# The Complexity of Approximating Bounded-Degree Boolean #CSP

Martin Dyer, Leslie Ann Goldberg, Markus Jalsenius, David Richerby

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 323-334
www.stacs-conf.org

# THE COMPLEXITY OF APPROXIMATING BOUNDED-DEGREE BOOLEAN #CSP

MARTIN DYER [1] AND LESLIE ANN GOLDBERG [2] AND MARKUS JALSENIUS [2,3] AND DAVID RICHERBY [1]

[1] School of Computing, University of Leeds, Leeds, LS2 9JT, U.K.
*E-mail address*: {M.E.Dyer,D.M.Richerby}@leeds.ac.uk

[2] Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, U.K.
*E-mail address*: L.A.Goldberg@liverpool.ac.uk

[3] Current address: Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, U.K.
*E-mail address*: M.Jalsenius@bristol.ac.uk

Abstract. The degree of a CSP instance is the maximum number of times that a variable may appear in the scope of constraints. We consider the approximate counting problem for Boolean CSPs with bounded-degree instances, for constraint languages containing the two unary constant relations $\{0\}$ and $\{1\}$. When the maximum degree is at least 25 we obtain a complete classification of the complexity of this problem. It is exactly solvable in polynomial-time if every relation in the constraint language is affine. It is equivalent to the problem of approximately counting independent sets in bipartite graphs if every relation can be expressed as conjunctions of $\{0\}$, $\{1\}$ and binary implication. Otherwise, there is no FPRAS unless $\mathbf{NP} = \mathbf{RP}$. For lower degree bounds, additional cases arise in which the complexity is related to the complexity of approximately counting independent sets in hypergraphs.

## 1. Introduction

In the constraint satisfaction problem (CSP), we seek to assign values from some domain to a set of variables, while satisfying given constraints on the combinations of values that certain subsets of the variables may take. Constraint satisfaction problems are ubiquitous in computer science, with close connections to graph theory, database query evaluation, type inference, satisfiability, scheduling and artificial intelligence [20, 22, 25]. CSP can also be reformulated in terms of homomorphisms between relational structures [14] and conjunctive query containment in database theory [20]. Weighted versions of CSP appear in statistical physics, where they correspond to partition functions of spin systems [31].

*1998 ACM Subject Classification:* F.2.2, G.2.1.

*Key words and phrases:* Boolean constraint satisfaction problem, generalized satisfiability, counting, approximation algorithms.

Funded in part by the EPSRC grant "The Complexity of Counting in Constraint Satisfaction Problems".

We give formal definitions in Section 2 but, for now, consider an undirected graph $G$ and the CSP where the domain is $\{\text{red}, \text{green}, \text{blue}\}$, the variables are the vertices of $G$ and the constraints specify that, for every edge $xy \in G$, $x$ and $y$ must be assigned different values. Thus, in a satisfying assignment, no two adjacent vertices are given the same colour: the CSP is satisfiable if, and only if, the graph is 3-colourable. As a second example, given a formula in 3-CNF, we can write a system of constraints over the variables, with domain $\{\text{true}, \text{false}\}$, that requires the assignment to each clause to satisfy at least one literal. Clearly, the resulting CSP is directly equivalent to the original satisfiability problem.

### 1.1. Decision CSP

In the *uniform constraint satisfaction problem*, we are given the set of constraints explicitly, as lists of allowable combinations for given subsets of the variables; these lists can be considered as relations over the domain. Since it includes problems such as 3-SAT and 3-COLOURABILITY, uniform CSP is **NP**-complete. However, uniform CSP also includes problems in **P**, such as 2-SAT and 2-COLOURABILITY, raising the natural question of what restrictions lead to tractable problems. There are two natural ways to restrict CSP: we can restrict the form of the instances and we can restrict the form of the constraints.

The most common restriction to CSP is to allow only certain fixed relations in the constraints. The list of allowed relations is known as the *constraint language* and we write CSP($\Gamma$) for the so-called *non-uniform* CSP in which each constraint states that the values assigned to some tuple of variables must be a tuple in a specified relation in $\Gamma$.

The classic example of this is Schaefer's dichotomy for Boolean constraint languages $\Gamma$ (i.e., those with domain $\{0, 1\}$; often called "generalized satisfiability") [26]. He showed that CSP($\Gamma$) is in **P** if $\Gamma$ is included in one of six classes and is **NP**-complete, otherwise. More recently, Bulatov has produced a corresponding dichotomy for the three-element domain [2]. These two results restrict the size of the domain but allow relations of arbitrary arity in the constraint language. The converse restriction — relations of restricted arity, especially binary relations, over arbitrary finite domains — has also been studied in depth [16, 17].

For all $\Gamma$ studied so far, CSP($\Gamma$) has been either in **P** or **NP**-complete and Feder and Vardi have conjectured that this holds for every constraint language [14]. Ladner has shown that it is not the case that every problem in **NP** is either in **P** or **NP**-complete since, if $\mathbf{P} \neq \mathbf{NP}$, there is an infinite, strict hierarchy between the two [23]. However, there are problems in **NP**, such as graph Hamiltonicity and even connectedness, that cannot be expressed as CSP($\Gamma$) for any finite $\Gamma$ [1] and Ladner's diagonalization does not seem to be expressible in CSP [14], so a dichotomy for CSP appears possible.

Restricting the tree-width of instances has also been a fruitful direction of research [15, 21]. In contrast, little is known about restrictions on the degree of instances, i.e., the maximum number of times that any variable may appear. Dalmau and Ford have shown that, for any fixed Boolean constraint language $\Gamma$ containing the constant unary relations $R_{\text{zero}} = \{0\}$ and $R_{\text{one}} = \{1\}$, the complexity of CSP($\Gamma$) for instances of degree at most three is exactly the same as the complexity of CSP($\Gamma$) with no degree restriction [6]. The case where variables may appear at most twice has not yet been completely classified; it is known that degree-2 CSP($\Gamma$) is as hard as general CSP($\Gamma$) whenever $\Gamma$ contains $R_{\text{zero}}$ and $R_{\text{one}}$ and some relation that is not a $\Delta$-matroid [13]; the known polynomial-time cases come from restrictions on the kinds of $\Delta$-matroids that appear in $\Gamma$ [6].

[1] This follows from results on the expressive power of existential monadic second-order logic [12].

### 1.2. Counting CSP

A generalization of classical CSP is to ask how many satisfying solutions there are. This is referred to as counting CSP, #CSP. Clearly, the decision problem is reducible to counting: if we can efficiently count the solutions, we can efficiently determine whether there is at least one. The converse does not hold: for example, we can determine in polynomial time whether a graph admits a perfect matching but it is #**P**-complete to count the perfect matchings, even in a bipartite graph [29].

