National and Kapodistrian University of Athens

Department of Mathematics
Graduate Program in Logic and Theory of Algorithms and Computation

Complexity Dichotomies for Approximations of Counting Problems

Master Thesis
of
Andreas-Nikolas Göbel

Supervisor: Stathis Zachos
Professor

July 2012

Η παρούσα Διπλωματική Εργασία
εκπονήθηκε στα πλαίσια των σπουδών
για την απόκτηση του
Μεταπτυχιακού Διπλώματος Ειδίκευσης
στη
Λογική και Θεωρία Αλγορίθμων και Υπολογισμού
που απονέμει το
Τμήμα Μαθηματικών
του
Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 23η Ιουλίου 2012 από Εξεταστική Επιτροπή
αποτελούμενη από τους:

Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1. : Ε. Ζάχος	Καθηγητής	.....
2. : Αρ. Παγουρτζής	Επίκ. Καθηγητής	.....
3. : Δ. Φωτάκης	Λέκτορας	.....

Abstract

This thesis is a survey of dichotomy theorems for computational problems, focusing in counting problems. A dichotomy theorem in computational complexity, is a complete classification of the members of a class of problems, in computationally easy and computationally hard, with the set of problems of intermediate complexity being empty. Due to Ladner's theorem we cannot find a dichotomy theorem for the whole classes NP and #P, however there are large subclasses of NP (#P), that model many "natural" problems, for which dichotomy theorems exist.

We continue with the decision version of constraint satisfaction problems (CSP), a class of problems in NP. for which Ladner's theorem doesn't apply. We obtain a dichotomy theorem for some special cases of CSP. We then focus on counting problems presenting the following frameworks: graph homomorphisms, counting constraint satisfaction (#CSP) and Holant problems; we provide the known dichotomies for these frameworks.

In the last and main chapter of this thesis we relax the requirement of exact computation, and settle in approximating the problems. We present the known cassification theorems for cases of #CSP. Many questions in terms of approximate counting problems remain open.

The appendix introduces a recent technique for obtaining exact polynomial-time algorithms for counting problems, namely the holographic algorithms.

Acknowledgements

I am truly grateful to my advisor Stathis Zachos, for putting up with me, and guiding me all these years. Without his influence I wouldn't have made it this far.

I would like to express my gratitude to Aris Pagourtzis for giving me the opportunity to work with him in the field of computational complexity. He was always there for me whenever I needed his help.

I wish to thank Dimitris Fotakis for always providing me with up-to-the-point advice, and believing in me.

Part of this thesis was joint work with Stelios Despotakis. Without him this thesis wouldn't have been completed.

During the preparation of this thesis I received help (not only in subjects related to this thesis), and useful comments form the following members of Corelab (in random order): Charis, Helen, Themis, Antony, Paris.

I also wish to thank the rest of Corelab members for their fruitful discussions (in seniority order): Katerina, Petros, Georgia, Thanasis, Chris, Christina, Dimitris, Matoula, Sotiris, Markos, Theodore, Lydia.

I am most grateful to my parents for supporting me in multiple ways all the years of my study.

Contents

Chapter 1

Introduction and Motivation

This chapter begins with some basic definitions we will use later on. It continues with Ladner's theorem and its variations, which is the reason we will restrict our classes of problems in later chapters. Finally we present examples of dichotomy theorems in computational complexity.

1.1 Basic Definitions

Computational Complexity classify problems into complexity classes. In this section we are going to define some complexity classes and reductions which are used in this theses. For more details we refer the reader to [2, 43].

When restricting the time recourses of a Turing Machine we have the following definition:

Definition 1.1.1. *Let f be a function from $\mathbb{N}$ to $\mathbb{N}$. We say that the TM M operates within time $f(n)$ if, for any input string x, the time required by M on x is at most $f(|x|)$* [1]*. Suppose that a language $L \subseteq (\Sigma - \{\sqcup\})^*$ is decided by a TM operating in time $f(n)$. We say that $L \in \mathrm{TIME}(f(n))$.*

And when we restrict the working space we get:

Definition 1.1.2. *Suppose that f is a function from $\mathbb{N}$ to $\mathbb{N}$. We say that the TM M operates within space bound $f(n)$, if for any input x, M requires space at most $f(|x|)$. Let L be a language. $L \in \mathrm{SPACE}(f(n))$ if there is a TM that decides L with the use of $f(n)$ cells on its work tape.*

For nondeterministic TM's we denote the time and space bound complexity classes with $\mathrm{NTIME}(f(n))$ and $\mathrm{NPACE}(f(n))$ respectively. We have the following complexity classes:

$\mathrm{L} = \mathrm{SPACE}(\log n)$
$\mathrm{NL} = \mathrm{NSPACE}(\log n)$

[1] $|x|$ denotes the length of the string x

$\mathrm{P} = \bigcup_{k>0} \mathrm{TIME}(n^k)$
$\mathrm{NP} = \bigcup_{k>0} \mathrm{NTIME}(n^k)$
$\mathrm{EXP} = \bigcup_{k>0} \mathrm{TIME}(2^{n^k})$
$\mathrm{PSPACE} = \bigcup_{k>0} \mathrm{SPACE}(n^k)$
$\mathrm{NPSPACE} = \bigcup_{k>0} \mathrm{NSPACE}(n^k)$

The above classes are related in the following way:

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$$

For most of the inclusions, it remains open whether they are open or not. The central open in computational complexity is the relation between P and NP, because for most of the problems that arise naturally in the bibliography, those we can solve efficiently are in P and those we can't are in NP.

The above definitions hold for decision problems (languages). In this thesis we will discuss more about counting problems (functions), where we are interested in counting the solutions of a given problem. The core class for counting complexity is the following:

Definition 1.1.3 (Valiant [45])**.** *$\#\boldsymbol{P}$ is the class of all functions $f : \Sigma^* \to N$ such that for any $x \in \Sigma^*$, $f(x) = \mathrm{acc}_M(x)$, and M is a NPTM.*

Additionally the class of function computed by a deterministic polynomial-time Turing machine is FP. The open counting problem, which is analogue of the P vs. NP, is whether $\mathrm{FP} = \#\mathrm{P}$ or $\mathrm{FP} \subsetneq \#\mathrm{P}$, with the latter considered to be the case.

In order to provide completeness results later on we will use the following reductions:

Definition 1.1.4. *We say that a problem A reduces to a problem B by Karp reduction (polynomial-time many one) and we denote $A \leq_m^p B$, if there exists a polynomial-time computable function f such that for all x, $x \in A$ if and only if $f(x) \in B$.*

In case of functions f_A and f_B the above reduction is defined as follows:

$$f_A(x) \leq_m^p f_B : \exists g \in \mathrm{FP}, \forall x f_A(x) = g(f_B(x)$$

Definition 1.1.5. *We say that a problem A (or function) reduces to a problem B by Cook (polynomial-time Turing) reduction and we denote $A \leq_T^p B$, if A can be computed by a deterministic TM M within polynomial time with the use of an oracle for B. If M makes only one call to B, then we have Cook* [1] *reduction, denoted $A \leq_{[1]-T}^p B$.*

Note that the polynomial-time Turing reduction holds not only for languages(problems), but for functions also. For counting problems we will use also the parsimonious reduction:

Definition 1.1.6. *We say that a counting problem $\#A$ reduces to $\#B$ by parsimonious reduction if there exists a polynomial-time computable function f such that for all x, $|\{y \mid (x, y) \in A\}| = |\{z \mid (f(x), z) \in B\}|$.*

1.2 Ladner's Theorem and Implications

Computational complexity deals with classifying problems in terms of the recourses required to compute them. One of the main open problems is a proof or a disproof for $\mathrm{P} \neq \mathrm{NP}$. Since the latter is still open, the best one can hope is to classify every problem in NP to be either NP-complete or solvable in P. The above statement also holds for counting problems, that is in shortage of a proof for $\mathrm{FP} \neq \#\mathrm{P}$, can we classify every problem in #P to be either #P-complete or solvable in polynomial time?

A negative answer for both questions comes from the following theorem due to Ladner [39].

Theorem 1.2.1. *If* $\mathrm{P} \neq \mathrm{NP}$ *then there exists a language* $L \in \mathrm{NP}$ *that is neither in* P *nor* NP*-complete.*

The following proof can be found in [2] (with some details omitted):

Proof of theorem 1.2.1. Consider the language $\mathrm{SAT}_H = \{\phi 01^{n^{H(n)}} \mid \phi \in \mathrm{SAT} \wedge n = |\phi|\}$, where $H : \mathbb{N} \to \mathbb{N}$ is the function defined as follows:
$H(n)$ is the smallest number $i < \log\log n$ such that for every $x \in \{0,1\}^\star$ with $|x| \leq \log n$, M_i outputs $\mathrm{SAT}_H(x)$ within $|x|^i$ steps. If there is no such number i then $H(n) = \log\log n$.

In order to compute $H(n)$ we first need to compute $H(k)$ for every $k \leq \log n$ recursively. Then simulate at most $\log\log n$ Turing machines for every input of length at most $\log n$ for at most $(\log n)^{\log\log n}$ steps; this requires at most $o(n^2)$ computational steps. At every such step we need to check that indeed $M_i(x) = \mathrm{SAT}_H(x)$ on all inputs of length at most $\log n$. Finally the computation of $H(n)$ can be shown to require at most $O(n^3)$ steps.

Now assume that $\mathrm{SAT}_H \in \mathrm{P}$, and therefore there exists a Turing Machine M that solves SAT_H in at most n^c steps. That means that there exists a number $i > c$ such that $M_i = M$. By definition, for $n > 2^{2^i}$, $H(n) \leq i$ and therefore $H(n) = O(1)$.

On the other hand, if $H(n) = O(1)$, then H can take only finitely many values, therefore there exists an i such that $H(n) = i$, for finitely many n's. The latter implies that M_i solves SAT_H in n^i time. Thus, the last two paragraphs prove that $\mathrm{SAT}_h \in \mathrm{P} \iff H(n) = O(1)$. Moreover the latter argument holds even if we only assumed that there is some constant C such that $H(n) \leq C$ for infinitely many n's, hence proving that if $\mathrm{SAT}_H \notin \mathrm{P}$ then $H(n)$ tends to infinity with n.

Now let $\mathrm{SAT}_H \in \mathrm{P}$: From the above this would imply that $H(n) \leq C$ for some constant C, implying that SAT_H is simply SAT padded with at most a polynomial number of 1's. But then a polynomial-time algorithm for SAT_H can be used to solve SAT in polynomial time, implying that $\mathrm{P} = \mathrm{NP}$.

Suppose that SAT_H is NP-complete: Therefore there is a reduction r from SAT to SAT_H that runs in time $O(n^k)$ for some constant k. We have already shown that $\mathrm{SAT}_H \notin \mathrm{P}$, thus $H(n)$ tends to infinity. Since r works in $O(n^k)$ time, for large enough n it must map SAT instances of size n to SAT_H instances of size smaller than $n^H(n)$. Thus for large enough formulae ψ, the reduction r must map it to a string of the type $\phi 01^H(|\phi|)$ where ϕ is smaller by some fixed polynomial factor. More explicitly $|\phi|+1+|\phi|^{H(|\phi|)} = O(|\psi|^i)$, that

is $|\phi| = o(|\psi|)$. By recursively repeating the process until we get a small enough formulae, we have a polynomial-time algorithm for SAT, contradicting the assumption P $\neq$ NP.

