Counting problems and clones of funtions

Andrei A. Bulatov
Simon Fraser University
abulatov@cs.sfu.ca

Abstract

Counting solutions of various combinatorial problems is a well established and intensively studied area of theoretical computer science. Its applications range from networking to statistical physics. The central question in this area is the complexity of and algorithms for problems of this type. It turns out that in many important cases these questions can be answered with aid of certain clones of functions of multi-valued logic. We show the connection between the two areas and survey recent advances in this research direction.

Counting solutions of various combinatorial problems is a task ubiquitous in computer science, combinatorics and elsewhere. Applications of counting problems range across a variety of research areas from graph theory, to logic, to networking, to statistical physics, and quantum computing. The *Counting Constraint Satisfaction Problem* (#CSP for short) provides a powerful yet convenient formalism to express counting problems. In this paper we survey recent results on the complexity of #CSP and some related problems.

The central motif of this survey is the connection between the complexity of counting CSPs and clones of functions of multi-valued logic and co-clones of relations. This approach, sometimes referred to as the *algebraic approach*, has proved to be very efficient in the study of the decision CSPs, see, e.g., [2, 7, 3, 18]. We show how this approach can be applied to counting problems. In some cases, such as exact counting of solutions to CSPs, this conncetion allows one to classify completely the complexity of such problems and develop new solution algorithms. In other cases, such as approximate counting, the connection is weaker, allows one to employ only partial functions and co-clones of the corresponding type. In those cases many problems remain open.

1 Counting Problems

Definitions and examples. For a set D by D^n we denote the set of all n-*tuples* of elements of D. An n-ary relation is any set $R \subseteq D^n$. Tuples will be denoted in boldface, say, $\mathbf{a}$, and their entries denoted by $\mathbf{a}[1], \dots, \mathbf{a}[n]$. We will also need predicates corresponding to relations. To simplify the notation we use the same symbol for a relation and the corresponding predicate, for instance, for an n-ary relation R the corresponding predicate $R(x_1, \dots, x_n)$ is given by $R(\mathbf{a}[1], \dots, \mathbf{a}[n]) = 1$ if and only if $\mathbf{a} \in R$. A set of relations over the same set D is called a *constraint language* over D.

Definition 1 *An instance of the #CSP is a triple $\mathcal{P} = (V, D, \mathcal{C})$, where V is a finite set of* variables*, D is a set of possible* values *of the variables (also finite in this paper), and $\mathcal{C}$ is a collection of* constraints*. Each constraint $C = \langle \mathbf{s}, R\rangle$ is a pair consisting of an n_C-tuple $\mathbf{s}$ of variables, called the* constraint scope*, and an n_C-ary relation, called the* constraint relation*.*

A solution *of instance $\mathcal{P}$ is a mapping $\varphi\colon V \to D$ such that for each constraint $C = \langle \mathbf{s}, R\rangle$ the tuple $(\varphi(\mathbf{s}[1]), \dots, \varphi(\mathbf{s}[n_C]))$ belongs to R. The objective is to find the number of solutions to $\mathcal{P}$.*

In this paper we will mostly be concerned with problems of the form $\#\mathrm{CSP}(\Gamma)$, where Γ is a constraint language over a fixed finite set D. The restriction Γ imposes is that $\#\mathrm{CSP}(\Gamma)$ allows only those instances whose constraint relations belong to Γ.

Example 2 (3-SAT, [8, 9, 24, 23]) In the #3-SATISFIABILITY problem, given a propositional formula in the conjunctive normal form such that every clause contains exactly 3 literals, the goal is to find the number of satisfying assignments to this formula. It is not hard to see that #3-SAT is exactly the problem $\#\mathrm{CSP}(\Gamma_{3-\mathrm{SAT}})$, where $\Gamma_{3-\mathrm{SAT}}$ denotes the constraint language over $\{0, 1\}$ consisting of 8 ternary relations; each of these relations contains all the triples satisfying a clause with a certain combination of positive and negative literals.