#**P** is the class of functions $f$ for which there is a nondeterministic, polynomial-time Turing machine that has exactly $f(x)$ accepting paths for input $x$ [28]. It is easily seen that the counting version of any **NP** decision problem is in #**P** and #**P** can be considered the counting "analogue" of **NP**. Note, though that problems that are #**P**-complete under appropriate reductions are, under standard complexity-theoretic assumptions, considerably harder than **NP**-complete problems: $\mathbf{P}^{\#\mathbf{P}}$ includes the whole of the polynomial hierarchy [27], whereas $\mathbf{P}^{\mathbf{NP}}$ is generally thought not to.

Although no dichotomy is known for CSP, Bulatov has recently shown that, for all $\Gamma$, #CSP($\Gamma$) is either computable in polynomial time or #**P**-complete [3]. However, Bulatov's dichotomy sheds little light on which constraint languages yield polynomial-time counting CSPs and which do not. The criterion of the dichotomy is based on "defects" in a certain infinite algebra built up from the polymorphisms of $\Gamma$ and it is open whether the characterization is even decidable. It also seems not to apply to bounded-degree #CSP.

So, although there is a full dichotomy for #CSP($\Gamma$), results for restricted forms of constraint language are still of interest. Creignou and Hermann have shown that only one of Schaefer's polynomial-time cases for Boolean languages survives the transition to counting: #CSP($\Gamma$) $\in$ **FP** (i.e., has a polynomial time algorithm) if $\Gamma$ is affine (i.e., each relation is the solution set of a system of linear equations over $\mathrm{GF}_2$) and is #**P**-complete, otherwise [5]. This result has been extended to rational and even complex-weighted instances [4,10] and, in the latter case, the dichotomy is shown to hold for the restriction of the problem in which instances have degree 3. This implies that the degree-3 problem #$\mathrm{CSP}_3(\Gamma)$ (#CSP($\Gamma$) restricted to instances of degree 3) is in **FP** if $\Gamma$ is affine and is #**P**-complete, otherwise.

### 1.3. Approximate counting

Since #CSP($\Gamma$) is very often #**P**-complete, approximation algorithms play an important role. The key concept is that of a *fully polynomial randomized approximation scheme* (FPRAS). This is a randomized algorithm for computing some function $f(x)$, taking as its input $x$ and a constant $\epsilon > 0$, and computing a value $Y$ such that $e^{-\epsilon} \leqslant Y/f(x) \leqslant e^{\epsilon}$ with probability at least $\frac{3}{4}$, in time polynomial in both $|x|$ and $\epsilon^{-1}$. (See Section 2.4.)

Dyer, Goldberg and Jerrum have classified the complexity of approximately computing #CSP($\Gamma$) for Boolean constraint languages [9]. When all relations in $\Gamma$ are affine, #CSP($\Gamma$) can be computed exactly in polynomial time by the result of Creignou and Hermann discussed above [5]. Otherwise, if every relation in $\Gamma$ can be defined by a conjunction of pins (i.e., assertions $v = 0$ or $v = 1$) and Boolean implications, then #CSP($\Gamma$) is as hard to approximate as the problem #BIS of counting independent sets in a bipartite graph; otherwise, #CSP($\Gamma$) is as hard to approximate as the problem #SAT of counting the satisfying truth assignments of a Boolean formula. Dyer, Goldberg, Greenhill and Jerrum have shown that the latter problem is complete for #**P** under appropriate approximation-preserving reductions (see Section 2.4) and has no FPRAS unless **NP** = **RP** [8], which is thought to

be unlikely. The complexity of #BIS is currently open: there is no known FPRAS but it is not known to be #**P**-complete, either. #BIS is known to be complete for a logically-defined subclass of #**P** with respect to approximation-preserving reductions [8].

### 1.4. Our result

We consider the complexity of approximately solving Boolean #CSP problems when instances have bounded degree. Following Dalmau and Ford [6] and Feder [13] we consider the case in which $R_{\text{zero}} = \{0\}$ and $R_{\text{one}} = \{1\}$ are available. We proceed by showing that any Boolean relation that is not definable as a conjunction of ORs or NANDs can be used in low-degree instances to assert equalities between variables. Thus, we can side-step degree restrictions by replacing high-degree variables with distinct variables asserted to be equal.

Our main result, Corollary 6.6, is a trichotomy for the case in which instances have maximum degree $d$ for some $d \geqslant 25$. If every relation in $\Gamma$ is affine, then $\#\text{CSP}_d(\Gamma \cup \{R_{\text{zero}}, R_{\text{one}}\})$ is solvable in polynomial time. Otherwise, if every relation in $\Gamma$ can be defined as a conjunction of $R_{\text{zero}}$, $R_{\text{one}}$ and binary implications, then $\#\text{CSP}_d(\Gamma \cup \{R_{\text{zero}}, R_{\text{one}}\})$ is equivalent in approximation complexity to #BIS. Otherwise, it has no FPRAS unless $\mathbf{NP} = \mathbf{RP}$. Theorem 6.5 gives a partial classification of the complexity when $d < 25$. In the new cases that arise here, the complexity is given in terms of the complexity of counting independent sets in hypergraphs with bounded degree and bounded hyper-edge size. The complexity of this problem is not fully understood and we explain what is known about it in Section 6.

## 2. Preliminaries

### 2.1. Basic notation

We write $\overline{a}$ for the tuple $\langle a_1, \ldots, a_r \rangle$, which we often shorten to $\overline{a} = a_1 \ldots a_r$. We write $a^r$ for the $r$-tuple $a \ldots a$ and $\overline{a}\overline{b}$ for the tuple formed from the elements of $\overline{a}$ followed by those of $\overline{b}$. The *bit-wise complement* of a relation $R \subseteq \{0,1\}^r$ is the relation $\widetilde{R} = \{\langle a_1 \oplus 1, \ldots, a_r \oplus 1 \rangle \mid \overline{a} \in R\}$, where $\oplus$ denotes addition modulo 2.

We say that a relation $R$ is *ppp-definable*[2] in a relation $R'$ and write $R \leqslant_{\text{ppp}} R'$ if $R$ can be obtained from $R'$ by some sequence of the following operations:

- permutation of columns (for notational convenience only);
- pinning (taking sub-relations of the form $R_{i \mapsto c} = \{\overline{a} \in R \mid a_i = c\}$ for some $i$ and some $c \in \{0,1\}$); and
- projection ("deleting the $i$th column" to give the relation $\{a_1 \ldots a_{i-1} a_{i+1} \ldots a_r \mid a_1 \ldots a_r \in R\}$).