□

Note that this proof technique can be applied to other classes as well, providing the following corollaries.

Corollary 1.2.2. *Provided* P $\neq$ NP*, there is an infinite hierarchy of separate complexity classes that lie between* P *and* NP*.*

Proof. We apply the proof of theorem 1.2.1 to show that there is an language that is neither in P nor inter-reducible with $\mathrm{SAT}_H =_{\mathrm{def}} \mathrm{SAT}_{H_0}$:

Consider the language $\mathrm{SAT}_{H_1} = \{\phi 01^{n^{H(n)}} \mid \phi \in \mathrm{SAT}_{H_0} \wedge n = |\phi|\}$, where H is the function defined in the proof of 1.2.1; SAT_{H_1} is not only in NP, but also reduces to SAT_{H_0} (since H is polynomially computable). On the other hand, following the same arguments as before, $\mathrm{SAT}_{H_1} \in$ P would imply that $\mathrm{SAT}_{H_0} \in$ P. Furthermore, a reduction $\mathrm{SAT}_{H_0} \leq_p \mathrm{SAT}H_1$, can provide us with a polynomial algorithm for SAT_{H_0}.

Now consider the following family of languages: $\mathrm{SAT}_{H_i} = \{\phi 01^{n^{H(n)}} \mid \phi \in \mathrm{SAT}_{H_{i-1}} \wedge n = |\phi|\}$. The complexity classes defined by the downwards Karp closure of SAT_{H_i}, is an infinite hierarchy of classes between P and NP. □

Corollary 1.2.3. *If* FP $\neq$ #P *then there is an infinite hierarchy of separate complexity classes between* FP *and* #P*.*

Proof. Since $\mathrm{SAT}_H \in$ NP, there exists a succinct[2] certificate, call it y, such that $x \in \mathrm{SAT}_H$, if there exists a polynomially computable relation R, such that $(x, y) \in R$. The counting version of SAT_H, namely $\#\mathrm{SAT}_H$, is the following: given x compute the number of such certificates y. If $\#\mathrm{SAT}_H$ were complete for #P, this would imply that #SAT reduces to $\#\mathrm{SAT}_H$. Similarly with the decision proof we would be able to construct a polynomial-time algorithm by reducing the size of the instance of #SAT recursively. On the other hand if we could compute $\#\mathrm{SAT}_H$ in polynomial time, then we would be able to decide SAT_H in polynomial time also. This argument extends for the counting versions of all the problems SAT_{H_i} defined previously. □

Except from the artificial problems described in the above proofs (SAT_H), there are some natural candidates that are not NP-complete, and they are conjectured to be neither in P nor NP-complete. Example of problems is the graph isomorphism and the decision version of factoring, namely given an integer n and an integer M with $1 \leq M \leq n$, does n have a factor d with $1 < d < M$? For the counting setting we can consider the problem of counting the number of divisors of a given positive integer.

On the other hand, the majority of decision(counting) problems that arise naturally are either tractable –efficiently computable –in P(FP), or hard –NP-complete(#P-complete). Since we cannot prove that the latter holds for every problem in NP(#P), we would be

[2]Of length at most polynomial of $|x|$

interested in finding proper subclasses, containing as many problems as possible, such that the problems of this subclass are either efficiently computable or hard. We would like for this class be restricted enough so that Ladner's theorem will not hold for it, but to contain as many problems as possible. This will allow us to completely define the complexity of the problems contained in such a class. Such theorems in computational complexity, where a set of problems is proven to contain either tractable or hard problems, and no problems of intermediate complexity, are known as dichotomy theorems (basically we partition the problems of the class in to two sets).

1.3 Three Examples of Computational Dichotomies

One of the first dichotomy theorems, after Shaefer's result which we will discuss in the next chapter, is due to Jaeger, Vertigan and Welsh [29], on the Tutte polynomial of a graph.

Definition 1.3.1. *The Tutte polynomial of a graph G is:*

$$T(G;x,y)=\sum_{A\subseteq E}(x-1)^{\kappa(V,A)-\kappa(V,E)}(y-1)^{|A|-(|V|-\kappa(V,A))}$$

where $\kappa(V,A)$ = the number of connected components of the graph (V,A).

The Tutte polynomial is defined for every undirected graph and encodes information about how the graph is connected. Some examples, that can be found in Welsh's Book [48] are the following:

- $T(G;1,1)$ counts the number of spanning trees of a connected graph G.
- $T(G;2,1)$ counts the number of forests in G
- $T(G;1,2)$ counts the number of edge subsets that are connected and span G.
- $T(G;2,0)$ counts the number of acyclic orientations of G.
- The chromatic polynomial $P(G;\lambda)$ of a graph G with n vertices, m edges and k connected components is given by: $P(G;\lambda)=(-1)^{n-k}\lambda^k T(G;1-\lambda,0)$.
  Where λ is a positive integer, $P(G;\lambda)$ counts the proper λ-colorings of G.

For fixed rationals x,y the dichotomy theorem of Jaeger, Vertigan and Welsh is on the following problem:

- Tutte(x,y):
  - Input: A graph G.
  - Output: $T(G;x,y)$.

More explicitly they have proven that the problem of evaluating the Tutte polynomial of a graph at a point in the (x, y)-plane is #P-hard except when $(x-1)(y-1) = 1$ or when (x, y) equals $(1, 1), (-1, -1), (0, -1), (-1, 0), (i, -i), (-i, i), (j, j^2), (j^2, j)$ where $j = e^{2\pi i/3}$. A visualization can be seen in figure 1.1, where the red points correspond to tractable cases, and the blue hard for general graphs but tractable for planar graphs.

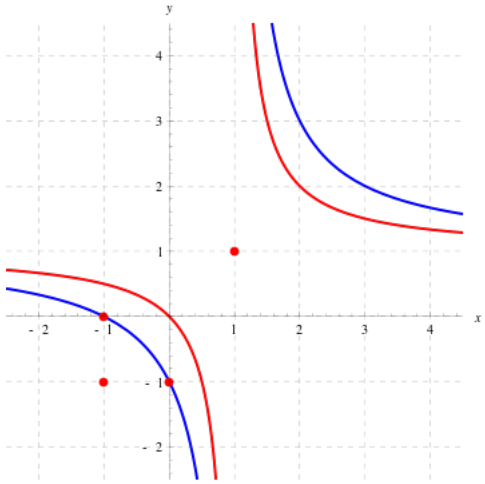

Figure 1.1: A visualization of the tractable cases of the Tutte polynomial.

The other dichotomy theorem that came out in 1990's was for the H-Coloring Problem. An H-coloring of G is just a homomorphism from G to H.

For example when $H = K_3$ then K_3-coloring is simply the problem 3-Colorability, and if $H = K_q$ then we have the q-Colorability problem.

Hell and Nesetril [28] proved that the H-coloring problem is in P if H is bipartite, otherwise it is NP-complete. (counting version also here?)

Many recent dichotomy theorems concern problems that come from statistical physics. In statistical physics they model spin systems on graphs, and they are interested in computing the Partition Function of a Spin System on a graph. One of the most intensively studied models in statistical physics is the Ising model, introduced in the 1920's by Lenz and Ising to study ferro-magnetism. An instance of the model is given by a set of n sites, a set of interaction energies V_{ij} for each unordered pair of sites i, j, a magnetic field intensity B, and an inverse temperature β. A configuration of the system defined by these parameters is one of 2^n possible assignments σ of ± 1 spins to each site. The energy of a configuration σ is given by the Hamilton $H(\sigma)$ defined by:

$$H(\sigma) = -\sum_{\{i,j\}} V_{ij}\sigma_i\sigma_j - B\sum_k \sigma_k$$

The interesting part of this sum is the first term, consisting of a contribution from pairs of sites. The importance of this expression comes from the Gibbs distribution, according to

which the probability that the system is in configuration σ is proportional $\exp(-\beta H(\sigma))$. This implies that the probability of configuration σ is $1/Z \times \exp(-\beta H(\sigma))$, where the normalizing factor Z, called the partition function of the system, is

$$Z = \sum_{\sigma \in \{-1,1\}^n} \exp(-\beta H(\sigma))$$

Note that computing the partition function of the Ising model equals to the problem $\textsc{Tutte}(x, y)$ with x, y such that $(x-1)(y-1) = 2$. Other notable statistical physics models are the Hard-Core Model (which is equal to computing the independent sets of a graph), the Potts Model and the Anti-ferromagnetic Potts Model (equal to counting colorings). The dichotomy theorems we have are in special cases of these models and state that the partition function can be approximated (FPRAS) under some values of a parameter of a system. For the Ising model the parameter is β.

What is most remarkable with these dichotomies is that they coincide with phase transitions proven by physicists. So when the parameter is under a critical value, the problem is tractable, and when the parameter surpasses this value, the problem becomes #P-hard, and furthermore the system changes phase.

Chapter 2

Decision Version of Constraint Satisfaction Problem

The purpose of this chapter is to introduce the constraint satisfaction problem (CSP) as a decision problem and present some of the main results about it.

Constraint Satisfaction problem first appeared in the bibliography in a 1974 article by Ugo Montanari [42]. The fore-mentioned article applied CSP to picture processing problems.

To give an intuition about the problem, consider CSP as a general framework modeling numerous known combinatorial problems. Examples of such problems can be found in graph theory (graph coloring, vertex cover, ...), in Logic (SAT), in database theory, in artificial intelligence, in operations research, in optimization and many other areas.

2.1 Definition of the Decision Problem

We begin with some elementary definitions.

Definition 2.1.1. *Domain is a finite set D of elements, usually denoted as $D = \{0, 1, \cdots, n-1\}$.*

Definition 2.1.2. *Cardinality of a domain D is denoted as $|D| = n$.*

Definition 2.1.3. *An k-ary relation R in domain D is a subset $R \subseteq D^k$.*

For example let $D = \{0, 1, 2\}$ be a domain, a binary relation over D is $L = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2)\}$.

Definition 2.1.4. *A k-ary constraint is a predicate applied to a vector of k variables.*

Examples of constraints are $C_1(x, y) = (x = y)$, $C_2(x, y) = (x \leq y)$ and $C_3(x, y, z) = (x \neq y \neq z \neq x)$.

From a relation R we can construct a constraint $R(\vec{x})$ such that R corresponds to the solution set of $R(\vec{x})$. For example let L be the fore-mentioned relation with domain

$D = \{0, 1, 2\}$, then the constraint $L(x, y)$ corresponds to the constraint $x \leq y$ with domain D.

Observe that from a relation R we may obtain more than one constraints. For example $R(x, y)$ and $R(y, x)$ are different constraints.

Let us now define the problem CSP. As input of CSP we are given a formula ϕ, which consists of a finite conjunction over constraints with D as domain. The output is "yes" when there exists an assignment of the variables used in ϕ , such that all the constraints are satisfied.

