathcal{F})$ be a k-tuple hypergraph. Let $\mathcal{V} = (V_{\mathsf{pin0}}, V_{\mathsf{pin1}}, \mathcal{V}_{\mathsf{eq}})$ where V_{pin0} and V_{pin1} are disjoint subsets of V and $\mathcal{V}_{\mathsf{eq}}$ is a (possibly empty) set of disjoint subsets of $V \backslash (V_{\mathsf{pin0}} \cup V_{\mathsf{pin1}})$. Suppose that: (i) $V_{\mathsf{pin0}} = \emptyset$ if f does not support pinning-to-0, (ii) $V_{\mathsf{pin1}} = \emptyset$ if f does not support pinning-to-1, (iii) $\mathcal{V}_{\mathsf{eq}} = \emptyset$ if f does not support equality, (iv) it holds that $\mu_{f;H}(\sigma_{V_{\mathsf{pin0}}} = \mathbf{0}, \sigma_{V_{\mathsf{pin1}}} = \mathbf{1}, \bigcap_{W \in \mathcal{V}_{\mathsf{eq}}} \sigma_W^{\mathsf{eq}}) > 0$. We will then say that "$\mathcal{V}$ is *admissible* for H with respect to f" and we will denote by $\mu_{f;H}^{\mathrm{cond}(\mathcal{V})}$ the probability distribution $\mu_{f;H}(\cdot \mid \sigma_{V_{\mathsf{pin0}}} = \mathbf{0}, \sigma_{V_{\mathsf{pin1}}} = \mathbf{1}, \bigcap_{W \in \mathcal{V}_{\mathsf{eq}}} \sigma_W^{\mathsf{eq}})$.

4.3 Simulating hard functions and inapproximability results

We can now give a formal definition of "simulation", along the lines that was informally discussed in § 3 (Equation (1)).

▶ **Definition 14.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $g : \{0,1\}^t \to \mathbb{R}_{\geq 0}$. The function f simulates the function g if there is a k-tuple hypergraph H, an admissible set $\mathcal{V}$ for H with respect to f, and t vertices $v_1, v_2, \ldots, v_t$ of H such that, for all $(s_1, s_2, \ldots, s_t) \in \{0,1\}^t$,

$$\mu_{f;H}^{\mathrm{cond}(\mathcal{V})}(\sigma(v_1) = s_1, \sigma(v_2) = s_2, \ldots, \sigma(v_t) = s_t) = \frac{g(s_1, s_2, \ldots, s_t)}{\sum_{(s'_1, s'_2, \ldots, s'_t) \in \{0,1\}^t} g(s'_1, s'_2, \ldots, s'_t)}.$$

If $\mathcal{V} = (\emptyset, \emptyset, \emptyset)$, then we say that f *perfectly simulates* g. More generally, we say that f simulates a set of functions $\mathcal{G}$ if f simulates every $g \in \mathcal{G}$.

The connection betweeen "hard" as defined in Definition 7 and intractability is given in the following lemma. The lemma is stated for symmetric functions in [11], but the proof also works for asymmetric functions.

▶ **Lemma 15** ([11, Lemma 18]). *Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. If f simulates a hard function, then for all sufficiently large Δ, there exists $c > 1$ such that #$\mathsf{Multi2Spin}_{\Delta,c}(f)$ is* NP*-hard.* ◀

5 Proof Sketch

In this section, for a Boolean function $f : \{0,1\}^k \to \{0,1\}$, we consider the complexity of the problems #$\mathsf{Multi2Spin}_{\Delta}(f)$ and #$\mathsf{Multi2Spin}_{\Delta,c}(f)$. Classifying the complexity of these problems is the most important step in the proof of Theorem 6. Namely, to obtain Theorem 6, it suffices to show that for every non-affine function f, we have that:

- If f is in IM_2, then for all sufficiently large Δ, #$\mathsf{Multi2Spin}_{\Delta}(f)$ is #BIS-equivalent.
- If f is not in IM_2, then for all sufficiently large Δ, there exists a real number $c > 1$ such that #$\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard.

Our main technical theorem to prove this is the following classification of Boolean functions, which asserts that every non-affine function either supports perfect equality or simulates a hard function. A proof sketch is provided later.