It is easy to see that $\leqslant_{\text{ppp}}$ is reflexive and transitive and that, if $R \leqslant_{\text{ppp}} R'$, then $R$ can be obtained from $R'$ by first permuting the columns, then making some pins and then projecting.

We write $R_= = \{00, 11\}$, $R_{\neq} = \{01, 10\}$, $R_{\text{OR}} = \{01, 10, 11\}$, $R_{\text{NAND}} = \{00, 01, 10\}$, $R_{\to} = \{00, 01, 11\}$ and $R_{\leftarrow} = \{00, 10, 11\}$. For $k \geqslant 2$, we write $R_{=,k} = \{0^k, 1^k\}$, $R_{\text{OR},k} = \{0,1\}^k \setminus \{0^k\}$ and $R_{\text{NAND},k} = \{0,1\}^k \setminus \{1^k\}$ (i.e., $k$-ary equality, OR and NAND).

[2]This should not be confused with the concept of primitive positive definability (pp-definability) which appears in algebraic treatments of CSP and #CSP, for example in the work of Bulatov [3].

### 2.2. Boolean constraint satisfaction problems

A *constraint language* is a set $\Gamma = \{R_1, \ldots, R_m\}$ of named Boolean relations. Given a set $V$ of variables, the set of *constraints* over $\Gamma$ is the set $\mathrm{Cons}(V, \Gamma)$ which contains $R(\overline{v})$ for every relation $R \in \Gamma$ with arity $r$ and every $\overline{v} \in V^r$. Note that $v = v'$ and $v \neq v'$ are not constraints unless the appropriate relations are included in $\Gamma$. The *scope* of a constraint $R(\overline{v})$ is the tuple $\overline{v}$, which need not consist of distinct variables.

An *instance* of the constraint satisfaction problem (CSP) over $\Gamma$ is a set $V$ of variables and a set $C \subseteq \mathrm{Cons}(V, \Gamma)$ of constraints. An *assignment* to a set $V$ of variables is a function $\sigma\colon V \to \{0, 1\}$. An assignment to $V$ *satisfies* an instance $(V, C)$ if $\langle \sigma(v_1), \ldots, \sigma(v_r) \rangle \in R$ for every constraint $R(v_1, \ldots, v_r)$. We write $Z(I)$ for the number of satisfying assignments to a CSP instance $I$. We study the counting CSP problem $\#\mathrm{CSP}(\Gamma)$, parameterized by $\Gamma$, in which we must compute $Z(I)$ for an instance $I = (V, C)$ of CSP over $\Gamma$.

The *degree* of an instance is the greatest number of times any variable appears among its constraints. Note that the variable $v$ appears twice in the constraint $R(v, v)$. Our specific interest in this paper is in classifying the complexity of bounded-degree counting CSPs. For a constraint language $\Gamma$ and a positive integer $d$, define $\#\mathrm{CSP}_d(\Gamma)$ to be the restriction of $\#\mathrm{CSP}(\Gamma)$ to instances of degree at most $d$. Instances of degree 1 are trivial.

**Theorem 2.1.** *For any* $\Gamma$, $\#\mathrm{CSP}_1(\Gamma) \in \mathbf{FP}$. ∎

When considering $\#\mathrm{CSP}_d$ for $d \geqslant 2$, we follow established practice by allowing *pinning* in the constraint language [6, 13]. We write $R_{\mathrm{zero}} = \{0\}$ and $R_{\mathrm{one}} = \{1\}$ for the two singleton unary relations. We refer to constraints in $R_{\mathrm{zero}}$ and $R_{\mathrm{one}}$ as *pins*. To make notation easier, we will sometimes write constraints using constants instead of explicit pins. That is, we will allow the constants 0 and 1 to appear in the place of variables in the scopes of constraints. Such constraints can obviously be rewritten as a set of "proper" constraints, without increasing degree. We let $\Gamma_{\mathrm{pin}}$ denote the constraint language $\{R_{\mathrm{zero}}, R_{\mathrm{one}}\}$.

### 2.3. Hypergraphs

A *hypergraph* $H = (V, E)$ is a set $V = V(H)$ of vertices and a set $E = E(H) \subseteq \mathcal{P}(V)$ of non-empty *hyper-edges*. The *degree* of a vertex $v \in V(H)$ is the number $d(v) = |\{e \in E(H) \mid v \in e\}|$ and the degree of a hypergraph is the maximum degree of its vertices. If $w = \max\{|e| \mid e \in E(H)\}$, we say that $H$ has *width* $w$. An *independent set* in a hypergraph $H$ is a set $S \subseteq V(H)$ such that $e \nsubseteq S$ for every $e \in E(H)$. Note that an independent set may contain more than one vertex from any hyper-edge of size at least three.

We write $\#w$-HIS for the problem of counting the independent sets in a width-$w$ hypergraph $H$, and $\#w$-$\mathrm{HIS}_d$ for the restriction of $\#w$-HIS to inputs of degree at most $d$.

### 2.4. Approximation complexity

A *randomized approximation scheme* (RAS) for a function $f\colon \Sigma^* \to \mathbb{N}$ is a probabilistic Turing machine that takes as input a pair $(x, \epsilon) \in \Sigma^* \times (0, 1)$, and produces, on an output tape, an integer random variable $Y$ with $\Pr(e^{-\epsilon} \leqslant Y/f(x) \leqslant e^{\epsilon}) \geqslant \frac{3}{4}$.[3] A *fully polynomial randomized approximation scheme (FPRAS)* is a RAS that runs in time $\mathrm{poly}(|x|, \epsilon^{-1})$.

[3]The choice of the value $\frac{3}{4}$ is inconsequential: the same class of problems has an FPRAS if we choose any probability $p$ with $\frac{1}{2} < p < 1$ [18].

To compare the complexity of approximate counting problems, we use the AP-reductions of [8]. Suppose $f$ and $g$ are two functions from some input domain $\Sigma^*$ to the natural numbers and we wish to compare the complexity of approximately computing $f$ to that of approximately computing $g$. An *approximation-preserving* reduction from $f$ to $g$ is a probabilistic oracle Turing machine $M$ that takes as input a pair $(x, \epsilon) \in \Sigma^* \times (0, 1)$, and satisfies the following three conditions: (i) every oracle call made by $M$ is of the form $(w, \delta)$ where $w \in \Sigma^*$ is an instance of $g$, and $0 < \delta < 1$ is an error bound satisfying $\delta^{-1} \leqslant \text{poly}(|x|, \epsilon^{-1})$; (ii) $M$ is a randomized approximation scheme for $f$ whenever the oracle is a randomized approximation scheme for $g$; and (iii) the run-time of $M$ is polynomial in $|x|$ and $\epsilon^{-1}$.