As an example consider the formula $(a = b) \wedge (b \neq d) \wedge (d < c) \wedge (b < c)$ with domain $D = \{0, 1, 2, 3\}$, then the assignment $a = 2, b = 2, c = 3, d = 2$ is not a solution as the second constraint is violated. On the other hand the assignment $a = 1, b = 1, c = 3, d = 2$ satisfies all the constraints, making the formula satisfiable.

Examples of specific problems we can express under the CSPframework are the following. The simplest example is the 3-Colorability of a graph. We are given an undirected graph $G = (V, E)$, and we want to decide whether there exists a coloring of it's vertices with three colors, such that no two adjacent vertices have the same color. Consider the domain $D = \{0, 1, 2\}$, so that each element of D corresponds to a each color. For every edge $v_i \in V$ we introduce the variable x_i of out CSP. In order to express the graph of figure 2.1 we will use as variable the set $V = \{x_a, x_b, x_c, x_d\}$.

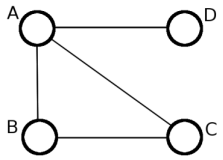

Figure 2.1: 3-Colorability as CSP

Finally for every edge $u, v \in E$, we have the constraint that corresponds to the relation NEQ $(x_u \neq x_v)$. In our current example the input of the CSP instance is $(x_a \neq x_b) \wedge (x_a \neq x_c) \wedge (x_b \neq x_c) \wedge (x_a \neq x_d)$. A solution can be seen in figure 2.2, and corresponds to the assignment $x_a = 0, x_b = 2, x_c = x_d = 1$, hence G is 3-colorable.

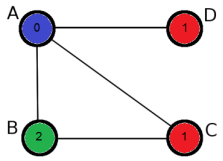

Figure 2.2: A 3-coloring for G

A second example of a problem that can be formulated as a CSP is the Vertex Covering problem. We are given as input a graph $G = (V, E)$ and an integer k and we want to decide

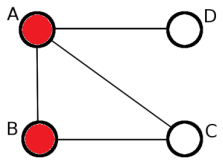

Figure 2.3: Vertex Cover as CSP

whether there is a set K of at most k vertices, that saturates all the edges of G, meaning each edge has at least one endpoint in K. The variables of the CSP will correspond to the vertices of G, that is for $v \in V$ we have the variable x_v; for the example in figure 2.3 the variable set will consist of $V = \{x_a, x_b, x_c, x_d\}$. The domain for vertex cover, formulated as a CSP is $D = \{0, 1\}$, where the evaluation of a variable $\sigma(x_v) = 1$ will correspond to the event $v \in K$. There will be two kinds of constraint: one constraint $\text{OR}(x_u, x_v)$ for every edge $(u, v) \in E$, where by OR we denote the relation $\text{OR}= \{(01), (10), (11)\}$, ensuring that every edge will be covered by at least one vertex; and another constraint K that will ensure that $|K| \leq k$, containing every variable in the problem. In the example depicted in figure 2.3, the CSP instance will be $\text{OR}(x_a, x_b) \wedge \text{OR}(x_a, x_c) \wedge \text{OR}(x_b, x_c) \wedge \text{OR}(x_a, x_d) \wedge K(x_a, x_b, x_c, x_d)$. A solution for $k = 2$, is given by $x_a = x_b = 1$, $x_c = x_d = 0$.

Besides general CSP we can restrict the type of constraints and obtain the so called parametric CSP. The parameter S is a set of relations over the domain D. The problem $\text{CSP}(S)$ now, has as input a formula φ, that is a conjunction of relations exclusively from S. Again the question of the problem is whether φ is satisfiable.

The general CSP it is known to be NP-complete: as shown in the examples both 3-Colorability and vertex cover, known NP-complete problems, are special cases of CSP. The interest around parametric CSP is that given a domain D, there exist some parameters S, for which $\text{CSP}(S) \in \text{P}$. For example $\text{CSP}(\{\text{OR}, K\}$ over $D = \{0, 1\}$ is NP-complete, because as shown in a previous example, it is the NP-complete problem vertex cover. Furthermore the problem $\text{CSP}(\{\text{NEQ}\})$ is the 3-Colorability problem when the domain is $\{0, 1, 2\}$, and hence NP-complete; but when as domain we consider the set $D = \{0, 1\}$, we have the 2-Colorability problem (check whether the given graph is bipartite), which can be solved in polynomial time.

2.2 Boolean CSP

For this section the domain of the relations used will be the boolean domain $D = \{0, 1\}$. Many natural problems are modeled as CSPs with relations over the boolean domain, hence it is an interesting case. Furthermore we have a complete classification of the complexity of parametric boolean CSP, due to Schaefer's theorem [44].

The classical problem 3Sat is an example of problem that can be modeled as CSP over the boolean domain. In order to formally define 3Sat we will need the following definition for boolean formulas.

Definition 2.2.1. *A formula is in conjunctive normal form (CNF), when it consists of a conjunction of clauses. The clauses in CNF are disjunction of literals, where every literal is either a variable x, or the negation of a variable $\bar{x}$.*

The input of 3SAT is a finite CNF-formula φ, such that each clause has at most three literals, and the output is "yes" when there exists a truth assignment of the variables that satisfies φ, or "no" otherwise. The problem 3SAT is known to be NP-complete due to Karp [35].

In order to express 3SATas a CSP we require the following relations: $R_0 = \{0,1\}^3 \setminus \{(0,0,0)\}$, $R_1 = \{0,1\}^3 \setminus \{(0,0,1)\}$, $R_2 = \{0,1\}^3 \setminus \{(0,1,1)\}$ and $R_3 = \{0,1\}^3 \setminus \{(1,1,1)\}$. For example we can obtain the following constraints $(x \vee y \vee z) = R_0(x,y,z), (x \vee y \vee \bar{z}) = R_1(x,y,z), (x \vee \bar{y} \vee z) = R_1(x,z,y), (\bar{x} \vee y \vee \bar{z}) = R_2(y,x,z)$, e.t.c.. Therefore 3SAT corresponds to the problem CSP$(\{R_0, R_1, R_2, R_3\})$.

Before presenting Shaefer's theorem we need to define the following properties over constraints.

Definition 2.2.2. *A clause will be called:*

- *Horn if it contains at most one positive literal,*
- *dual Horn if it contains at most one negative literal,*
- *bijunctive if it contains at most two literals*
- *affine if it can be described by an affine equation over GF[2].*

Furthermore a formula $\varphi = c_1 \wedge \cdots \wedge c_p$ is called Horn, dual Horn, bijunctive or affine if every clause c_i of φ is Horn, dual Horn, bijunctive or affine respectively.

For the previous definition (2.2.2), we always consider that the formulas are in CNF. Here are some examples for the above definitions:

- the formula $c_1 := (\bar{x} \vee \bar{y} \vee z)$ is Horn,
- the formula $c_2 := (\bar{x} \vee y \vee z)$ is dual Horn,
- the formula $c_3 := (\bar{x} \vee y)$ is bijunctive, Horn and dual Horn,
- the formula $c_4 := (x \vee y)$ is bijunctive and dual Horn,
- the formula $c_5 := (x + y + z = 1) \mod 2$ is affine,
- the formula $c_6 := (x = y)$ is affine, bijunctive, Horn and dual Horn,
- the formula $c_7 := (x \neq y)$ is affine and bijunctive.

While the general SAT problem is NP-complete, there are some special cases that can be solved in polynomial time (in P). Such a case is HornSat, where we are given a finite Horn formula φ, and we want to know whether it is satisfiable of not.

Similarly we can define the problem DualHornSat, with input now a dual horn formula. Note that a formula $\varphi(x_1, \cdots, x_k)$ is Horn if and only if the formula $\varphi(\bar{x_1}, \cdots, \bar{x_k})$ is dual Horn. Therefore we conclude that DualHornSat can be solved with a polynomial time algorithm.

Let us now restrict the input of the problem to contain only bijunctive formulas. We then obtain the problem 2SAT, a problem known to be a member of P.

Finally we will define the problem AffineSat. As an input we are given a finite affine formula φ and we want to determine whether φ is satisfiable or not. From another point of view AffineSat is the following problem: given a system of affine equations $A\vec{x} = b$, over the ring $\mathbb{Z}_2$ as input, determine whether it has a solution. By Gaussian elimination we can check whether the system $A\vec{x} = b$ has a solution within polynomial time, hence AffineSat $\in$ P.

Many natural problems can be solved as AffineSat. For example in 2-Colorability we are given a graph $G = (V, E)$, and we want to know if there exists a function $f : V \to \{0, 1\}$, such that it holds $f(x) \neq f(y)$ for each edge $(x, y) \in E$. Consider the relation $2col = \{(0, 1), (1, 0)\}$, that corresponds to the affine constraint $2col(x, y) = (x + y = 1) \mod 2$. Therefore for every edge $(x, y) \in E$ we have the constraint $(x + y = 1)$. All the constraint together define an affine system of equations $A\vec{x} = b$ in $\mathbb{Z}_2$, which we can solve in polynomial time with Gaussian elimination. This is a polynomial time solution of 2-Colorability.

To sum thing up the problem 3Sat is NP-complete, but its restrictions HornSat, DualHornSat, 2SAT and AffineSat are in P.

Another SAT variant we are going to use later on is the problem Nae3Sat. We are given a finite formula $\varphi = c_1 \wedge \cdots \wedge c_p$, and each clause contains exactly 3 literals. The difference from 3Sat now is that we want to know whether φ has a satisfying assignment such that in every clause c_i there is at least one literal taking the value "1" and at least one literal taking the value "0". Due to Shaefer [44], this problem is known to be NP-complete.

Lets see how we can express Nae3Sat as CSP. Consider a relation set S that contains the relation NAE $= \{001, 010, 011, 100, 101, 110\}$. By using NAE we can define the constraint $\text{NEQ}(x, y) = \text{NAE}(x, y, y)$, which is true if and only if x is assigned to a different value from y. With the use of NAE and OR we can now define the following constraints:

- $\text{NAE}_1(x, y, z) = \text{NAE}(x, y, \bar{z}) = \exists v(\text{NEQ}(z, v) \wedge \text{NAE}(x, y, v))$
- $\text{NAE}_2(x, y, z) = \text{NAE}(x, \bar{y}, \bar{z}) = \exists v(\text{NEQ}(y, v) \wedge \text{NAE}_1(x, v, z))$
- $\text{NAE}_3(x, y, z) = \text{NAE}(\bar{x}, \bar{y}, \bar{z}) = \exists v(\text{NEQ}(x, v) \wedge \text{NAE}_2(v, y, z))$

Note that similarly with 3Sat, the above constraints suffice to express Nae3Satas a CSP problem, i.e. Nae3Sat = CSP({NAE}), hence CSP({NAE}) is NP-complete.

Consider now the following trivial relations:

Definition 2.2.3. *Let $d \in \{0,1\}$. A relation R is called d-valid if it contains the vector $(d,d,\cdots,d)$. A set of relations S is called d-valid if every relation $R \in S$ is d-valid.*

Immediate from the definitions one can see that if S is a 0-valid or 1-valid set of relations then $\mathrm{CSP}(S)$ has an obvious solution (by assigning every variable to the value "0" or "1" respectively, and therefore $\mathrm{CSP}(S) \in \mathrm{P}$.