Example 3 ($\#H$-Coloring, [13, 16, 19]) Let $H = (V, E)$ be a (di)graph. In the $\#H$-COLORING problem, given a (di)graph $G = (W, F)$, the goal is to find the number of *homomorphisms* from G to H, that is, mappings from W to V such that every edge of G is mapped into an edge of H. Observe that if H is a complete graph with k vertices then $\#H$-COLORING PROBLEM is the regular $\#k$-COLORING problem, in which the goal is find the number of proper k-colorings of a given graph G.

As is easily seen, the $\#H$-COLORING PROBLEM is equivalent to $\#\mathrm{CSP}(\{E\})$, that is #CSP over the constraint language containing only one relation, the edge relation of the (di)graph. The constraints then correspond to the edges in F.

Example 4 (Systems of linear equations) The objective in the #SYSTEMS OF LINEAR EQUATIONS problem is to count the number of solutions of a given system of linear equations over a finite field. This problem is equivalent to $\#\mathrm{CSP}(\Gamma_{\mathrm{Lin}})$, where Γ_{Lin} is the set of all relations representable as the solution space of a linear equation.

Example 5 (Fourier coefficients) Evaluating the quantum Fourier transform is one of the very few tasks that can be performed in polynomial time on a quantum computer, but not known to have an efficient classical algorithms. In the simplest form this problem can be viewed as a generalization of #3-SAT and can be expressed as follows. Let f be an n-ary Boolean function, $f\colon \{0,1\}^n \to \{0,1\}$. Let $S = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, *Fourier coefficient* $\hat{f}(S)$ of f is given by

$$\hat{f}(S) = \frac{1}{2^n} \sum_{(a_1,\ldots,a_n)\in\{0,1\}^n} (-1)^{f(a_1,\ldots,a_n)} \cdot (-1)^{a_{i_1}+\ldots+a_{i_k}}$$

(here all the computations are done over the reals). It is known that

$$(-1)^{f(x_1,\ldots,x_n)} = \sum_{S\subseteq\{1,\ldots,n\}} \hat{f}(S) \cdot (-1)^{\sum_{i\in S} x_i},$$

and the expression on the right side is called the *Fourier expansion* of f.

There is a quantum algorithm that given a set $S \subseteq \{1, \ldots, n\}$ 'computes' in a certain sense Fourier coefficient $\hat{f}(S)$. On the other hand, computing a Fourier coefficient reduces to computing the number of solutions of the equation $f(x_1, \ldots, x_n) + x_{i_1} + \ldots + x_{i_k} = 0$. Indeed, if we denote this number by N then $\hat{f}(S) = \frac{1}{2^n}(2N - 2^n)$. Now, the function f is usually given by a circuit, so if this circuit is a 3-CNF, computing the Fourier coefficient is equivalent to #3-SAT.

Example 6 (Max-Cut) Recall that a *cut* of a graph $G = (V, E)$ is a partition of V into two subsets, W_1, W_2. A cut is *maximal* if the number of edges whose ends belong to different sets W_1, W_2 is maximal possible. In the #MAX-CUT problem the goal is, given a graph G, find the number of maximal cuts.

Although this problem is best representable through so called *weighted CSP*, it also admits representation as a counting CSP. However, such representation requires certain technicalities and we omit it here. More about #MAX-CUT and counting CSPs can be found in [6].

Co-clones and weak co-clones. In an instance of a constraint satisfaction problem some possible combinations of values of variables are given explicitly by constraints, while other dependencies arise implicitly through interaction of constraints.

Example 7 ([2]) Let Γ be a constraint language containing a single binary relation R over the set $D = \{0, 1, 2\}$, where R is given by $R = \{(0,0), (0,1), (1,0), (1,2), (2,1), (2,2)\}$. Consider the following instance of $\#\mathrm{CSP}(\Gamma)$

$$\mathcal{P} = (\{v_1, v_2, v_3, v_4\}, D, \{C_1, C_2, C_3, C_4, C_5\}),$$

where $C_1 = \langle (v_1, v_2), R\rangle$, $C_2 = \langle (v_1, v_3), R\rangle$, $C_3 = \langle (v_2, v_3), R\rangle$, $C_4 = \langle (v_2, v_4), R\rangle$, $C_5 = \langle (v_3, v_4), R\rangle$. There is no explicit constraint on the pair (v_1, v_4). However, by considering all solutions to $\mathcal{P}$, it can be shown that the possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $R' = R \cup \{(1, 1)\}$.