If there is an approximation-preserving reduction from $f$ to $g$, we write $f \leqslant_{\text{AP}} g$ and say that $f$ is *AP-reducible* to $g$. If $g$ has an FPRAS, then so does $f$. If $f \leqslant_{\text{AP}} g$ and $g \leqslant_{\text{AP}} f$, then we say that $f$ and $g$ are *AP-interreducible* and write $f \equiv_{\text{AP}} g$.

## 3. Classes of relations

A relation $R \subseteq \{0, 1\}^r$ is *affine* if it is the set of solutions to some system of linear equations over $\text{GF}_2$. That is, there is a set $\Sigma$ of equations in variables $x_1, \ldots, x_r$, each of the form $x_{i_1} \oplus \cdots \oplus x_{i_n} = c$, where $\oplus$ denotes addition modulo 2 and $c \in \{0, 1\}$, such that $\overline{a} \in R$ if, and only if, the assignment $x_1 \mapsto a_1, \ldots, x_r \mapsto a_r$ satisfies every equation in $\Sigma$. Note that the empty and complete relations are affine.

We define IM-conj to be the class of relations defined by a conjunction of pins and (binary) implications. This class is called $\text{IM}_2$ in [9].

**Lemma 3.1.** *If $R \in$ IM-conj is not affine, then $R_{\to} \leqslant_{\text{ppp}} R$.* ∎

Let OR-conj be the set of Boolean relations that are defined by a conjunction of pins and ORs of any arity and NAND-conj the set of Boolean relations definable by conjunctions of pins and NANDs (i.e., negated conjunctions) of any arity. We say that one of the defining formulae of these relations is *normalized* if no pinned variable appears in any OR or NAND, the arguments of each individual OR and NAND are distinct, every OR or NAND has at least two arguments and no OR or NAND's arguments are a subset of any other's.

**Lemma 3.2.** *Every* OR*-conj (respectively,* NAND*-conj) relation is defined by a unique normalized formula.* ∎

Given the uniqueness of defining normalized formulae, we define the *width* of an OR-conj or NAND-conj relation $R$ to be $\text{wd}(R)$, the greatest number of arguments to any of the ORs or NANDs in the normalized formula that defines it. Note that, from the definition of normalized formulae, there are no relations of width 1.

**Lemma 3.3.** *If $R \in$* OR*-conj has width $w$, then $R_{\text{OR},2}, \ldots, R_{\text{OR},w} \leqslant_{\text{ppp}} R$. Similarly, if $R \in$* NAND*-conj has width $w$, then $R_{\text{NAND},2}, \ldots, R_{\text{NAND},w} \leqslant_{\text{ppp}} R$.* ∎

Given tuples $\overline{a}, \overline{b} \in \{0, 1\}^r$, we write $\overline{a} \leqslant \overline{b}$ if $a_i \leqslant b_i$ for all $i \in [1, r]$. If $\overline{a} \leqslant \overline{b}$ and $\overline{a} \neq \overline{b}$, we write $\overline{a} < \overline{b}$. We say that a relation $R \subseteq \{0, 1\}^r$ is *monotone* if, whenever $\overline{a} \in R$ and $\overline{a} \leqslant \overline{b}$, then $\overline{b} \in R$. We say that $R$ is *antitone* if, whenever $\overline{a} \in R$ and $\overline{b} \leqslant \overline{a}$, then $\overline{b} \in R$. Clearly, $R$ is monotone if, and only if, $\widetilde{R}$ is antitone. Call a relation *pseudo-monotone* (respectively, *pseudo-antitone*) if its restriction to non-constant columns is monotone (respectively, antitone). The following is a consequence of results in [19, Chapter 7.1.1].

**Proposition 3.4.** *A relation $R \subseteq \{0, 1\}^r$ is in* OR*-conj (respectively,* NAND*-conj) if, and only if, it is pseudo-monotone (respectively, pseudo-antitone).* ∎

## 4. Simulating equality

An important ingredient in bounded-degree dichotomy theorems [4] is expressing equality using constraints from a language that does not necessarily include the equality relation.

A constraint language $\Gamma$ is said to *simulate* the $k$-ary equality relation $R_{=,k}$ if, for some $\ell \geqslant k$, there is a $(\Gamma \cup \Gamma_{\text{pin}})$-CSP instance $I$ with variables $x_1, \ldots, x_\ell$ that has exactly $m \geqslant 1$ satisfying assignments $\sigma$ with $\sigma(x_1) = \cdots = \sigma(x_k) = 0$, exactly $m$ with $\sigma(x_1) = \cdots = \sigma(x_k) = 1$ and no other satisfying assignments. If, further, the degree of $I$ is $d$ and the degree of each variable $x_1, \ldots, x_k$ is at most $d-1$, we say that $\Gamma$ *$d$-simulates* $R_{=,k}$. We say that $\Gamma$ *$d$-simulates equality* if it $d$-simulates $R_{=,k}$ for all $k \geqslant 2$.

The point is that, if $\Gamma$ $d$-simulates equality, we can express the constraint $y_1 = \cdots = y_r$ in $\Gamma \cup \Gamma_{\text{pin}}$ and then use each $y_i$ in one further constraint, while still having an instance of degree $d$. The variables $x_{k+1}, \ldots, x_\ell$ in the definition function as auxiliary variables and are not used in any other constraint. Simulating equality makes degree bounds moot.

**Proposition 4.1.** *If $\Gamma$ $d$-simulates equality, then* $\#\text{CSP}(\Gamma) \leqslant_{\text{AP}} \#\text{CSP}_d(\Gamma \cup \Gamma_{\text{pin}})$. ∎

We now investigate which relations simulate equality.

**Lemma 4.2.** *$R \in \{0,1\}^r$ 3-simulates equality if $R_= \leqslant_{\text{ppp}} R$, $R_{\neq} \leqslant_{\text{ppp}} R$ or $R_{\to} \leqslant_{\text{ppp}} R$.*

*Proof.* For each $k \geqslant 2$, we show how to 3-simulate $R_{=,k}$. We may assume without loss of generality that the ppp-definition of $R_=$, $R_{\neq}$ or $R_{\to}$ from $R$ involves applying the identity permutation to the columns, pinning columns 3 to $3+p-1$ inclusive to zero, pinning columns $3+p$ to $3+p+q-1$ inclusive to one (that is, pinning $p \geqslant 0$ columns to zero and $q \geqslant 0$ to one) and then projecting away all but the first two columns.