Now we are set to state the main theorem of this section, which provides a dichotomy for boolean unweighted(constraints are relations) CSP, due to Schaefer [44].

Theorem 2.2.4. *Let S be a set of relations over the boolean domain $\{0,1\}$. If S is*

- *0-valid,*
- *or 1-valid,*
- *or Horn,*
- *or dual Horn,*
- *or bijunctive,*
- *or affine,*

then $\mathrm{CSP}(S)$ *is in* P. *In every other case* $\mathrm{CSP}(S)$ *is* NP*-complete.*

Proof. We have already proven the cases for which $\mathrm{CSP}(S) \in \mathrm{P}$. We will show that in any other case, that is if S does not satisfy any of the six cases of theorem 2.2.4, S will contain the relation NAE we mentioned before. In that case, since $\mathrm{CSP}(\{\mathrm{NAE}\})$ is NP-complete, it also holds that $\mathrm{CSP}(S)$ is also NP-complete.

Note that the proof we are going to sketch here is not the original proof given by Schaefer (which is much more complicated). It is based in [1] and contains parts of universal algebra.

Before going to the main part of the proof we will need the following definitions.

Definition 2.2.5. *Define as Cartesian product of two relations $R_1 \subseteq D^k$ and $R_2 \subseteq D^m$, the relation $R_1 \times R_2 = \{z \in D^{k+m} \mid \exists x \in R_1, y \in R_2 : z_i = x_i$ for $i = 1,2,\cdots,k$ and $z_{k+j} = y_j$ for $j = 1,2,\cdots,m\}$.*

For example given the relations $R_1 = \{(0,1),(2,3)\}$ and $R_2 = \{(4,5,6),(7,8,9)\}$, then $R_1 \times R_2 = \{(0,1,4,5,6),(0,1,7,8,9),(2,3,4,5,6),(2,3,7,8,9)\}$

Definition 2.2.6. *We call projection of a relation $R \subseteq D^k$ on the t-th coordinate the relation $\mathrm{pr}_t R = \{z \in D \mid \exists x \in R : x_t = z\}$.*

In a similar manner we can define the projection of a relation in more than one coordinates. For example given $R = \{(1,2,3),(4,5,6),(7,8,9)\}$, the projection of R in the first and third coordinate is the relation $\mathrm{pr}_{1,3} R = \{(1,3),(4,6),(7,9)\}$ $(= \mathrm{pr}_1 R \times \mathrm{pr}_3 R)$

Definition 2.2.7. *We call identification of a relation $R \subseteq D^k$ in the coordinates s and t the relation $R' = \{z \in D^{k-1} \mid \exists x \in R : x_s = x_t$ and $z_i = x_i$ for $i = 1,2,\cdots,t-1,t+1,\cdots,k\}$.*

As an example let $R = \{(1,2,2),(3,4,5),(7,7,7)\}$. The identification of R at the last two coordinates is the relation $R' = \{(1,2),(7,7)\}$. In other words R' is the set of solutions of the constraint $R'(x,y)$ defined as $R'(x,y) = R(x,y,y)$.

Definition 2.2.8. *Let S be a set of relations. Define S to be closed in terms of the operation op if $\forall R \in S, op(R) \in S$. Informally, by applying the operation op, we cannot get a relation outside of S.*

Definition 2.2.9. *Let S be a set of relations. We can construct a new set, namely S', such that it contains all the relations of S, all the projections of the relations of S, all the projections of the projections of the relations of S, e.t.c.. The set S' is called the projection closure of S.*

As an example consider the set $S = \{R\}$, where $R = \{(1,2,3),(4,5,6),(7,8,9)\}$, then we can construct the relations:

- $R_1 := \mathrm{pr}_{2,3}R = \{(2,3),(5,6),(8,9)\}$,
- $R_2 := \mathrm{pr}_{1,3}R = \{(1,3),(4,6),(7,9)\}$,
- $R_3 := \mathrm{pr}_{1,2}R = \{(1,2),(4,5),(7,8)\}$,
- $R_4 := \mathrm{pr}_{2}R_1 = \{(3),(6),(9)\}$,
- $R_5 := \mathrm{pr}_{1}R_1 = \{(2),(5),(8)\}$,
- $R_6 := \mathrm{pr}_{1}R_2 = \{(1),(4),(7)\}$,

The projection closure of S is the set $S' = S \cup \bigcup_{1 \le i \le 6} \{R_i\}$.

Let EQ denote the equality relation, that is $\mathrm{EQ} = \{(d,d) \mid d \in D\}$.

Definition 2.2.10. *The* relational clone *(co-clone) of S, denoted with $\langle S \rangle_R$, is the "smallest" (infinite) set of relations such that:*

- $S \subseteq \langle S \rangle_R$,
- $EQ \in \langle S \rangle_R$,
- $\langle S \rangle_R$ *is closed under Cartesian product,*
- $\langle S \rangle_R$ *is closed under projection,*
- $\langle S \rangle_R$ *is closed under identification.*

The reason we define relational clones is the following theorem:

Theorem 2.2.11. *For any set S of relations, we have $\#\mathrm{CSP}(S) \equiv_{\mathrm{AP}} \#\mathrm{CSP}(\langle S \rangle_R)$.*

In other words relational clones can be considered to be normal forms of sets of relations. We can have S_1 and S_2 two different sets of relations, but $\langle S_1 \rangle = \langle S_2 \rangle$. Due to theorem 2.2.11 we don't have to deal with the sets S_1, S_2 separately, when studying the complexity of their CSPs.

Corollary 2.2.12. *Let S_1 and S_2 be two different sets of relations. If $\langle S_1 \rangle = \langle S_2 \rangle$ then CSP(S_1)$\equiv_P$ CSP(S_2).*

Definition 2.2.13. *A* lattice *is a partially ordered set in which any two elements have a unique supremum (also called a least upper bound) and a unique infimum (also called a greatest lower bound).*

Let S be the set of all relational clones of all the different relations that have domain $D = \{0, 1\}$, then the structure of the partial relation $(S, \subseteq)$,is a lattice, called Post's lattice. Furthermore the structure of Post's lattice is completely known and depicted in figure 2.4: each circle corresponds to a relational clone in the Boolean domain, when two relational clones are connected with an edge, that is interpreted that the clone placed lower in the lattice is a subset if the clone placed higher.

The next theorem [32, 31, 30] is crucial to the proof of Shaefer's theorem.

Theorem 2.2.14. *If we have $S_1 \subseteq \langle S_2 \rangle$ (with S_1 finite) then the reduction CSP(S_1)$\leq_P$ CSP(S_2) holds.*

Immediate corollaries from theorem 2.2.14

Corollary 2.2.15. *If a relational clone S implies CSP(S)$\in P$, then for all co-clones S' below S, we have CSP(S')$\in P$.*

Corollary 2.2.16. *If a co-clone S implies CSP(S)$\in$* NP*-complete, then for all co-clones S' above S, we have CSP(S')$\in$* NP*-complete.*

Let us mention again Shaefer's theorem, in combination with Post's lattice. The relational clones that correspond to the six polynomially decidable cases of CSP(S) as stated in the theorem are the following:

- 0-valid: IL_0,
- 1-valid: IL_1,
- Horn: IE_2
- dual Horn: IV_2,
- bijunctive: D_2,
- affine: IL_2.

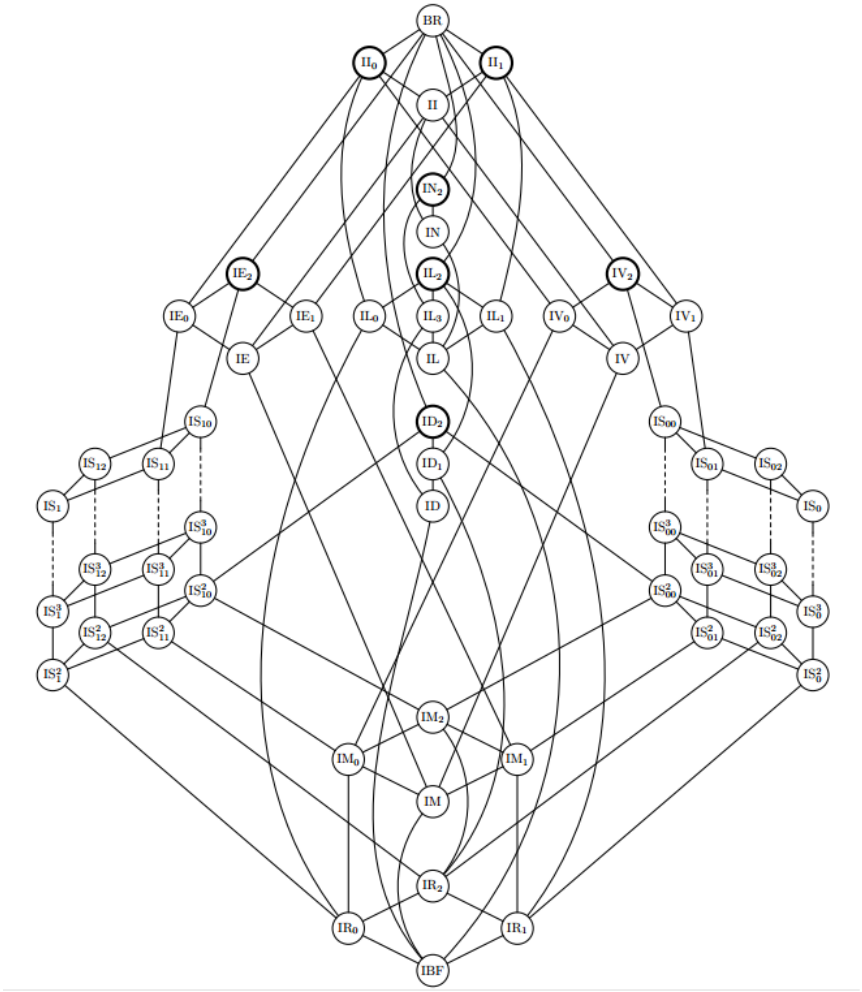

Figure 2.4: Post's Lattice

For relations in the above relational clones we have already proven that the corresponding CSP's are in P. Due to corollary 2.2.15 it holds that for all relational clones bellow the fore-mentioned clones in the lattice, their corresponding CSP's will also be polynomially decidable.

This leaves us with only two relations which have CSP's of unknown complexity:

- The total relational clone BR, contains all possible boolean relations.
- The relational clone IN_2, which is the clone generated by the relation NAE.

We already know that CSP({NAE} is NP-complete, hence CSP(BR) and CSP(IN_2) are also NP-complete. This concludes the proof. Note that in a later chapter we will discuss clones more explicitly. □

Shaefer's theorem can be specialized even more. Namely in [1] we can obtain completeness results in subclasses of P, i.e. $\oplus$L, NL, L, for the polynomially decidable CSP's.

2.3 The General Case

The general case (not only boolean domain) has been studied by Feder and Vardi in [22]. They consider subclasses of NP for which Ladner's theorem doesn't hold. The widest such class is called MMSNP (Monotone Monadic Strict NP without inequality), named so due to a logical characterization. It is defined as a special case of Fagin's theorem for NP, having three restriction on the expressibility of the formula. Note that ignoring any of these three restrictions, we cannot obtain a dichotomy theorem, as Ladner's theorem applies. Additionally it is shown that MMSNP contains exactly the problems that can be formulated as CSP's. It is conjectured that the general parametric CSP is either in P or NP-complete (Feder-Vardi Conjecture).