The *co-clone* generated by a constraint language Γ includes all relations that can be implicitly expressed by an instance of $\#\mathrm{CSP}(\Gamma)$. More precisely,

Definition 8 *For a constraint language Γ over a set D, the set $\langle\langle\Gamma\rangle\rangle$ includes all relations that can be expressed (as a predicate) using (a) relations from Γ, together with the binary equality relation $=_D$ on D, (b) conjunctions, and (c) existential quantification. This set is called the* co-clone generated by Γ*, and every relation from $\langle\langle\Gamma\rangle\rangle$ is said to be* primitive positive *(*pp*-)* definable *in Γ.*

Weak co-clone generated by Γ *is obtained in a similar way by disallowing existential quantification. $\langle\Gamma\rangle$ includes all relations that can be expressed using (a) relations from Γ, together with $=_D$, and (b) conjunctions,*

Example 9 (1) The relation R' in Example 6 can be expressed as follows

$$R'(x, y) = \exists z, t(R(x, z) \wedge R(x, t) \wedge R(z, t) \wedge R(z, y) \wedge R(t, y)).$$

Hence, $R' \in \langle\langle\{R\}\rangle\rangle$.

(2) Let $\neq_D$ be the binary disequality relation on the set $D = \{0, 1, 2\}$, that is $\neq_D = \{(a, b) \in D^2 \mid a \neq b\}$ and Q the ternary disequality relation on the same set, $(a, b, c) \in R \iff \{a, b, c\} = \{0, 1, 2\}$. Then

$$Q(x, y, z) = (\neq_D (x, y) \wedge \neq_D (y, z) \wedge \neq_D (z, x)),$$

and therefore $Q \in \langle\{\neq\}\rangle$.

Galois connections. Co-clones and weak co-clones can often be conveniently and concisely represented through functions and partial functions, respectively.

Let R be a (k-ary) relation on a set D, and $f\colon D^n \to D$ an n-ary function on the same set. Function f *preserves* R, or is a *polymorphism* of R, if for any n tuples $\mathbf{a}_1, \ldots, \mathbf{a}_n \in R$ the tuple $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ obtained by component-wise application of f also belongs to R. Relation R in this case is said to be *invariant* with respect to f. The set of all functions that preserve every relation from a constraint language Γ is denoted by $\mathsf{Pol}(\Gamma)$, the set of all relations invariant with respect to a set of functions C is denoted by $\mathsf{Inv}(C)$.

Example 10 Let R be the solution space of a system of linear equations over a field F. Then the function $m(x, y, z) = x - y + z$ is a polymorphism of R. Indeed, let $A \cdot \mathbf{x} = \mathbf{b}$ be the system defining R, and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in R$. Then

$$A \cdot m(\mathbf{x}, \mathbf{y}, \mathbf{z}) = A \cdot (\mathbf{x} - \mathbf{y} + \mathbf{z}) = A \cdot \mathbf{x} - A \cdot \mathbf{y} + A \cdot \mathbf{z} = \mathbf{b}.$$

In fact, the converse can also be shown: if R is invariant under m then it is the solution space of a certain system of linear equations.

Operators Inv and Pol form a Galois connection between sets of functions and sets of relations. Sets of the form $\mathsf{Inv}(C)$ are precisely co-clones; on the operational side there is another type of closed sets.

A set of functions is said to be a *clone* of functions if it is closed under superpositions and contain all the *projection* functions, that is functions of the form $f(x_1, \ldots, x_n) = x_i$. It is known [10, 22] that sets of functions of the form $\mathsf{Pol}(\Gamma)$ are exactly clones of functions.

The study of the #CSP also makes use of another Galois connection, a connection between weak co-clones and sets of *partial functions*. An n-ary partial function f on a set D is just a partial mapping $f\colon D^n \to D$. As in the case of total functions, a partial function f *preserves* relation R, if for any n tuples $\mathbf{a}_1, \ldots, \mathbf{a}_n \in R$ the tuple $f(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ obtained by component-wise application of f is either undefined or belongs to R. The set of all partial functions that preserve every relation from a constraint language Γ is denoted by $\mathsf{pPol}(\Gamma)$.