Suppose first that $R_= \leqslant_{\text{ppp}} R$ or $R_{\to} \leqslant_{\text{ppp}} R$. $R$ must contain $\alpha \geqslant 1$ tuples that begin $000^p1^q$, $\beta \geqslant 0$ that begin $010^p1^q$ and $\gamma \geqslant 1$ that begin $110^p1^q$, with $\beta = 0$ unless we are ppp-defining $R_{\to}$. We consider, first, the case where $\alpha = \gamma$, and show that we can 3-simulate $R_{=,k}$, expressing the constraint $R_{=,k}(x_1, \ldots, x_k)$ with the constraints

$$R(x_1x_20^p1^q*),\ R(x_2x_30^p1^q*), \ldots,\ R(x_{k-1}x_k0^p1^q*),\ R(x_kx_10^p1^q*)\,,$$

where $*$ denotes a fresh $(r-2-p-q)$-tuple of variables in each constraint. These constraints are equivalent to $x_1 = \cdots = x_k = x_1$ or to $x_1 \to \ldots \to x_k \to x_1$ so constrain the variables $x_1, \ldots, x_k$ to have the same value, as required. Every variable appears at most twice and there are $\alpha^k$ solutions to these constraints that put $x_1 = \cdots = x_k = 0$, $\gamma^k = \alpha^k$ solutions with $x_1 = \cdots = x_k = 1$ and no other solutions. Hence, $R$ 3-simulates $R_{=,k}$, as required.

We now show, by induction on $r$, that we can 3-simulate $R_{=,k}$ even in the case that $\alpha \neq \gamma$. For the base case, $r = 2$, we have $\alpha = \gamma = 1$ and we are done. For the inductive step, let $r > 2$ and assume, w.l.o.g. that $\alpha > \gamma$ ($\alpha < \gamma$ is symmetric). In particular, we have $\alpha \geqslant 2$, so there are distinct tuples $000^p1^q\overline{a}$, and $000^p1^q\overline{b}$ and $110^p1^q\overline{c}$ in $R$. Choose $j$ such that $a_j \neq b_j$. Pinning the $(2+p+q+j)$th column of $R$ to $c_j$ and projecting out the resulting constant column gives a relation $R'$ of arity $r-1$ containing at least one tuple beginning $000^p1^q$ and at least one beginning $110^p1^q$: by the inductive hypothesis, $R'$ 3-simulates $R_{=,k}$.

Finally, we consider the case that $R_{\neq} \leqslant_{\text{ppp}} R$. $R$ contains $\alpha \geqslant 1$ tuples beginning $010^p1^q$ and $\beta \geqslant 1$ beginning $100^p1^q$. We express the constraint $R_{=,k}(x_1, \ldots, x_k)$ by introducing fresh variables $y_1, \ldots, y_k$ and using the constraints

$$\begin{array}{llll} R(x_1y_10^p1^q*), & R(x_2y_20^p1^q*), & \ldots, & R(x_{k-1}y_{k-1}0^p1^q*), \quad R(x_ky_k0^p1^q*), \\ R(y_1x_20^p1^q*), & R(y_2x_30^p1^q*), & \ldots, & R(y_{k-1}x_k0^p1^q*), \quad R(y_kx_10^p1^q*)\,. \end{array}$$

There are $\alpha^k\beta^k$ solutions when $x_1 = \cdots = x_k = 0$ (and $y_1 = \cdots = y_k = 1$) and $\beta^k\alpha^k$ solutions when the $x$s are 1 and the $y$s are 0. There are no other solutions and no variable is used more than twice. ∎

For $c \in \{0,1\}$, an $r$-ary relation is *$c$-valid* if it contains the tuple $c^r$.

**Lemma 4.3.** *Let $r \geqslant 2$ and let $R \subseteq \{0,1\}^r$ be 0- and 1-valid but not complete. Then $R$ 3-simulates equality.* ∎

In the following lemma, we do not require $R$ and $R'$ to be distinct. The technique is to assert $x_1 = \cdots = x_k$ by simulating the formula $\mathrm{OR}(x_1, y_1) \wedge \mathrm{NAND}(y_1, x_2) \wedge \mathrm{OR}(x_2, y_2) \wedge \mathrm{NAND}(y_2, x_3) \wedge \cdots \wedge \mathrm{OR}(x_k, y_k) \wedge \mathrm{NAND}(y_k, x_1)$.

**Lemma 4.4.** *If $R_{\mathrm{OR}} \leqslant_{\mathrm{ppp}} R$ and $R_{\mathrm{NAND}} \leqslant_{\mathrm{ppp}} R'$, then $\{R, R'\}$ 3-simulates equality.* ∎

## 5. Classifying relations

We are now ready to prove that every Boolean relation $R$ is in OR-conj, in NAND-conj or 3-simulates equality. If $R_0$ and $R_1$ are $r$-ary, let $R_0 + R_1 = \{0\overline{a} \mid \overline{a} \in R_0\} \cup \{1\overline{a} \mid \overline{a} \in R_1\}$.

**Lemma 5.1.** *Let $R_0, R_1 \in$ OR-conj and let $R = R_0 + R_1$. Then $R \in$ OR-conj, $R \in$ NAND-conj or $R$ 3-simulates equality.*

*Proof.* Let $R_0$ and $R_1$ have arity $r$. We may assume that $R$ has no constant columns. If it does, let $R'$ be the relation that results from projecting them away. $R' = R'_0 + R'_1$, where both $R'_0$ and $R'_1$ are OR-conj relations. By the remainder of the proof, $R' \in$ OR-conj, $R' \in$ NAND-conj or $R'$ 3-simulates equality. Re-instating the constant columns does not alter this. For $R$ without constant columns, there are two cases.

**Case 1.** $R_0 \subseteq R_1$. Suppose $R_i$ is defined by the normalized OR-conj formula $\phi_i$ in variables $x_2, \ldots, x_{r+1}$. Then $R$ is defined by the formula

$$\phi_0 \vee (x_1 = 1 \wedge \phi_1) \equiv (\phi_0 \vee x_1 = 1) \wedge (\phi_0 \vee \phi_1) \equiv (\phi_0 \vee x_1 = 1) \wedge \phi_1\,, \tag{5.1}$$

where the second equivalence is because $\phi_0$ implies $\phi_1$, because $R_0 \subseteq R_1$. $R_1$ has no constant column, since such a column would have to be constant with the same value in $R_0$, contradicting our assumption that $R$ has no constant columns. There are two cases.