▶ **Theorem 17.** *Let $k \geq 2$ and let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. Then at least one of three following propositions is true:*

1. *f is affine;*
2. *f supports perfect equality;*
3. *f simulates a hard function.*

When f simulates a hard function, using Lemma 15, we can immediately conclude that for all sufficiently large Δ, there exists $c > 1$ such that #$\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard. As we already discussed in § 3, it is important that, in the case where f does not simulate a hard function, Theorem 17 guarantees that f supports *perfect* equality (rather than simple imperfect equality); this allows us to recover the connection to #BIS for those $f \in IM_2$. In fact, when f supports perfect equality, we can effectively carry out (a strengthening of) the program in [10] to obtain the following classification which perfectly aligns with Theorem 6.

▶ **Theorem 18.** *Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function that is not affine. Suppose that f supports perfect equality.*

1. *If f is in IM_2, then for all sufficiently large Δ, $\#\mathsf{Multi2Spin}_\Delta(f)$ is #BIS-equivalent.*
2. *If f is not in IM_2, then for all sufficiently large Δ, there exists a real number $c > 1$ such that $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard.*

Theorems 17 and 18 together achieve the desired classification of $\#\mathsf{Multi2Spin}_\Delta(f)$ when $f \in IM_2$ as well as the strong inapproximability results when $f \notin IM_2$. Before presenting a more elaborate sketch of the proof of Theorem 17, it will be instructive to give the main ideas behind both proofs.

To prove Theorem 17, our proof departs from the previous approaches in the related works [10] and [11]. In these works, f was used to directly encode a binary function which was feasible because of the presence of equality in [10] and the symmetry of f in [11]. Instead, we take a much more painstaking combinatorial approach by using induction on the arity of the function f.

The base case of the induction (proving Theorem 17 for arity-2 functions) is fairly simple to handle, so let us focus on the induction step. The rough idea, to put the induction hypothesis to work, is to study whether f supports pinning-to-0 or pinning-to-1; then, provided that at least one these pinnings is available, we need to pin appropriately some arguments of f to obtain a function h of smaller arity. Our goal is then to ensure that h is non-affine; then, we can invoke the induction hypothesis and obtain that h either supports perfect equality or simulates a hard function. From there, since h was obtained by pinning some arguments of f, we will obtain by a transitivity argument (cf. Lemma 33 in the full version) that f either supports perfect equality or f simulates the same hard function as h. (Note, in the case where h supports perfect equality, to conclude that f supports perfect equality, we need to ensure that the pinnings of f used to obtain h were perfect.)

Determining which arguments of f need to be pinned is the most challenging aspect of this scheme. Our method for reducing the number of functions under consideration is to symmetrise f in a natural way and obtain a new function f^* which is now symmetric (see definitions in § 6). Then, it turns out that there are seven possibilities for the function f^* which we need to consider in detail. That is, when the symmetrisation of f is one of these seven functions, we have to figure out whether f supports perfect equality and, if not, work out the combinatorial structure of f and pinpoint which arguments are suitable to be pinned. The details of the argument can be found in the full version of this paper.

The proof of Theorem 18, where f supports perfect equality, basically follows the approach of [10]. However, to get the stronger inapproximability results, we have to take a detour studying self-dual functions (functions whose value does not change when we complement their arguments). We show that if f is self-dual then it simulates a hard function (Theorem 46 of the full version). The problem with self-dual functions is that they do not support pinning-to-0 or pinning-to-1, so we are not able to use the relevant results from [10]. After proving Theorem 46 and demonstrating (Lemma 42) that "implementations in CSPs" work in the repeat-free setting when f supports perfect equality, the techniques of [10] can be adapted to get Theorem 18.

6 A partial sketch of the proof of Theorem 17

Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. For $S \subseteq [k]$, χ_S denotes the characteristic vector of S, which is the length-k Boolean vector such that, for all $i \in [k]$, the i-th bit of χ_S is 1 iff $i \in S$. $\Omega_f = \{S \subseteq [k] \mid \chi_S \in R_f\}$. The function f is said to be *semi-trivial* iff there is a set S such that $\Omega_f = \{T \mid S \subseteq T \subseteq [k]\}$ or $\Omega_f = \{T \mid T \subseteq S\}$. Every semi-trivial Boolean function is affine. Let P_k denote the set of all permutations $\pi : [k] \to [k]$. Then the symmetrisation f^* of f is the function $f^* : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ defined by $f^*(x_1, \dots, x_k) = \prod_{\pi \in P_k} f(x_{\pi(1)}, \dots, x_{\pi(k)})$. For $s \in \{0,1\}$, let $\delta_s : \{0,1\} \to \{0,1\}$ be the Boolean function defined by $\delta_s(s) = 1$ and $\delta_s(1 \oplus s) = 0$. Define $f_{i \to s}$ to be the function obtained from f by pinning the i-th argument of f to s, i.e. $f_{i\to s}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) = \sum_{x_i \in \{0,1\}} f(x_1, \dots, x_k) \cdot \delta_s(x_i)$. Similarly, for $S, T \subseteq [k]$, let $f_{S\to 0, T\to 1}$ be the $(k - |S \cup T|)$-ary function obtained from f by pinning the arguments in S to 0 and the arguments in T to 1. So if $\mathbf{x}'$ denotes the $|S \cup T|$-ary vector containing all x_i with $i \in S \cup T$ and $\mathbf{x}''$ denotes the $k - |S \cup T|$-ary vector containing all x_i with $i \in [k] \setminus S \cup T$, $f_{S\to 0, T\to 1}(\mathbf{x}'') = \sum_{\mathbf{x}' \in \{0,1\}^{|S\cup T|}} f(x_1, \dots, x_k) \cdot \prod_{i \in S} \delta_0(x_i) \cdot \prod_{j \in T} \delta_1(x_j)$. If $S = \emptyset$ or $T = \emptyset$, we will omit $S \to 0$ or $T \to 1$ from the notation.