An extension to Schaefer's result came from Bulatov [3], where we have a dichotomy theorem for the three element CSP, that is the domain of the constraints is $D = \{0, 1, 2\}$. The Feder-Vardi conjecture stated previously remains open to date.

Chapter 3

Dichotomies on Counting Problems

This section contains three frameworks that model counting problems, namely Graph Homomorphisms, counting constraint satisfaction problem #CSP, and holant problems. We will prove that the #CSP is a generalization of graph homomorphisms, and that holant is a generalization of #CSP. Additionally we will survey the dichotomy theorems that have been proved for the three fore-mentioned counting problem frameworks.

3.1 Counting Frameworks

In this section we will introduce three frameworks for counting problems. The goal is to define proper subclasses of problems of #P so we can classify them completely in terms of their complexity.

3.1.1 Graph Homomorphisms

The first an the simplest of the frameworks we are going to present in this thesis, is the graph homomorphism problem.

Definition 3.1.1. *Let $\boldsymbol{A} = (A_{i,j}) \in \mathbb{C}^{q\times q}$ be a complex matrix. The* graph homomorphism *problem EVAL(*$\mathbf{A}$*) is the following:*
***Input**: A graph $G = (V, E)$.*
***Output**: The partition function:*

$$Z_{\mathbf{A}}(G) = \sum_{\sigma:V\to[q]} \prod_{(u,v)\in E} A_{\sigma(u),\sigma(v)}.$$

Depending on the choice of matrix $\mathbf{A}$, the partition function can express different properties, and the problem EVAL may correspond to a different natural problem. Some examples follow:

- Let $\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$, then $Z_{\mathbf{A}}(G)$ counts the number of vertex covers in G.

- Let $\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, then $Z_{\mathbf{A}}(G)$ counts the number of 3-colorings in G.
- And in general if $\mathbf{A} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}$, then $Z_{\mathbf{A}}(G)$ counts the number of k-colorings in G, where $\mathbf{A}$ is a $k \times k$ matrix.
- Counting the number of induced subgraphs of G with an even number of edges: $\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

3.1.2 Counting Constraint Satisfaction Problems

The second framework of problems we are going to examine here is #CSP. It is defined as follows:

Definition 3.1.2. *Let $[q]$ be a domain set. A constraint language $\mathcal{F}$ is a finite set of functions $\{f_1, \cdots, f_h\}$ in which $f_i : [q]^{r_i} \to \mathbb{C}$ is an r_i-ary function for some $r_i \geq 1$. The problem $\#\mathrm{CSP}(\mathcal{F})$ is:*
***Input**: Let $x = (x_1, \cdots, x_n)$ be a set of n variables. The input is a collection I of m tuples $(f, i_1, \cdots, i_r)$ in which f is an r-ary function in $\mathcal{F}$ and $i_1, \cdots, i_r \in [n]$.*
***Output**: The partition function:*

$$Z_{\mathcal{F}}(I) = \sum_{x \in [q]^n} \prod_{(f, i_1, \cdots, i_r) \in I} f(x_{i_1}, \cdots, x_{i_r}).$$

If the functions in $\mathcal{F}$ have as range the set $\{0, 1\}$, then we can see them as relations and we have the unweighted $\#\mathrm{CSP}(\mathcal{F})$.

It is easy to see that #CSP, is the counting version of the decision CSP, where we want to decide whether $Z_{\mathcal{F}}(I) > 0$ as discussed in chapter 2. An expected example of a problem that can be modeled as #CSPis #3Sat. More explicitly let $D = \{0, 1\}$ and $\mathcal{L} = \{f_0, f_1, f_2, f_3\}$, where:

$$\begin{aligned} f_0(x, y, z) &= x \vee y \vee z, \\ f_1(x, y, z) &= \bar{x} \vee y \vee z, \\ f_2(x, y, z) &= \bar{x} \vee \bar{y} \vee z, \\ f_3(x, y, z) &= \bar{x} \vee \bar{y} \vee \bar{z}, \end{aligned}$$

hence $\#\mathrm{CSP}(\mathcal{L})$ is precisely the problem #3Sat.

As we have already mentioned in the introduction #CSP is a generalization of the graph homomorphisms problem. This is the case because an instance of graph homomorphisms,

with matrix $\mathbf{A} = (A_{i,j})$, can be expressed as #CSP($\mathcal{L}$), where $\mathcal{L} = \{f_A\}$ contains only one function f_A. Let the function $f_A(i,j) = A_{i,j}$, and correspond to every vertex $v_1, v_2 \dots v_n$ a variable $x_1, x_2 \dots x_n$. Apply a constraint $f_A(x_i, x_j)$ if and only if $(v_i, v_j) \in E(G)$. Hence graph homomorphisms is indeed a special case of #CSP.

3.1.3 Holant Problems

The third, and the most general model we are going to discuss in this thesis, is the Holant problem defined in [14].:

Definition 3.1.3. *Let $[q]$ be a domain set and F be a finite set of complex-valued functions over $[q]$. The* Holant(F) *problem is:*
***Input**: A signature grid $\Omega = (G, F, \pi)$, where $G = (V, E)$ is a labeled graph and π labels each vertex $v \in V$ with a function $f_v \in F$ so that the arity of f_v is the same as the degree of v.*
***Output**:* Holant$_\Omega$*, where*

$$\text{Holant}_\Omega = \sum_{\sigma: E \to [q]} \prod_{v \in V} f_v(\sigma \mid_{E(v)}).$$

As an example consider the problem of counting perfect matchings of a graph. Freedman Lovász and Schrijver have proven that this problem cannot be expressed as graphs homomorphisms [24]. Counting the perfect matchings of a graph G can be modeled as a Holant problem in the following way. Let $[q] = \{0, 1\}$ and F contain all the ExactOne functions, that is the functions that return 1, if exactly one of their arguments (input variables) is set to 1, and return 0 in any other case. In every vertex v of G we correspond the function $f_v \in$ ExactOne that has the proper arity. The product $\prod_{v \in V} f_v(\sigma \mid_{E(v)})$ equals with 1, if $\sigma^{-1}(1) \subseteq E$ is a perfect matching, otherwise $\prod_{v \in V} f_v(\sigma \mid_{E(v)}) = 0$. Therefore Holant$_\Omega$ counts the number of perfect matchings of G. If instead functions in ExacOne, we consider AtMostOne functions, then we count the number of (not necessarily perfect) matchings.

Definition 3.1.4. *Let A be a set of functions. We can define a sub-framework* Holant$^{\mathcal{A}}$ *of Holant as*

$$\text{Holant}^{\mathcal{A}}(\mathcal{F}) = \text{Holant}(\mathcal{F} \cup \mathcal{A}).$$

In this sub-framework, we call the functions in $\mathcal{A}$, freely available functions.

If all equality functions are assumed to be freely available, then the sub-framework which is created is exactly the #CSP problem, i.e.

$$\#\text{CSP}(\mathcal{F}) = \text{Holant}(\mathcal{F} \cup \mathit{Equalities}).$$

In order to be convinced for the above, consider the following:

- Represent an instance of #CSP by a bipartite graph where the Left Hand Side (LHS) is labeled by variables and the Right Hand Side (RHS) is labeled by constraints (functions).
- The signature grid Ω is as follows: Every variable node on LHS is attached an EQUALITY function and every constraint node on RHS has the given constraint function.
- The EQUALITY function on each variable node forces the incident edges to take the same value; this effectively reduces edge assignments to vertex assignments assigning values to each variable on LHS as in #CSP.

Here are some special cases of the Holant problem:

Definition 3.1.5. *Let U denote the set of all unary functions. Then,*

$$\text{Holant}^{\star}(\mathcal{F}) = \text{Holant}(\mathcal{F} \cup U).$$

Definition 3.1.6. *Let Δ_i be the unary function which gives value 1 on inputs $i \in [q]$, and 0 on all other inputs. Then,*

$$\text{Holant}^{c}(F) = \text{Holant}(\mathcal{F} \cup \{\Delta_1, \cdots, \Delta_q\}).$$

Clearly, Holant^c is a super framework of Holant*. Furthermore, Holant^c can be also viewed as a super framework of #CSP [17].

The main reason we define subclasses of the Holant problem, is that in shortage of a general dichotomy theorem for Holant, we look for dichotomies for the special cases of the Holant problem. The Venn diagram of the classes of problems we defined in this section is depicted in figure 3.1.

3.2 Dichotomy Theorems

In this section we will mention the dichotomy theorems that exist for the classes of problems we defined in the previous section (3.1).

3.2.1 Graph Homomorphisms

Let us first discuss the case of Graph Homomorphisms for an undirected graph, that is the case where the matrix $\mathbf{A}$ is symmetrical. If additionally, the matrix $\mathbf{A}$ contains only 0,1 valued entries, Dyer and Greenhil in [20] have proved the following:

Theorem 3.2.1. *The counting Graph Homomorphisms problem of an undirected graph $H_{\mathbf{A}}$ is in* #P*-complete if $H_{\mathbf{A}}$ has a connected component which is not a complete graph with all loops present or a complete bipartite graph with no loops present. Otherwise, the counting problem is in* FP*.*

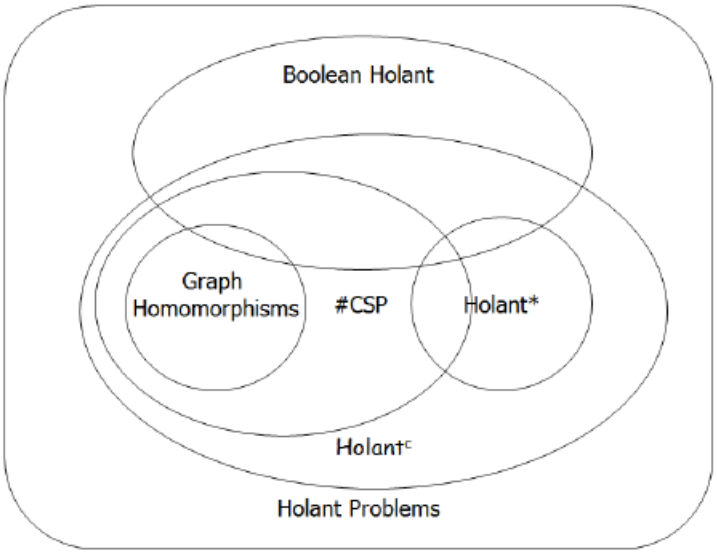

Figure 3.1: The classes of counting problems

Later on Bulatov and Grohe in [7] extended the result to the non-negatively weighted setting. Goldberg, Grohe, Jerrum, and Thurley in [26] have proven the corresponding dichotomy theorem for all symmetric matrices with real number values.