**Case 1.1.** *$R_0$ has no constant columns.* $x_1 = 1$ is equivalent to $\mathrm{OR}(x_1)$ and $\phi_0$ contains no pins, so we can rewrite $\phi_0 \vee x_1 = 1$ in CNF. Therefore, (5.1) is OR-conj.

**Case 1.2.** *$R_0$ has a constant column.* Suppose first that the $k$th column of $R_0$ is constant-zero. $R_1$ has no constant columns, so the projection of $R$ onto its first and $(k+1)$st columns gives the relation $R_{\leftarrow}$, and $R$ 3-simulates equality by Lemma 4.2. Otherwise, all constant columns of $R_0$ contain ones. Then $\phi_0$ is in CNF, since every pin $x_i = 1$ in $\phi_0$ can be written $\mathrm{OR}(x_i)$. Thus, we can write $\phi_0 \vee x_1 = 1$ in CNF, so (5.1) defines an OR-conj relation.

**Case 2.** $R_0 \not\subseteq R_1$. We will show that $R$ 3-simulates equality or is in NAND-conj. We consider two cases (recall that no relation has width 1).

**Case 2.1.** *At least one of $R_0$ and $R_1$ has positive width.* There are two sub-cases.

**Case 2.1.1.** *$R_1$ has a constant column.* Suppose the $k$th column of $R_1$ is constant. If the $k$th column of $R_0$ is also constant, then the projection of $R$ to its first and $(k+1)$st columns is either equality or disequality (since the corresponding column of $R$ is not constant) so $R$ 3-simulates equality by Lemma 4.2. Otherwise, if the projection of $R$ to the first and $(k+1)$st

columns is $R_{\to}$, then $R$ 3-simulates equality by Lemma 4.2. Otherwise, that projection must be $R_{\mathrm{NAND}}$. By Lemma 3.3 and the assumption of Case 2.1, $R_{\mathrm{OR}}$ is ppp-definable in at least one of $R_0$ and $R_1$ so $R$ 3-simulates equality by Lemma 4.4.

**Case 2.1.2.** *$R_1$ has no constant columns.* By Proposition 3.4, $R_1$ is monotone. Let $\overline{a} \in R_0 \setminus R_1$: by applying the same permutation to the columns of $R_0$ and $R_1$, we may assume that $\overline{a} = 0^\ell 1^{r-\ell}$. We must have $\ell \geqslant 1$ as every non-empty $r$-ary monotone relation contains the tuple $1^r$. Let $\overline{b} \in R_1$ be a tuple such that $a_i = b_i$ for a maximal initial segment of $[1, r]$. By monotonicity of $R_1$, we may assume that $\overline{b} = 0^k 1^{r-k}$. Further, we must have $k < \ell$, since, otherwise, we would have $\overline{b} < \overline{a}$, contradicting our choice of $\overline{a} \notin R_1$.

Now, consider the relation $R' = \{a_0 a_1 \dots a_{\ell-k} \mid a_0 0^k a_1 \dots a_{\ell-k} 1^{r-\ell} \in R\}$, which is the result of pinning columns 2 to $(k+1)$ of $R$ to zero and columns $(r-\ell+1)$ to $(r+1)$ to one and discarding the resulting constant columns. $R'$ contains $0^{\ell-k+1}$ and $1^{\ell-k+1}$ but is not complete, since $10^{\ell-k} \notin R'$. By Lemma 4.3, $R'$ and, hence, $R$ 3-simulates equality.

**Case 2.2.** *Both $R_0$ and $R_1$ have width zero,* i.e., are complete relations, possibly padded with constant columns. For $i \in [1, r]$, let $R'_i$ be the relation obtained from $R$ by projecting onto its first and $(i+1)$st columns. Since $R$ has no constant columns, $R'_i$ is either complete, $R_=$, $R_{\neq}$, $R_{\mathrm{OR}}$, $R_{\mathrm{NAND}}$, $R_{\to}$ or $R_{\leftarrow}$. If there is a $k$ such that $R'_k$ is $R_=$, $R_{\neq}$, $R_{\to}$ or $R_{\leftarrow}$, then $R_=$, $R_{\neq}$ or $R_{\to}$ is ppp-definable in $R$ and hence $R$ 3-simulates equality by Lemma 4.2. If there are $k_1$ and $k_2$ such that $R'_{k_1} = R_{\mathrm{OR}}$ and $R'_{k_2} = R_{\mathrm{NAND}}$, then $R$ 3-simulates equality by Lemma 4.4. It remains to consider the following two cases.

**Case 2.2.1.** *Each $R'_i$ is either $R_{\mathrm{OR}}$ or complete.* $R_1$ must be complete, which contradicts the assumption that $R_0 \not\subseteq R_1$.

**Case 2.2.1.** *Each $R'_i$ is either $R_{\mathrm{NAND}}$ or complete.* $R_0$ must be complete. Let $I = \{i \mid R'_i = R_{\mathrm{NAND}}\}$. Then $R = \bigwedge_{i \in I} \mathrm{NAND}(x_1, x_{i+1})$, so $R \in$ NAND-conj. ∎

Using the duality between OR-conj and NAND-conj relations, we can prove the corresponding result for $R_0, R_1 \in$ NAND-conj. The proof of the classification is completed by a simple induction on the arity of $R$. Decomposing $R$ as $R_0 + R_1$ and assuming inductively that $R_0$ and $R_1$ are of one of the stated types, we use the previous results in this section and Lemma 4.4 to show that $R$ is.

**Theorem 5.2.** *Every Boolean relation is* OR-*conj or* NAND-*conj or 3-simulates equality.* ∎

## 6. Complexity

The complexity of approximating #CSP($\Gamma$) where the degree of instances is unbounded is given by Dyer, Goldberg and Jerrum [9, Theorem 3].

**Theorem 6.1.** *Let $\Gamma$ be a Boolean constraint language.*

- *If every $R \in \Gamma$ is affine, then* #CSP($\Gamma$) $\in$ **FP**.
- *Otherwise, if $\Gamma \subseteq$ IM-conj, then* #CSP($\Gamma$) $\equiv_{\mathrm{AP}}$ #BIS.
- *Otherwise,* #CSP($\Gamma$) $\equiv_{\mathrm{AP}}$ #SAT.