The set of all tuples from D^n on which f is defined is called the *domain* of f and denoted by $\mathsf{dom}(f)$. A set of functions is said to be *down-closed* if along with a function f it contains any function f' such that $\mathsf{dom}(f') \subseteq \mathsf{dom}(f)$ and $f'(a_1, \ldots, a_n) = f(a_1, \ldots, a_n)$ for every tuple $(a_1, \ldots, a_n) \in \mathsf{dom}(f')$. A down-closed set of functions, containing all projections and closed under superpositions is called a *partial clone*. Partial clones are exactly the sets of the form $\mathsf{pPol}(\Gamma)$ for a certain Γ, and the weak co-clones are precisely the sets $\mathsf{Inv}(C)$ for collections C of partial functions [15].

2 Exact Counting

We now turn to the complexity of the #CSP.

Complexity and algorithms. Complexity classes of counting problems are somewhat similar to those for ordinary decision problems. We need two of them, the class FP of counting problems (actually, of problems computing a function) solvable in polynomial time, and the class #P of problems in which the goal is to find the number of solutions to a problem from NP. There are several standard types of reductions between counting problems. The most widely accepted one is *Turing* reduction. Suppose we have counting problems A and B. A Turing reduction is a polynomial time algorithm that solves A using B as an *oracle*, or a subroutine. A counting problem A is said to be *#P-complete* if any problem from #P is Turing reducible to A. More details about counting complexity classes and reductions can be found in [21]. For the purpose of this paper we consider problems from FP as easy (*tractable*), and #P-complete problems as hard. We assume that P$\neq$NP.

The complexity of the counting problems considered in Examples 1–5 is as follows: #3-SAT is #P-complete, the $\#H$-COLORING problem is tractable for certain graphs and #P-complete otherwise, see Theorem 4, #SYSTEM OF LINEAR EQUATIONS is tractable, computing Fourier coefficient is tractable if and only if f can be represented by a quadratic polynomial over the 2-element field, and #P-complete otherwise, finally #MAX-CUT is #P-complete.

Boolean #CSP. The simplest type of constraint languages are those over a 2-element set. They are usually referred to as *Boolean* constraint languages. For a Boolean language Γ the problem $\#\mathrm{CSP}(\Gamma)$ can be thought of as a generalized version of the SATISFIABILITY problem.

Theorem 11 (Creignou, Hermann [8, 9]) *Let Γ be a Boolean constraint language. Problem $\#\mathrm{CSP}(\Gamma)$ is tractable if and only if every relation from Γ is the set of all solutions to a system of linear equations over the 2-element field. Otherwise it is #P-complete.*

Observe that the latter condition holds if and only if the *affine* operation $x - y + z$ (addition and subtraction modulo 2) is a polymorphism of Γ.

Corollary 12 *For a Boolean constraint language Γ the problem $\#\mathrm{CSP}(\Gamma)$ is tractable if and only if $\mathsf{Pol}(\Gamma)$ contains the affine operation.*

Non-Boolean #CSP. For non-Boolean #CSPs researchers have mostly focused on the $\#H$-COLORING problem.

Theorem 13 (Dyer, Greenhill [13]) *The $\#H$-COLORING problem for an undirected graph H is tractable if and only if every connected component of H is a single vertex, or a complete bipartite graph without loops, or a complete graph with all loops present. Otherwise it is #P-complete. Moreover, the #P-complete problems remain #P-complete even if restricted to graphs of degree at most 3.*

In [1, 5] it is shown that an undirected graph satisfies the conditions of Theorem 4 if and only if its edge relation has a *Mal'tsev* operation as a polymorphism, that is a function m satisfying equations $m(x, y, y) = m(y, y, x) = x$ for any x, y.

A *directed acyclic graph* (DAG) is a digraph without directed cycles.

Theorem 14 (Dyer, Goldberg, Paterson [11]) *For any DAG H the $\#H$-COLORING problem is either tractable or #P-complete. The DAGs that give rise to polynomial time solvable problems are those satisfying the* Lovasz-goodness *condition, see, [11] for details.*

Observe that in all the cases considered above every counting CSP is either tractable or #P-complete. This suggests a natural question: Is it true for any problem of the form $\#\mathrm{CSP}(\Gamma)$? This question (similar questions also arise for other types of constraint satisfaction problems), or rather a positive resolution of it, are known as *dichotomy conjecture* for counting problems. In the rest of this section we survey a sequence of results that eventually lead to a proof of the dichotomy conjecture.