Finally Cai, Chen and Lu in [11] provided the following general theorem:

Theorem 3.2.2. *Let* $\mathbf{A}$ *be a symmetric complex matrix. Then counting graph Homomorphisms to* $H_{\mathbf{A}}$ *either can be computed in polynomial time or is* #P*-hard.*

For the unsymmetric case (i.e. for non-directed graphs), Dyer, Goldberg and Paterson in [19] proved a dichotomy for an acyclic family of graphs. Cai and Chen in [9] solved the problem for all acyclic graphs with non-negative weights.

As discussed in section 3.1.2, Graph Homomorphism is a special case of #CSP, therefore the remaining cases are included in the #CSP dichotomy theorems, where we have a complete dichotomy theorem.

3.2.2 Counting Constraint Satisfaction Problems

We begin by stating the results for the boolean #CSP, with constraints that have domain $D = \{0, 1\}$. For the unweighted case, were the constraint functions can be considered relations we have the following theorem due to Creignou and Hermann.

Theorem 3.2.3. *Let* $\mathcal{F}$ *be a set of relations over a Boolean domain* $\{0, 1\}$*. The problem* $\#\mathrm{CSP}(\mathcal{F})$ *is in* FP *if every relation in* $\mathcal{F}$ *is affine. Otherwise,* $\#\mathrm{CSP}(\mathcal{F})$ *is* #P*-complete.*

Let us now generalize the affine relations[1] to pure affine functions by scaling them by a factor c. Denote with $\mathcal{P}$ the the class of functions expressible as a product of unary functions, binary equality functions and binary disequality functions. The following theorem is due to Dyer, Goldberg and Jerrum [17].

Theorem 3.2.4. *Let $\mathcal{F}$ be a set of non-negative functions over Boolean domain. Then* #CSP($\mathcal{F}$) *is* #P*-hard unless all the functions in $\mathcal{F}$ are pure affine or of the product type $\mathcal{P}$, in which case the problem is in* P*.*

In 2009, Cai, Lu, Xia [14] and independently Bulatov, Dyer, Goldberg, Jalsenius and Richerby [5] finally proved the dichotomy for complex weighted Boolean #CSP.

For the general case, where the domain of the functions D can be any finite set, we have Bulatov's groundbreaking result [5] for the unweighted case (relations as constraints). The proof of his dichotomy criterion was based in universal algebra, making it hard to understand, and furthermore it wasn't shown to be decidable. Dyer and Richerby in [21] gave a simpler dichotomy criterion, which was proven to be equal with Bulatov's. They also proved that the criterion is decidable.

For the weighted case we the result of Bulatov, Dyer, Goldberg, Jalsenius, Jerrum and Richerby [4], provides a dichotomy for the CSP with positive rational weighted functions as constraints. Cai, Chen and Lu in [12] extended this result for all non-negative weights. The general case was proved by Cai and Chen in [10], were the constraints of $\mathcal{F}$ can be any complex valued function. Whether their criterion, consisting of three conditions, is decidable remains open.

Finally Cai, Lu and Xia in [63] prove a trichotomy for #CSP, depending on the input graph[2] of the problem. The problems can be separated in three classes:

- Tractable for every input graph.
- Hard for every input graph.
- Hard for general input graphs, but tractable for planar input graphs. The planar cases of such CSP's are precisely the problems solved by holographic algorithms[3].

Figure 3.2 is a visualization of the above facts.

3.2.3 Holant Problems

In this section we will mention the dichotomy results for the problems Holant* and Holant^c.

Cai, Lu, and Xia, in [14] have proven a dichotomy for the symmetric, boolean Holant* with complex weights. Later on they extended this result so that it includes the non-symmetric case [15].

[1] Consider them as functions with boolean range.

[2] The input graph of a CSP is a bipartite graph. One partition contains vertices corresponding to variables, the other contains vertices corresponding to constraint appearances in the input. Connect a variable vertex with a constraint vertex when the variable appears in the constraint.

[3] For an introduction to holographic algorithms the reader is referred to the appendix A

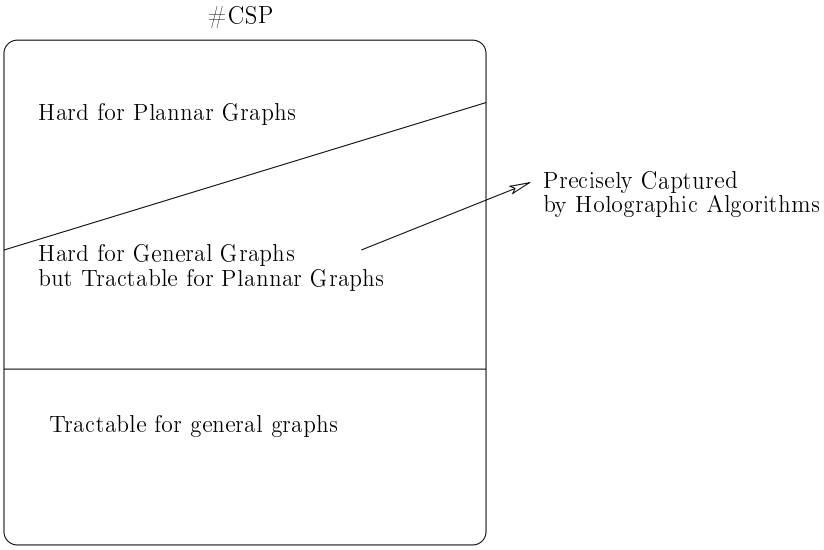

Figure 3.2: A trichotomy for #CSP

In the same paper [14] provided a dichotomy theorem for the symmetric boolean Holant^c with real number weights. The complex number weighted case was resolved by Cai, Huang and Lu in [13].

Most of the cases, for the general Holant class remain open.

Chapter 4

Approximating #CSP

In this final chapter, we relax the requirement of exact computations, and study the problems in terms of approximate computations. We define the properties required by an approximation, and define a reduction that preserves the approximability of the problems. Boolean Unweighted #CSP, is the simplest case we consider, and for which we have a solid "trichotomy" theorem, partitioning the problems in three classes. Finally we present the first results for the weighted case. Many questions remain open.

4.1 FPRAS and AP-Reductions

The following section contains the definitions of the approximation algorithms for counting problems we are going to use, and the reductions we will use in order to obtain complexity theoretic results in terms of approximations.

4.1.1 Approximation Algorithms for Counting Problems

For counting problems we are going to use the following notion of approximation:

Definition 4.1.1 (Randomized Approximation Scheme)**.** *A randomized approximation scheme for f is a randomized algorithm $\mathcal{A}$, such that given an input x and an error tolerance $0 < \varepsilon$, and outputs an integer z such that for every instance x*

$$Pr[(1-\varepsilon)f(x) \leq z \leq (1+\varepsilon)f(x)] \geq \frac{3}{4}.$$

Furthermore a RAS is said to be a fully polynomial randomized approximation scheme *(FPRAS), if it runs in time polynomial in $|x|$ and ε^{-1}.*

In terms of obtaining an FPRAS, it is sometimes useful, and equivalent to substitute the approximation interval from $[(1-\varepsilon)f(x), (1+\varepsilon)f(x)]$, with $[e^{-\varepsilon}f(x), e^{\varepsilon}f(x)]$. Additionally it was shown in [34], that the $3/4$ on the left hand side of the previous equation can be any number in $(1/2, 1)$.

The first question that comes in mind, is whether relaxing the requirement of an algorithm to give the exact value of a function, allows us to solve problems in polynomial time, that in the exact case would be #P-complete. The answer is positive, and here are examples of problems #P-complete problems that have an FPRAS:

- #DnfSat:
  Given a boolean formula in disjunctive normal form, count the number of satisfying assignments. This problem can be shown to be #P-complete under Turing Reductions. Let φ be an instance of #SAT, and $\text{acc}(\varphi)$ the number of its satisfying assignments. The formula $\neg\varphi$ can be converted to DNF-form in polynomial time, and has $2^n - \text{acc}(\varphi)$ satisfying assignments (assuming that n is the number of variables of φ). Therefore provided an oracle for #DnfSat we can solve #SAT in polynomial time. The FPRAS of #DnfSat was given by Karp and Luby in [36].

- Permanent:
  Given a $n \times n$ non-negative matrix A compute the permanent of a matrix, $\text{per}(A) = \sum_\sigma \prod_i a(i, \sigma(i))$, were the summation is over all permutations σ of $[n]$. This particular problems has been well studied, and has various applications to other sciences. It was show to be #P-complete by Valiant, in the paper [45] that pretty much initiated the area of counting complexity. An pretty innovative FPRAS was given in [33].

4.1.2 AP-Reductions and Three Complexity Classes

As we are interested in complexity results for approximating computations, we have to define an appropriate reduction. Furthermore we would like the class of problems admitting FPRAS to be closed under this reduction, while the reduction is as general as possible. The following reduction will be used:

Definition 4.1.2 (Approximation Preserving Reduction [16])**.** *Suppose f, g functions from $\Sigma^\star$ to $\mathbb{N}$. An* approximation-preserving reduction *from f to g ($f \leq_{\text{AP}} g$) is a randomized algorithm $\mathcal{A}$ for computing f using an oracle for g. Takes as input a pair $(x, \varepsilon) \in \Sigma^\star \times (0, 1)$ and satisfies the following:*

1. *Every oracle call is of the form (w, δ), with w an instance of g and $\delta \in (0, 1)$ satisfying $\delta^{-1} \leq \text{poly}(|x|, \varepsilon^{-1})$.*

2. *The algorithm is a RAS, whenever the oracle meets the specs of a RAS.*

3. *The run time of $\mathcal{A}$ is polynomial in $|x|$ and ε^{-1}*

Note that since the parsimonious reduction conserves the number of solutions, it can be considered as a special case of an AP-reduction. Furthermore since #SAT is complete for #P under parsimonious reduction, it is also complete for the problems of #P under the AP-reduction. For an NP-complete decision problem A, the counting problem $\#A$, is complete for #P with respect to AP-reducibility.

Lets assume that #SAT has an FPRAS. By the definition of FPRAS, we then could distinguish with high probability whether the given formula φ has zero solutions or not, in time polynomial. Furthermore as shown in [18] this error probability can be made one-sided, proving, under the assumption that #SAT has an FPRAS, that NP = RP, which is considered to be unlikely. In conclusion every #P-complete problem under the AP-reduction, cannot have an FPRAS unless NP = RP.

But not all #P-complete problems (under AP-reduction) have decision version that is NP-complete. An example of a problem with easy decision version, but hard even to approximate is the problem #IS: Count the number of independent sets (of all sizes) of a given graph G. The decision is trivial since every graph contains an independent set of size 0. As a problem of #P it is trivial that #IS $\leq_{\text{AP}}$ #SAT. It is remarkable that the inverse also holds, combining the last two statements we have that #IS $\equiv_{\text{AP}}$ #SAT, hence #IS cannot have an FPRAS, unless NP = RP.

How hard is it to approximate #IS if we restrict the input graph G to be bipartite? This is known as the problem #BIS. An FPRAS for #BIS is not known, and there is strong evidence that this problem is inapproximable. It can be shown to belong in a class named[1] #RHΠ_1, and furthermore that, all problems that are AP-interreducible with #BIS are in this class. Later on this thesis we are going to use another problem that belongs in #RHΠ_1, and therefore is AP-interreducible with #BIS. The problem #Downsets: Count the number of downsets of a given partially ordered set $(X, \preceq)$. Where a downset in $(X, \preceq)$, is a subset $D \subseteq X$ that is closed under $\preceq$; i.e., $x \preceq y$ and $y \in D$ implies $x \in D$.