Working towards our classification of the approximation complexity of #CSP($\Gamma$), we first deal with subcases. The IM-conj case and OR-conj/NAND-conj cases are based on links between those classes of relations and the problems of counting independent sets in bipartite and general graphs, respectively [8, 9], the latter extended to hypergraphs.

**Proposition 6.2.** *If $\Gamma \subseteq$ IM-conj contains at least one non-affine relation, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$ for all $d \geqslant 3$.* ∎

**Proposition 6.3.** *Let $R$ be an* OR-*conj or* NAND-*conj relation of width $w$. Then, for $d \geqslant 2$, $\#w\text{-HIS}_d \leqslant_{\mathrm{AP}} \#\mathrm{CSP}_d(\{R\} \cup \Gamma_{\mathrm{pin}})$.* ∎

**Proposition 6.4.** *Let $R$ be an* OR-*conj or* NAND-*conj relation of width $w$. Then, for $d \geqslant 2$, $\#\mathrm{CSP}_d(\{R\} \cup \Gamma_{\mathrm{pin}}) \leqslant_{\mathrm{AP}} \#w\text{-HIS}_{kd}$, where $k$ is the greatest number of times that any variable appears in the normalized formula defining $R$.* ∎

We now give the complexity of approximating $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}})$ for $d \geqslant 3$.

**Theorem 6.5.** *Let $\Gamma$ be a Boolean constraint language and let $d \geqslant 3$.*

- *If every $R \in \Gamma$ is affine, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \in \mathbf{FP}$.*
- *Otherwise, if $\Gamma \subseteq$ IM-conj, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$.*
- *Otherwise, if $\Gamma \subseteq$ OR-conj or $\Gamma \subseteq$ NAND-conj, then let $w$ be the greatest width of any relation in $\Gamma$ and let $k$ be the greatest number of times that any variable appears in the normalized formulae defining the relations of $\Gamma$. Then $\#w\text{-HIS}_d \leqslant_{\mathrm{AP}} \#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \leqslant_{\mathrm{AP}} \#w\text{-HIS}_{kd}$.*
- *Otherwise, $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$.*

*Proof.* The affine case is immediate from Theorem 6.1. ($\Gamma \cup \Gamma_{\mathrm{pin}}$ is affine if, and only if, $\Gamma$ is.) Otherwise, if $\Gamma \subseteq$ IM-conj and some $R \in \Gamma$ is not affine, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$ by Proposition 6.2. Otherwise, if $\Gamma \subseteq$ OR-conj or $\Gamma \subseteq$ NAND-conj, then $\#w\text{-HIS}_d \leqslant_{\mathrm{AP}} \#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \leqslant_{\mathrm{AP}} \#w\text{-HIS}_{kd}$ by Propositions 6.3 and 6.4.

Finally, suppose that $\Gamma$ is not affine, $\Gamma \nsubseteq$ IM-conj, $\Gamma \nsubseteq$ OR-conj and $\Gamma \nsubseteq$ NAND-conj. Since $(\Gamma \cup \Gamma_{\mathrm{pin}})$ is neither affine or a subset of IM-conj, we have $\#\mathrm{CSP}(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$ by Theorem 6.1 so, if we can show that $\Gamma$ $d$-simulates equality, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{CSP}(\Gamma \cup \Gamma_{\mathrm{pin}})$ by Proposition 4.1 and we are done. If $\Gamma$ contains a $R$ relation that is neither OR-conj nor NAND-conj, then $R$ 3-simulates equality by Theorem 5.2. Otherwise, $\Gamma$ must contain distinct relations $R_1 \in$ OR-conj and $R_2 \in$ NAND-conj that are non-affine so have width at least two. So $\Gamma$ 3-simulates equality by Lemma 4.4. ∎

Unless $\mathbf{NP} = \mathbf{RP}$, there is no FPRAS for counting independent sets in graphs of maximum degree at least 25 [7], and, therefore, no FPRAS for $\#w\text{-HIS}_d$ with $r \geqslant 2$ and $d \geqslant 25$. Further, since #SAT is complete for $\#\mathbf{P}$ under AP-reductions [8], #SAT cannot have an FPRAS unless $\mathbf{NP} = \mathbf{RP}$. From Theorem 6.5 above we have the following corollary.

**Corollary 6.6.** *Let $\Gamma$ be a Boolean constraint language and let $d \geqslant 25$.*

- *If every $R \in \Gamma$ is affine, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \in \mathbf{FP}$.*
- *Otherwise, if $\Gamma \subseteq$ IM-conj, then $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$.*
- *Otherwise there is no FPRAS for $\#\mathrm{CSP}_d(\Gamma \cup \Gamma_{\mathrm{pin}})$, unless $\mathbf{NP} = \mathbf{RP}$.* ∎

$\Gamma \cup \Gamma_{\mathrm{pin}}$ is affine (respectively, in OR-conj or in NAND-conj) if, and only if $\Gamma$ is, so the case for large-degree instances ($d \geqslant 25$) corresponds exactly in complexity to the unbounded case [9]. The case for lower degree bounds is more complex. To put Theorem 6.5 in context, we summarize the known approximability of $\#w\text{-HIS}_d$, parameterized by $d$ and $w$.

The case $d = 1$ is clearly in $\mathbf{FP}$ (Theorem 2.1) and so is the case $d = w = 2$, which corresponds to counting independent sets in graphs of maximum degree two. For $d = 2$ and width $w \geqslant 3$, Dyer and Greenhill have shown that there is an FPRAS for $\#w\text{-HIS}_d$ [11]. For $d = 3$, they have shown that there is an FPRAS if the the width $w$ is at most 3.

| Degree $d$ | Width $w$ | Approximability of $\#w$-HIS$_d$ |
|---|---|---|
| 1 | $\geqslant 2$ | **FP** |
| 2 | 2 | **FP** |
| 2 | $\geqslant 3$ | FPRAS [11] |
| 3 | 2, 3 | FPRAS [11] |
| 3, 4, 5 | 2 | PTAS [30] |
| $6, \ldots, 24$ | $\geqslant 2$ | The MCMC method is likely to fail [7] |
| $\geqslant 25$ | $\geqslant 2$ | No FPRAS unless **NP** = **RP** [7] |

Table 1: Approximability of $\#w$-HIS$_d$ (still open for all other values of $d$ and $w$).