Although all the results listed above are very non-trivial, the methods they use strongly depend on the nature of problems and the type of digraphs considered. To attack the dichotomy conjecture a more universal approach is needed. To develop such approach we make use of clones and coclones.

Clones and complexity. The following theorem shows a link between the complexity of counting CSPs and clones.

Theorem 15 (Bulatov, Dalmau [1, 5]) *Let Γ_1, Γ_2 be constraint languages over the same finite set such that Γ_2 is finite and $\Gamma_2 \subseteq \langle\langle\Gamma_1\rangle\rangle$. Then $\#\mathrm{CSP}(\Gamma_2)$ is Turing reducible to $\#\mathrm{CSP}(\Gamma_1)$. In particular, if $\#\mathrm{CSP}(\Gamma_1)$ is tractable then so is $\#\mathrm{CSP}(\Gamma_2)$; similarly, if $\#\mathrm{CSP}(\Gamma_2)$ is #P-complete then so is $\#\mathrm{CSP}(\Gamma_1)$.*

Due to the Galois connection between clones and coclones Theorem 6 can be restated as follows.

Corollary 16 *Let Γ_1, Γ_2 be constraint languages over the same finite set and Γ_2 is finite. If $\mathsf{Pol}(\Gamma_1) \subseteq \mathsf{Pol}(\Gamma_2)$ then $\#\mathrm{CSP}(\Gamma_2)$ is Turing reducible to $\#\mathrm{CSP}(\Gamma_1)$.*

Adding several other algebraic results we obtain the first general necessary condtion for tractability of counting CSPs.

Theorem 17 (Bulatov, Dalmau [1, 5]) *If Γ does not have a Mal'tsev polymorphism then $\#\mathrm{CSP}(\Gamma)$ is #P-complete.*

Congruences and a dichotomy theorem. Theorem 7 implies almost straightforwardly Theorems 3 and 4. However, the existence of a Mal'tsev polymorphism is not a sufficient condition for tractability as the following example shows.

Example 18 Consider the graphs shown in Fig. 1. They look very similar and they both have a Mal'tsev polymorphism. Intuitively, such polymorphism acts as $x - y + z$ on each of the components of the pairs (i, j) corresponding to the vertices in the middle layer of the graphs, and then extended straightforwardly on the vertices of other layers. (Clearly for the second graph one should be careful with the two copies of the leftmost vertex, but this obstacle can easily be handled.) However, the $\#H_1$-COLORING problem is tractable, while #MAX-CUT is reducible to $\#H_2$-COLORING, and therefore it is #P-complete.

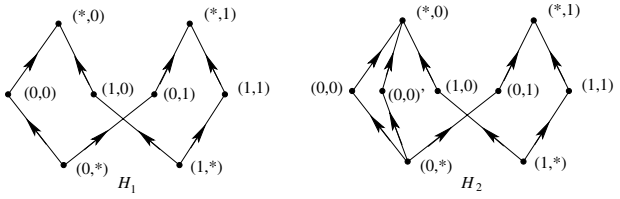

Figure 1.

Let Γ be a constraint language over a set D and $R \in \langle\langle\Gamma\rangle\rangle$ a k-ary relation. A *congruence* of R is a $2k$-ary relation $Q \subseteq \langle\langle\Gamma\rangle\rangle$ and satisfying the following conditions: (a) for any $\mathbf{a} \in Q$ the tuples $(\mathbf{a}[1], \ldots, \mathbf{a}[k])$ and $(\mathbf{a}[k+1], \ldots, \mathbf{a}[2k])$ belong to R, thus, Q can be viewed as a binary relation on R; (b) Q viewed as a binary relation on R is an equivalence relation. Now, let Q, Q_1, Q_2 be congruences of R such that $Q \subseteq Q_1, Q_2$, and let $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ be equivalence classes of Q_1

and Q_2, respectively. By M_{ij} we denote the number of Q-classes in $A_i \cap B_j$, and $M(R; Q_1, Q_2; Q)$ denotes the matrix (M_{ij}). Constraint language Γ is said to be *congruence singular* if, for any relation $R \in \langle\langle\Gamma\rangle\rangle$, and any congruences Q, Q_1, Q_2 of R with $Q \subseteq Q_1, Q_2$, the *row rank* of the matrix $M(R; Q_1, Q_2; Q)$ equals the number of classes of the smallest equivalence relation containing both Q_1 and Q_2.