To sum thing up, the results of [16] as discussed in this section are the following:

There are three degrees of approximability within problems of #P:

- Solvable by an FPRAS:
  Permanent, #DnfSat...
- AP-interreducible with #SAT:
  #SAT, #IS, #Mon2Sat...
- An Intermediate Class #RHΠ_1(AP-Interreducible with #BIS):
  #BIS, #Downsets...

4.2 Unweighted Boolean #CSP

In the parent section we will present a "Trichotomy theorem for the boolean unweighted #CSP, under AP-reductions. This is basically the analogue of Shaefer's dichotomy theorem, were the #CSPcounting problems are shown to either have an FPRAS, or be complete for the class #RHΠ_1 or for #P under approximation preserving reductions.

[1] This class has a logical characterization, in a way similar to Fagin's theorem for NP. The "RH" name comes from Reduced Horn from.

4.2.1 Relational Clones Revisited

The proof of the main theorem of the section is based once again in relational clones. In more detail we are going to prove that an instance of #CSP($\mathcal{F}$), where $\mathcal{F}$ contains relations that belong to the relational clone IM_2, which we will define later, captures precisely the complexity of #RHΠ_1.

Definition 4.2.1 (Primitive positive formula)**.** *Let $\mathcal{R}$ be a set of Boolean relations. A* pp-formula *over $\mathcal{R}$ is an existentially quantified product of atomic formulae:*

$$\psi = \exists x_{n+1} \dots x_{n+m} \bigwedge_{j=1}^{s} \varphi_j$$

A *basis* for the relational clone I is a set $\mathcal{R}$ of Boolean relations such that the relations in I are exactly the relations that can be implemented with a pp-formula over $\mathcal{R} \cup \{\mathrm{EQ}\}$. Therefore $I = \langle\mathcal{R}\rangle_R$, and note that every relational clone has such a basis. For $R \in \langle\mathcal{R}\rangle_R$, it holds that $\langle\mathcal{R} \cup \{R\}\rangle_R = \langle\mathcal{R}\rangle_R$. A basis $\mathcal{R}$ is called *plain basis* for $\langle\mathcal{R}\rangle_R$, if every member of $\langle\mathcal{R}\rangle_R$ is definable by a CNF($\mathcal{R}$)-formula.

The main idea is that implementations from pp-formulas for the constraints correspond to reductions for the counting problems defined by these constrains. Therefore in order to study the complexity for cases of #CSP, depending on which relational clone the constraints of the instance belong to, we only have to study the complexity of the instances containing relations from the basis of the clone.

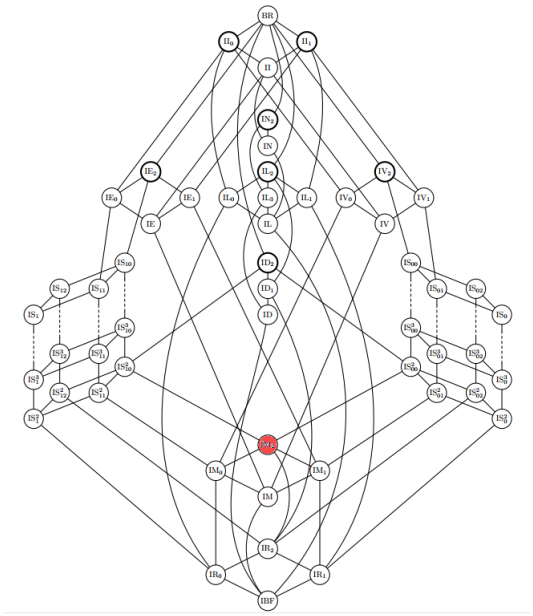

Figure 4.1: Post's lattice and the relational clone IM_2

The relational clone we are going to use is IM_2. Its position in Post's lattice can be seen in figure 4.1. The basis that defines IM_2 is $\mathcal{R} = \{\delta_0, \delta_1, \mathrm{IMP}\}$. Moreover it is proven to be a plain basis for IM_2. This means that every relation in IM_2 is logically equivalent with a conjunction of predicates of the following three forms:

1. $\delta_0(x_i) = \{(0)\} = \neg x_i$
2. $\delta_1(x_i) = \{(1)\} = x_i$
3. $\mathrm{IMP}(x_i, x_j) = \{(0,0), (0,1), (1,1)\} = x_i \to x_j$

4.2.2 The Trichotomy Theorem

For the unweighted boolean #CSP, in terms of approximation preserving reduction, we have a correspondence with the three classes of problems stated in section 4.1.2. Namely in this section we are going to present the proof of the following theorem as given in [18].

Theorem 4.2.2. *Let $\mathcal{F}$ be a constraint language with domain $\{0,1\}$. Then one of the following hold:*

1. *If every relation if $\mathcal{F}$ is affine then $\#\mathrm{CSP}(\mathcal{F}) \in \mathrm{FP}$.*
   (Admitting FPRAS.)
2. *If every relation in $\mathcal{F}$ is in IM_2 then $\#\mathrm{CSP}(\mathcal{F}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$.*
   (#RHΠ_1-complete)
3. *Otherwise $\#\mathrm{CSP}(\mathcal{F}) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$.*
   (Inapproximable under assumptions)

Proof. The lemmata used in this proof are going to be stated here unproven. The reader is referred to original paper for a complete proof.

1. If every relation if $\mathcal{F}$ is affine then $\#\mathrm{CSP}(\mathcal{F}) \in \mathrm{FP}$: This comes directly from the Craignou-Herman #CSP dichotomy as stated in theorem 3.2.3.

2. If every relation in $\mathcal{F}$ is in IM_2 then $\#\mathrm{CSP}(\mathcal{F}) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$: We begin with the case that if$\mathcal{F} \subseteq \mathrm{IM}_2$ then $\#\mathrm{CSP}(\mathcal{F}) \leq_{\mathrm{AP}} \#\textsc{Downsets} \equiv_{\mathrm{AP}} \#\mathrm{BIS}$. Let $\mathcal{F} \subseteq \mathrm{IM}_2$, hence every constraint an instance I of $\#\mathrm{CSP}(\mathcal{F})$ is a conjunction of the form $\delta_0(x_i)$ or $\delta_1(x_i)$ or $\mathrm{IMP}(x_i, x_j)$. But since a CSP instance is a conjunction of constrains, we can restate the previous phrase so that $\#\mathrm{CSP}(\mathcal{F})$ has constraints of the form $\delta_0(x_i)$ or $\delta_(x_i)$ or $\mathrm{IMP}(x_i, x_j)$. Define $\mathrm{IMP}^\star(x_i, x_j)$ the transitive closure of IMP, that is $\mathrm{IMP}(x_i, x_j) \wedge \mathrm{IMP}(x_j, x_k) \implies \mathrm{IMP}^\star(x_i, x_k)$. Now let $N_0(I)$ be the set of variables forced to take the value 0. A variable x_i can be in $N_0(I)$, if $\delta_0(x_i) \in I$ or $\mathrm{IMP}^\star(x_i, x_j)$ and $\delta_0(x_j)$ are in I. Similarly define $N_1(I)$ to be the set of variables forced take the value 1. I.e. if $x_i \in N_1(I)$ then either $\delta_1(x_i) \in I$ or $\mathrm{IMP}(x_j, x_i)$ and $\delta_1(x_j)$ are in I. Removing all the variables of the instance I that are in $N_0(I)$ and in $N_1(I)$,

and the constraints that are applied to them, doesn't affect the number of solutions. Variables forced to take one value can only be satisfied in one way, therefore the only nonzero addends of the partition function are the ones for witch these variables take their forced value. Furthermore we can identify all pairs of variables s.t. $\text{IMP}^\star(x_i, x_j)$ and $\text{IMP}^\star(x_j, x_i)$. Again this action has no effect on the number of solutions, as x_i and x_j will create nonzero addend of the partition function only if they take the same value. The remaining variables and relations define a partial order, namely $(X, \preceq)$. Let σ be an evaluation of the variables. The evaluation of a variable, such that $\sigma(x_i) = 1$, corresponds to x_i belonging in the downset. Hence it cannot hold $\text{IMP}(x_i, x_j)$ with $\sigma(x_j) = 1$ and $\sigma(x_i) = 0$, the satisfying assignments are in one to one correspondence with the downsets of $(X, \preceq)$. Note in fact that we have proven the stronger $\#\text{CSP}(\mathcal{F}) \leq_m^p \#\textsc{Downsets}$.

Now consider the case where $\mathcal{F} \subseteq \text{IM}_2$ and not Affine. We will show that $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\mathcal{F})$. We will make use of the following two lemmata:

Lemma 4.2.3. *For a constraint language $\mathcal{F}$ with domain $\{0,1\}$, either $\#\text{CSP}(\mathcal{F} \cup \{\delta_0\}) \leq_{\text{AP}} \#\text{CSP}(\mathcal{F})$ or $\#\text{CSP}(\mathcal{F} \cup \{\delta_1\}) \leq_{\text{AP}} \#\text{CSP}(\mathcal{F})$.*

Note that lemma 4.2.3 is the only part of this proof that takes advantage of the requirement relaxation from exact computations to approximate with high probability.

Lemma 4.2.4. *If R is a non affine relation over $\{0,1\}$ then $\{R, \delta_0\}$ implements either OR= $\{(0,1),(1,0),(1,1)\}$, IMP, NAND= $\{(0,0),(0,1),(1,0)\}$. (same holds for $\{R, \delta_1\}$)*

Observe that if $\mathcal{F}$ implements R, then $\#\text{CSP}(\mathcal{F} \cup \{R\}) \leq_m^p \#\text{CSP}(\mathcal{F})$. This also holds for the AP-reduction. By combining the previous lemmata, we have one of the following cases:

(a) $\#\text{CSP}(\mathcal{F} \cup \{\delta_0, \text{IMP}\}) \leq_{\text{AP}} \#\text{CSP}(\mathcal{F} \cup \{\delta_0\})$

(b) $\#\text{CSP}(\mathcal{F} \cup \{\delta_0, \text{NAND}\}) \leq_{\text{AP}} \#\text{CSP}(\mathcal{F} \cup \{\delta_0\})$

(c) $\#\text{CSP}(\mathcal{F} \cup \{\delta_0, \text{OR}\}) \leq_{\text{AP}} \#\text{CSP}(\mathcal{F} \cup \{\delta_0\})$

The same hold for δ_1 in place of δ_0, but since the proof is symmetrical, will will only show it for the three cases above. In order to complete the proof we now have to prove that the following hold: $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\{\text{IMP}\})$, $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\{\text{NAND}\})$ $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\{\text{OR}\})$.