For larger width, the approximability of $\#w$-HIS$_3$ is still not known. With the width restricted to $w = 2$ (normal graphs), Weitz has shown that, for degree $d \in \{3, 4, 5\}$, there is a deterministic approximation scheme that runs in polynomial time (a PTAS) [30]. This extends a result of Luby and Vigoda, who gave an FPRAS for $d \leqslant 4$ [24]. For $d > 5$, approximating $\#w$-HIS$_d$ becomes considerably harder. More precisely, Dyer, Frieze and Jerrum have shown that for $d = 6$ the Monte Carlo Markov chain technique is likely to fail, in the sense that "cautious" Markov chains are provably slowly mixing [7]. They also showed that, for $d = 25$, there can be no polynomial-time algorithm for approximate counting, unless **NP** = **RP**. These results imply that for $d \in \{6, \ldots, 24\}$ and $w \geqslant 2$ the Monte Carlo Markov chain technique is likely to fail and for $d \geqslant 25$ and $w \geqslant 2$, there can be no FPRAS unless **NP** = **RP**. Table 1 summarizes the results.

Returning to bounded-degree #CSP, the case $d = 2$ seems to be rather different to degree bounds three and higher. This is also the case for decision CSP — recall that degree-$d$ CSP($\Gamma \cup \Gamma_{\text{pin}}$) has the same complexity as unbounded-degree CSP($\Gamma \cup \Gamma_{\text{pin}}$) for all $d \geqslant 3$ [6], while degree-2 CSP($\Gamma \cup \Gamma_{\text{pin}}$) is often easier than the unbounded-degree case [6,13] but the complexity of degree-2 CSP($\Gamma \cup \Gamma_{\text{pin}}$) is still open for some $\Gamma$.

Our key techniques for determining the complexity of $\#\text{CSP}_d(\Gamma \cup \Gamma_{\text{pin}})$ for $d \geqslant 3$ were the 3-simulation of equality and Theorem 5.2, which says that every Boolean relation is in OR-conj, in NAND-conj or 3-simulates equality. However, it seems that not all relations that 3-simulate equality also 2-simulate equality so the corresponding classification of relations does not appear to hold. It seems that different techniques will be required for the degree-2 case. For example, it is possible that there is no FPRAS for $\#\text{CSP}_3(\Gamma \cup \Gamma_{\text{pin}})$ except when $\Gamma$ is affine. However, Bubley and Dyer have shown that there is an FPRAS for degree-2 #SAT, even though the exact counting problem is #**P**-complete [1]. This shows that there is a class $\mathcal{C}$ of constraint languages for which $\#\text{CSP}_2(\Gamma \cup \Gamma_{\text{pin}})$ has an FPRAS for every $\Gamma \in \mathcal{C}$ but for which no exact polynomial-time algorithm is known.

We leave the complexity of degree-2 #CSP and of #BIS and the the various parameterized versions of the counting hypergraph independent sets problem as open questions.

## References

[1] R. Bubley and M. Dyer. Graph orientations with no sink and an approximation for a hard case of #SAT. In *8th ACM–SIAM Symp. on Discrete Algorithms (SODA 1997)*, pages 248–257, 1997.

[2] A. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element domain. *J. ACM*, 53(1):66–120, 2006.

[3] A. A. Bulatov. The complexity of the counting constraint satisfaction problem. In *35th Intl Colloq. on Automata, Languages and Programming (ICALP 2008) Part I*, volume 5125 of *LNCS*, pages 646–661. Springer, 2008.
[4] J.-Y. Cai, P. Lu, and M. Xia. The complexity of complex weighted Boolean #CSP. Upcoming journal submission, 2009.
[5] N. Creignou and M. Hermann. Complexity of generalized satisfiablility counting problems. *Inform. and Comput.*, 125(1):1–12, 1996.
[6] V. Dalmau and D. K. Ford. Generalized satisfiability with limited occurrences per variable: A study through Delta-matroid parity. In *Math. Founds of Comput. Sci. (MFCS 2003)*, volume 2747 of *LNCS*, pages 358–367. Springer, 2003.
[7] M. Dyer, A. Frieze, and M. Jerrum. On counting independent sets in sparse graphs. *SIAM J. Computing*, 31(5):1527–1541, 2002.
[8] M. Dyer, L. A. Goldberg, C. S. Greenhill, and M. Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.
[9] M. Dyer, L. A. Goldberg, and M. Jerrum. An approximation trichotomy for Boolean #CSP. To appear in *J. Comput. Sys. Sci.* http://arxiv.org/abs/0710.4272, 2007.
[10] M. Dyer, L. A. Goldberg, and M. Jerrum. The complexity of weighted Boolean CSP. *SIAM J. Comput.*, 38(5):1970–1986, 2009.
[11] M. Dyer and C. S. Greenhill. On Markov chains for independent sets. *J. Algorithms*, 35(1):17–49, 2000.
[12] R. Fagin, L. J. Stockmeyer, and M. Y. Vardi. On monadic NP vs monadic co-NP. *Inform. and Comput.*, 120(1):78–92, 1995.
[13] T. Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1–2):281–293, 2001.
[14] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.
[15] E. C. Freuder. Complexity of $k$-tree structured constraint satisfaction problems. In *8th Conf. of American Assoc. for Art. Intelligence*, pages 4–9. AAAI Press/MIT Press, 1990.
[16] P. Hell and J. Nešetřil. On the complexity of $h$-coloring. *J. Combin. Theory B*, 48(1):92–110, 1990.
[17] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
[18] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.
[19] D. E. Knuth. The Art of Computer Programming, Vol. 4A: Combinatorial Algorithms. In preparation.
[20] Ph. G. Kolaitis and M. Y. Vardi. Conjunctive query containment and constraint satisfaction. *J. Comput. Sys. Sci.*, 61(2):302–332, 2000.
[21] Ph. G. Kolaitis and M. Y. Vardi. A game-theoretic approach to constraint satisfaction. In *17th Conf. of American Assoc. for Artif. Intelligence*, pages 175–181. AAAI Press/MIT Press, 2000.
[22] V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):33–42, 1992.
[23] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.
[24] M. Luby and E. Vigoda. Fast convergence of the Glauber dynamics for sampling independent sets. *Random Structures and Algorithms*, 15(3–4):229–241, 1999.
[25] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–135, 1974.
[26] T. J. Schaefer. The complexity of satisfiability problems. In *10th ACM Symp. on Theory of Computing*, pages 216–226, 1978.
[27] S. Toda. On the computational power of PP and $\bigoplus$P. In *30th Ann. Symp. on Founds of Comput. Sci. (FOCS 1989)*, pages 514–519. IEEE Computer Society, 1989.
[28] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
[29] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.
[30] D. Weitz. Counting independent sets up to the tree threshold. In *38th ACM Symp. on Theory of Computing*, pages 140–149, 2006.
[31] D. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.