Partial Proof Sketch. We prove the theorem by induction on the arity of f. The base case, $k = 2$, is covered in the full version. For the induction step, assume $k \geq 3$. The inductive hypothesis is that for all $2 \leq k' < k$, all k'-ary functions f' satisfy at least one of the three propositions in the statement of the theorem. We now prove that an arbitrary $f : \{0,1\}^k \to \{0,1\}$ also satisfies at least one of the propositions. The easy case is when f^* is not affine. In this case, the work of [11] shows that f^* either simulates a hard function (in which case f simulates the hard function as well) or f^* supports perfect equality (in which case f does as well). The bulk of the proof deals with the case where f^* is affine. It turns out that there are seven possible symmetric affine functions f^*. To illustrate the ideas, we consider just one of them here. So from now on (to cover this special case), suppose that, for all $\mathbf{x} \in \{0,1\}^k$, $f^*(\mathbf{x}) = 0$.

We prove in the full version that either f supports perfect equality or f supports both perfect pinning-to-0 and perfect pinning-to-1. If f supports perfect equality, then we are done, so suppose from now on that f supports both perfect pinning-to-0 and perfect pinning-to-1.

We will show below that at least one of the following items holds. (1) f is affine (so we are finished), (2) there exist $S, T \subseteq [k]$ such that $f_{S\to 0, T\to 1}$ is not affine, (3) f supports perfect equality (so we are finished), or (4) f simulates a hard function (so we are finished). In situation (2), since $f_{S\to 0, T\to 1}$ is not affine, it must support perfect equality or simulate a hard function g by the induction hypothesis. So we finish by showing (Lemma 33) that f either supports perfect equality or simulates the same hard function g. We now discuss how to show that at least one of the four items holds.

We prove in the full version (Lemma 39) that for all $W \in \Omega_f$, $f_{\overline{W}\to 0}$ is semi-trivial (otherwise one of the items holds). Choose $S \in \Omega_f$ such that $|S|$ is as large as possible. Let $h = f_{\overline{S}\to 0}$. Since h is semi-trivial (by taking $W = S$ above), we claim that there is a T satisfying $\emptyset \subset T \subseteq S$ such that $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$. (To see this, note that the definition of semi-trivial implies that there is a subset T of S such that either $\Omega_h = \{U \mid U \subseteq T\}$ or $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$. The former is impossible since $\emptyset \notin \Omega_h$ since $h(\mathbf{0}) = f(\mathbf{0}) = f^*(\mathbf{0}) = 0$. Also, in the latter case, T is not empty because, once again, $\emptyset \notin \Omega_h$.)

Case 1. *Suppose that* $\forall X \in \Omega_f$, $T \subseteq X$*:* Recall that T is non-empty. Also, for every $i \in T$, $\{i\} \cup \Omega_{f_{i\to 1}} = \Omega_f$ so either f is affine (item (1)) or $f_{i\to 1}$ is not affine (item (2)).

Now, if Case 1 does not hold then there is an $X \in \Omega_f$ such that $T \setminus X$ is non-empty. Since $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ we conclude that $X \notin \Omega_h$. Since $h = f_{\overline{S}\to 0}$ we conclude that $X \setminus S$ is non-empty. Thus, the only other case to consider is as follows.