Theorem 19 (Bulatov [4]) *The problem* $\#\mathrm{CSP}(\Gamma)$ *is tractable if and only if* Γ *is congruence singular; otherwise it is #P-complete.*

Example 20 (Example 9 continued) Let $H_2 = (V, E)$. The unary relation $C = \{(0,0), (0,0)', (1,0), (0,1), (1,1)\}$ is pp-definable in $\{E\}$; indeed, $C(x) = \exists y, z(E(y,x) \wedge E(x,z))$. The equivalence relations on this set Q_1, Q_2 with classes $A_1 = \{(0,0), (0,0)', (0,1)\}$, $A_2 = \{(1,0), (1,1)\}$, and $B_1 = \{(0,0), (0,0)', (1,0)\}$, $B_2 = \{(0,1), (1,1)\}$, respectively, are congruences of C; say,

$$Q_1(x,y) = \exists z(C(x) \wedge C(y) \wedge E(z,x) \wedge E(z,y)).$$

Then $M(C; Q_1, Q_2; =_C) = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$. Clearly its rank equals 2, while the smallest equivalence relation on C containing both Q_1 and Q_2 is the total relation and so has only one class. Therefore $\#H_2$-COLORING problem is #P-complete.

3 Approximation

As it is seen from the results above, tractable counting CSPs are rare. Other efficiently solvable counting problems are also scarce. Of those that do not fall under one of the CSP type considered two can be mentioned: the #SPANNING TREE problem and the #PLANAR GRAPH MATCHING. Observe that both are reducible to solving systems of linear equations. This is why approximation algorithms are so important in this area. However, the study of approximation of counting problems is in its cradle. Here we list several initial results in this direction.

Relative error and classes of problems. Besides algorithms that find the exact number of solutions, *approximation* and *randomized* algorithms are used to solve counting problems. Let A be a counting problem. An algorithm Alg is said to be an *approximation algorithm* for A with relative error ε (which may depend on the size of the input) if it is polynomial time and for any instance I of A it outputs a certain number $\mathsf{Alg}(I)$ such that

$$\frac{|\#I - \mathsf{Alg}(I)|}{\#I} < \varepsilon,$$

where $\#I$ denotes the exact number of solutions to I. Algorithm Alg is *randomized* approximation algorithm, if it uses some source of randomness and for any instance I the above inequality is true with probability at least $\frac{2}{3}$.

The following framework is viewed as one of the most realistic models of effitient computations. A *fully polynomial approximation scheme* (FPAS, for short) for A is an algorithm Alg such that for instance I of A and any $\varepsilon > 0$ the relative error of Alg on the input I and ε is less than ε, and Alg is polynomial in the size of I and $\frac{1}{\varepsilon}$. A FPAS is called a *fully polynomial randomized approximation scheme* (FPRAS) if Alg is randomized and the above inequality is satisfied with probability at least $\frac{2}{3}$.

To determine the approximation complexity of problems yet another type of reductions is used. For a precise definition of *AP-reductions* the reader is referred to [12], we just mention that it is a randomized polynomial time algorithm that respects the relative error of solutions.

Approximation complexity of #CSP. Up to AP-reductions all #CSPs whose complexity is known cluster into 4 groups.

The first group consists of those problems solvable in polynomial time, and the members of this group are completely described in the previous section.

The second group is formed by problems solved by an FPRAS, and it contains problems #DUAL SAT, in which the goal is to count satisfying assignments of a propositional formula in DNF, and $\#H$-COLORING on graphs with bounded vertex degree [17, 20].

A representative of the third group is #BIPARTITE INDEPENDENT SET, in which the goal is to find the number of *independent sets* in a given bipartite graph. The #BIPARTITE INDEPENDENT SET problem can be represented as the $\#H$-COLORING problem, where H is a path of length 3. Every independent set of a connected bipartite graph G corresponds to two homomorphisms from G to H, while a non-bipartite graph does not have homomorphisms to H.