Case 2. *Suppose that there is an $X \in \Omega_f$ such that $T \setminus X$ and $X \setminus S$ are both non-empty:* Let $\Psi = \{X \in \Omega_f \mid T \setminus X \neq \emptyset \text{ and } X \setminus S \neq \emptyset\}$, $a = \min\{|T \setminus X| : X \in \Psi\}$, and $b = \min\{|X \setminus S| : X \in \Psi \text{ and } |T \setminus X| = a\}$. Choose $R \in \Psi$ with $|T \setminus R| = a$ and $|R \setminus S| = b$. Now before proceeding, we use the sets S, T and R to partition $[k]$. Specifically, let $A = \{i \in [k] \mid i \in S, i \in T, i \notin R\}$, $B = \{i \in [k] \mid i \in S, i \in T, i \in R\}$, $C = \{i \in [k] \mid i \in S, i \notin T, i \notin R\}$, $D = \{i \in [k] \mid i \in S, i \notin T, i \in R\}$, $E = \{i \in [k] \mid i \notin S, i \notin T, i \notin R\}$ and $F = \{i \in [k] \mid i \notin S, i \notin T, i \in R\}$. It is clear from the definitions that the sets A, B, C, D, E and F are disjoint. Also, since $T \subseteq S$, they partition $[k]$. From the definitions, $A = T \setminus R$ and $F = R \setminus S$ so, by the choice of R, A and F are non-empty. Let $g = f_{C \cup E \to 0, B \cup D \to 1}$.

By definition, every element of Ω_g is a subset of $A \cup F$. Also, for $Y \subseteq A \cup F$, "$Y \in \Omega_g$" means the same thing as "$Y \cup B \cup D \in \Omega_f$". We establish some facts before dividing the analysis into sub-cases.

Fact 1: $A \in \Omega_g$. We have $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ and $T = A \cup B$ so $A \cup B \cup D \in \Omega_h$. Since $A \cup B \cup D \subseteq S$, this means $A \cup B \cup D \in \Omega_f$. Equivalently, $A \in \Omega_g$.

Fact 2: $F \in \Omega_g$. From the definition of R, $R \in \Omega_f$. Also, $R = B \cup D \cup F$ so $F \cup B \cup D \in \Omega_f$. Equivalently, $F \in \Omega_g$.

Fact 3: If $Y \in \Omega_g$ then either $Y \cap A \in \{\emptyset, A\}$ or $Y \cap F = \emptyset$ (or both). Suppose for contradiction that $\emptyset \subset Y \cap A \subset A$ and $Y \cap F$ is non-empty. Note that $R = B \cup D \cup F$. Let $R' = B \cup D \cup Y$. Note that $T \setminus R = A$ and $T \setminus R' = A \setminus Y \subset A$ so $|T \setminus R'| < |T \setminus R|$. We will show a contradiction to the choice of R by showing that $R' \in \Psi$. First, since $Y \in \Omega_g$, $R' \in \Omega_f$. Also, $T \setminus R' = A \setminus Y$ is non-empty and $R' \setminus S = Y \cap F$ is non-empty.

Fact 4: If $Y \in \Omega_g$ and $Y \cap A = \emptyset$ then $Y \in \{\emptyset, F\}$. Suppose for contradiction that $\emptyset \subset Y \subset F$. As in the proof of Fact 3, let $R' = B \cup D \cup Y$. Note that $T \setminus R = T \setminus R' = A$. Also, $R \setminus S = F$ and $R' \setminus S = Y$ so $|R \setminus S| > |R' \setminus S|$. Once again, we will show a contradiction to the choice of R by showing that $R' \in \Psi$. As in the proof of Fact 3, since $Y \in \Omega_g$, $R' \in \Omega_f$. Also, $T \setminus R'$ is non-empty since $T \setminus R$ is. Finally, $R' \setminus S = Y$, which is non-empty.

Fact 5: If $Y \in \Omega_g$ and $Y \cap F = \emptyset$ then $Y = A$. Since $Y \in \Omega_g$, we have $Y \cup B \cup D \in \Omega_f$. But since $Y \subseteq A$, we have $Y \cup B \cup D \subseteq S$, so $Y \cup B \cup D \in \Omega_h$. Since $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ we have $T \subseteq Y \cup B \cup D$ so $A \subseteq Y$.

Given Facts 1–5, we have only the following sub-cases.