Finally, the fourth group contains most difficult problems in #P, they are interreducible with #SAT.

In the case of Boolean constraint languages Γ the clustering above can be refined to a complete characterization of approximation complexity of $\#\mathrm{CSP}(\Gamma)$. By C_0, C_1 we denote unary relations over $\{0,1\}$ that have only one element 0 or 1, respectively. By O we denote the binary relation which is the natural order on $\{0,1\}$.

Theorem 21 (Dyer, Goldberg, Jerrum [14]) *Let* Γ *be a constraint language over* $\{0,1\}$. *If every relation in* Γ *is the set of solutions of a system of linear equations,* $\#\mathrm{CSP}(\Gamma)$ *is tractable. Otherwise if* $\Gamma \subseteq \langle\langle C_0, C_1, O\rangle\rangle$ *then* $\#\mathrm{CSP}(\Gamma)$ *is interreducible with* #BIPARTITE INDEPENDENT SET. *Otherwise* $\#\mathrm{CSP}(\Gamma)$ *is interreducible with* #SAT.

Theorem 9 hints that the approximation complexity of counting CSPs may be again determined by the co-clone generated by a constraint language (or its polymorphisms). However, it remains an open question whether or not an analog of Theorem 6 is true. We can, however, prove a weaker result.

Theorem 22 *Let Γ_1, Γ_2 be constraint languages over the same finite set such that Γ_2 is finite and $\Gamma_2 \subseteq \langle\Gamma_1\rangle$. Then $\#\mathrm{CSP}(\Gamma_2)$ is AP-reducible to $\#\mathrm{CSP}(\Gamma_1)$.*

Due to the Galois connection between sets of relations and sets of partial clones, partial functions are in control of the approximation complexity of counting problems.

References

[1] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *FOCS*, pages 562–571, 2003.

[2] A. Bulatov, P. Jeavons, and A. Krokhin. Functions of multiple-valued logic and the complexity of constraint satisfaction: A short survey. In *ISMVL*, pages 343–351, 2003.

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

[4] A. Bulatov. The complexity of the counting constraint satisfaction problem. In *ICALP (1)*, pages 646–661, 2008.

[5] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inf. and Comp.*, 205(5):651–678, 2007.

[6] A. Bulatov and M. Grohe. The complexity of partition functions. *Theor. Comput. Sci.*, 348(2-3):148–186, 2005.

[7] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.

[8] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

[9] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

[10] K. Denecke and S.L. Wismath. *Universal algebra and applications in Theoretical Computer Science*. Chapman and Hall/CRC Press, 2002.

[11] M. Dyer, L. Goldberg, and M. Paterson. On counting homomorphisms to directed acyclic graphs. In *ICALP*, pages 38–49, 2006.

[12] M. Dyer, L.A. Goldberg, C. Greenhill, and M. Jerrum. On the relative complexity of approximate counting problems. In *APPROX*, volume 1913 of *LNCS*, pages 108–119. Springer-Verlag, 2000.

[13] M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17:260–289, 2000.

[14] M. Dyer, L.A. Goldberg, and M. Jerrum. An approximation trichotomy for boolean #csp. *CoRR*, abs/0710.4272, 2007.

[15] I. Fleischer and I.G. Rosenberg. The Galois connection between partial operations and relations. *Pacific J. Math.*, 79:93–97, 1978.

[16] P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. of Comb. Theor., Ser.B*, 48:92–110, 1990.

[17] M. Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Struct. Algorithms*, 7(2):157–166, 1995.

[18] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. In *LICS*, pages 201–210, 2006.

[19] L.A. Levin. Universal enumeration problems. *Problems on Information Transmission*, 9:265–266, 1973.

[20] M. Luby and B. Velickovic. On deterministic approximation of dnf. *Algorithmica*, 16(4/5):415–433, 1996.

[21] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[22] R. Pöschel and L.A. Kalužnin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.

[23] L. Valiant. The complexity of computing the permanent. *Theoretical Computing Science*, 8:189–201, 1979.

[24] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.