Case 2a: $\Omega_g = \{A, F\}$. In this case, we will show that f supports perfect equality. Let H_0 be a k-tuple hypergraph, with a vertex u_0 such that $\mu_{f;H_0}(\sigma_{u_0} = 0) = 1$. Let H_1 be a k-tuple hypergraph, with a vertex u_1 such that $\mu_{f;H_1}(\sigma_{u_1} = 0) = 1$. We have already noted that A is non-empty. Suppose, without loss of generality, that $1 \in A$ (otherwise, we simply re-order the arguments of $[k]$). Now let H' be the k-tuple hypergraph with vertices $v_0, v_1, \ldots, v_k$ and hyperarcs $(v_0, v_2, \ldots, v_k)$ and $(v_1, v_2, \ldots, v_k)$. Construct H from H' by doing the following:

- For every $i \in C \cup E$, take a new copy of H_0 and identify vertex u_0 with v_i.
- For every $i \in B \cup D$, take a new copy of H_1 and identify vertex u_1 with v_i.

Now since $\Omega_g = \{A, F\}$, $\mu_{f;H}(\sigma(v_0) = \sigma(v_1) = 0) = \mu_{f;H}(\sigma(v_0) = \sigma(v_1) = 1) = 1/2$. Thus, f supports perfect equality, so item (3) holds.

Case 2b: $\exists Y \in \Omega_g$ such that $Y \cap A = A$ and $Y \cap F$ is non-empty. We show in the full version that $f_{t \to 1}$ is not affine for some $t \in A$ (so item (2) holds). ◀

References

1 Ivona Bezáková, Andreas Galanis, Leslie A. Goldberg, Heng Guo, and Daniel Štefankovič. Approximation via Correlation Decay when Strong Spatial Mixing Fails. *ArXiv e-prints*, October 2015. `arXiv:1510.09193`.

2 Andrei A. Bulatov, Venkatesan Guruswami, Andrei Krokhin, and Dániel Marx. The Constraint Satisfaction Problem: Complexity and Approximability (Dagstuhl Seminar 15301). *Dagstuhl Reports*, 5(7):22–41, 2016. `doi:10.4230/DagRep.5.7.22`.

3 Jin-Yi Cai. Complexity dichotomy for counting problems. In *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 1–11, 2013. `doi:10.1007/978-3-642-37064-9_1`.

4 Jin-Yi Cai, Andreas Galanis, Leslie A. Goldberg, Heng Guo, Mark Jerrum, Daniel Štefankovič, and Eric Vigoda. #BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. *Journal of Computer and System Sciences*, 82(5):690–711, 2016. `doi:10.1016/j.jcss.2015.11.009`.

5 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted boolean #csp. *J. Comput. Syst. Sci.*, 80(1):217–236, 2014. `doi:10.1016/j.jcss.2013.07.003`.

6 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996. `doi:10.1006/inco.1996.0016`.

7 Víctor Dalmau and Daniel K. Ford. Generalized satisfability with limited occurrences per variable: A study through delta-matroid parity. In *Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings*, pages 358–367, 2003. `doi:10.1007/978-3-540-45138-9_30`.

8 Martin Dyer, Leslie A. Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003. `doi:10.1007/s00453-003-1073-y`.

9 Martin E. Dyer, Leslie A. Goldberg, Markus Jalsenius, and David Richerby. The complexity of approximating bounded-degree boolean #CSP. *Inf. Comput.*, 220:1–14, 2012. `doi:10.1016/j.ic.2011.12.007`.

10 Martin E. Dyer, Leslie A. Goldberg, and Mark Jerrum. An approximation trichotomy for boolean #csp. *J. Comput. Syst. Sci.*, 76(3-4):267–277, 2010. `doi:10.1016/j.jcss.2009.08.003`.

11 Andreas Galanis and Leslie A. Goldberg. The complexity of approximately counting in 2-spin systems on k-uniform bounded-degree hypergraphs. *Information and Computation*, 251:36–66, 2016.

12 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*. Oxford lecture series in mathematics and its applications. Oxford University Press, Oxford, New York, 2004. URL: `http://opac.inria.fr/record=b1121618`.

13 Jonathan Hermon, Allan Sly, and Yumeng Zhang. Rapid mixing of hypergraph independent set. *CoRR*, abs/1610.07999, 2016. URL: `http://arxiv.org/abs/1610.07999`.

14 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000. `doi:10.1006/jcss.2000.1713`.

15 Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992. URL: `http://www.aaai.org/ojs/index.php/aimagazine/article/view/976`.

16 Jingcheng Liu and Pinyan Lu. FPTAS for counting monotone CNF. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1531–1548, 2015. `doi:10.1137/1.9781611973730.101`.

17 Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974. `doi:10.1016/0020-0255(74)90008-5`.

18 Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

19 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226, 1978. doi:10.1145/800133.804350.

20 Renjie Song, Yitong Yin, and Jinman Zhao. Counting hypergraph matchings up to uniqueness threshold. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 46:1–46:29, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.46.

21 Dominic J. A. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, New York, NY, USA, 1993.