- $\#\text{BIS} \leq_{\text{AP}} \#\text{CSP}(\{\text{IMP}\})$: Let $G = (V_1, V_2, E)$ be an instance of #BIS. Consider the the reduction that corresponds to each vertex $v \in G$ the variable v, and to each edge $\{v_1, v_2\} \in E$, with $v_1 \in V_1$, the constraint $\text{IMP}(v_1, v_2)$. The independent sets of G are in one to one correspondence with the satisfying assignments of $\#\text{CSP}F$. A vertex $v_1 \in V_1$ is in the independent set of G if and only if $\sigma(v_1) = 1$. Respectively a vertex $v_2 \in V_2$ is in the Independent Set if and only if $\sigma(v_2) = 0$.

- #BIS $\leq_{\text{AP}}$ #SAT $\equiv_{\text{AP}}$ #IS $\leq_{\text{AP}}$ #CSP($\{$NAND$\}$): Let $G = (V, E)$ instance of #IS. Consider the the reduction that corresponds to each vertex $v \in G$ the variable v, and to each edge $\{v_1, v_2\} \in E$ the constraint NAND(v_1, v_2). The independent sets of G are in one to one correspondence with the satisfying assignments of #CSP($\mathcal{F}$). A vertex $v \in V$ is in the independent set of G if and only if $\sigma(v) = 1$.
- #BIS $\leq_{\text{AP}}$ #SAT $\equiv_{\text{AP}}$ #IS $\leq_{\text{AP}}$ #CSP($\{$OR$\}$): The reduction is the same with the one above; corresponds to each vertex $v \in G$ the variable v, and to each edge $\{v_1, v_2\} \in E$ the constraint NAND(v_1, v_2). The independent sets of G are in one to one correspondence with the satisfying assignments of #CSP($\mathcal{F}$). In this case a vertex $v \in V$ is in the independent set of G if and only if $\sigma(v) = 0$.

3. #CSP($\mathcal{F}$) $\equiv_{\text{AP}}$ #SAT. The fact that #CSP($\mathcal{F}$) $\leq_{\text{AP}}$ #SAT comes from the fact that #SAT is complete for #P with respect to the approximation preserving reduction. The part #SAT $\leq_{\text{AP}}$ #CSP($\mathcal{F}$), when $\mathcal{F}$ is neither affine, nor contains relations only from IM_2 is an immediate consequence of the following lemma:

   Lemma 4.2.5. *Let R_1 and R_2 be relations on $\{0, 1\}$ (not necessarily separate). If R_1 is not affine and R_2 is not in* IM_2 *then* #SAT $\leq_{\text{AP}}$ #CSP($\{R_1, R_2\}$).

□

4.3 Weighted Boolean #CSP

Now we consider the Weighted version. That is the constraints are functions with boolean domain, and their range is in positive real numbers. We could consider functions with negative numbers, but this will introduce cancellations, which will increase the difficulty of our analysis. There is a dichotomy theorem [49, 50, 51] for complex weighted functions, that prove dichotomies for the problems #CSP^c and #CSP*. These problems are the analogues of Holant^c and Holant* respectively. That means that all unweighted for (the c-case) or complex-weighted (for the $\star$ case) unary constraints are freely available. Furthermore, there is the restriction that the variables appear at most a bounded number of times among all given constraints.

In this section we are going to present more general results than the ones above, as they appear in [6, 41].

4.3.1 Functional Clones

In section 4.2.1 we defined the pp-formulas over relations. As we pointed out before, pp-implementations between constraints correspond to reductions among the CSP problems. As we intend to study weighted #CSP, we will generalize our definitions in order to explore functional clones.

Definition 4.3.1 (pps-Formula)**.** *Let $\mathcal{F}$ be a set of boolean functions.A* primitive positive summation formula *over $\mathcal{F}$ is a summation of a product of atomic formulae of $\mathcal{F}$:*

$$\psi = \sum_{x_{n+1}\ldots x_{n+m}} \prod_{j=1}^{s} \phi_j.$$

And specifies the function $F_\psi : D^n \to R$:

$$F_\psi(\mathbf{x}) = \sum_{\mathbf{y}\in D^m} \prod_{j=1}^{s} F_{\phi_j}(\mathbf{x}, \mathbf{y}).$$

Like in the relational setting we can define the functional clone $\langle\mathcal{F}\rangle$ to be the set of Boolean functions containing exactly the relations that can be implemented with a pps-formula over $\mathcal{R} \cup \{\text{EQ}\}$, where EQ is the function specified by the equality relation EQ$= \{(0,0),(1,1)\}$. Every functional clone has a basis. Note also that for $F \in \langle\mathcal{F}\rangle$, it holds that $\langle\mathcal{F} \cup \{F\}\rangle = \langle\mathcal{F}\rangle$.

If it were the case that we were interested in exact computations we would stop here, but we want to consider approximate computations, therefore we have to consider constraint functions that correspond to AP-reductions.

Definition 4.3.2 (pps$_\omega$ Definable Functions)**.** *We say that an α-ary function F with domain D is pps$_\omega$-definable over $\mathcal{F}$, if there exists a finite subset $S_F \subseteq \mathcal{F}$, such that, for every $\varepsilon > 0$, there exists an α-ary function $\hat{F}$, pps-definable over S_F , such that:*

$$\|F - \hat{F}\|_\infty = \max_{\mathbf{x}\in D^\alpha} |F - \hat{F}| < \varepsilon$$

Again we can define the pps$_\omega$ functional clone $\langle\mathcal{F}\rangle_\omega$ to be the set of Boolean functions containing exactly the relations that can be implemented with a pps-formula over $\mathcal{R}\cup\{\text{EQ}\}$. Every pps$_\omega$ functional clone has a basis. Note also that for $F \in \langle\mathcal{F}\rangle_\omega$, it holds that $\langle\mathcal{F} \cup \{F\}\rangle_\omega = \langle\mathcal{F}\rangle_\omega$.

Many approximation-preserving reductions in the literature are based not on a fixed "gadget" but on sequences of increasingly-large gadgets that come arbitrarily close to some property without actually attaining it. The notion of pps$_\omega$-definability provided here is intended to capture this phenomenon.

Will will use a computationally effective version of $\langle\mathcal{F}\rangle_\omega$, in order to obtain the main lemma of this section.

Definition 4.3.3 (Efficiently pps$_\omega$ Definable Functions)**.** *We say that a function F is efficiently pps$_\omega$-definable over $\mathcal{F}$ if there is a finite subset S_F of $\mathcal{F}$, and a TM M_{F,S_F} with the following property: on input $\varepsilon > 0$, M_{F,S_F} computes a pps-formula ψ over S_F such that F_ψ has the same arity as F and $\|F_\psi - F\|_\infty < \varepsilon$. The running time of M_{F,S_F} is at most a polynomial in* $\log \varepsilon^{-1}$.

Likewise the *efficiently pps$_\omega$-definable functional clone* $\langle\mathcal{F}\rangle_{\omega,p}$, is the set of all functions that can be represented efficiently by a pps$_\omega$-formula over $\mathcal{F}\cup\{\mathrm{EQ}\}$. And the corresponding closure property still holds, that is, $G \in \langle\mathcal{F}\rangle_{\omega,p} \implies \langle\mathcal{F}\cup\{G\}\rangle_{\omega,p} = \langle\mathcal{F}\rangle_{\omega,p}$.

Since we want to derive computational results, we now restrict attention to functions whose co-domains are restricted to efficiently-computable real numbers. A real number is polynomial-time computable if the first n bits of its binary expansion can be computed in time polynomial in n. For $n \in \mathbb{N}$, we will denote with $\mathcal{B}^p_n$ the set of boolean with arity n and domain the polynomial-time computable real numbers, and with $\mathcal{B}^p$ the boolean functions of all arities with domain the polynomial-time computable real numbers.

Finally we are ready to state the main theorem of this section:

Theorem 4.3.4. *Let $\mathcal{F}$ be a finite set of $\mathcal{B}^p$. If $F \in \langle\mathcal{F}\rangle_{\omega,p}$, then* $\#\mathrm{CSP}(F,\mathcal{F}) \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{F})$.

That is the notion of efficient pps$_\omega$-definable clone captures the complexity of approximation preserving reductions in terms of #CSP's. An immediate corollary is the following:

Corollary 4.3.5. *Let S_1 and S_2 be two different sets of relations, such that, $\langle S_1\rangle_{\omega,p} = \langle S_2\rangle_{\omega,p}$, then* $\#\mathrm{CSP}(S_1) \equiv_{\mathrm{AP}} \#\mathrm{CSP}(S_2)$.

4.3.2 Log-supermodular Functions

In contrast with the relational clones, where we have a complete picture of Post's lattice, we know little about the functional clones, and especially, since we are interested in AP-reductions, the efficient pps$_\omega$ functional clones. In order to present the first results regarding pps$_\omega$ functional clones we need to define Log-supermodular functions:

Definition 4.3.6. *A function F is* log-supermodular *(lsm) if $F(\mathbf{x}\vee\mathbf{y})F(\mathbf{x}\wedge\mathbf{y}) \geq F(\mathbf{x})F(\mathbf{y})$ and by **LSM**, we denote the class of all lsm functions.*

The term log-supermodular comes from the fact that F is lsm if and only if $\ln F$ is supermodular. It isn't hard to observe that the natural logarithm of all boolean functions of arity 1 is modular, so it is also supermodular, hence all boolean functions of arity 1 are lsm. Another example of lsm function, which is also a binary function is $\mathrm{IMP}(x,y)$.

The reason to study lsm functions is because of their closure property that pps$_\omega$-definable clones from lsm functions, contain only lsm functions. Formally stated:

Lemma 4.3.7. *If $\mathcal{F} \subseteq$ LSM is any set of lsm functions then $\langle\mathcal{F}\rangle_\omega \subseteq$ LSM.*

If we only consider binary functions, that is of arity two (members of $\mathcal{B}^p_2$), we can get a clear picture of the pps$_\omega$-functional clones. Every binary Boolean function can belong in one of the following sets:

- $\langle\mathcal{B}^p_1\rangle_{\omega,p}$, functions that can be efficiently pps$_\omega$-defined over unary functions.
- $\langle\mathrm{NEQ},\mathcal{B}^p_1\rangle_{\omega,p}$ functions that can be efficiently pps$_\omega$-defined over unary functions and the function specified by the relation NEQ$= \{(0,1),(1,0)\}$.

- $\langle \text{IMP}, \mathcal{B}_1^p \rangle_{\omega,p}$ functions that can be efficiently pps_ω-defined over unary functions and the function IMP.

- $\langle \text{OR}, \mathcal{B}_1^p \rangle_{\omega,p}$ functions that can be efficiently pps_ω-defined over unary functions and the function OR.

Furthermore the clone $\langle \text{OR}, \mathcal{B}_1^p \rangle_{\omega,p}$ can efficiently pps_ω-define all the binary functions (total clone for boolean binary functions). So when we restrict our functions to binary we have a quite complete picture of the clones lattice. Namely:

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

As the interest is focused in obtaining general results, we want to study the functional clones of all functions of $\mathcal{B}^p$. The main theorem of [6] states that if a function is not generated[2] by the NEQ-clone ($\langle \text{NEQ}, \mathcal{B}_1^p \rangle_{\omega,p}$), then it generates the function IMP. But the most important part of this theorem is the fact that there are no functional clones lying strictly between the clone of lsm functions and the total clone, or in other words, if a clone can generate a non-lsm function, then it can generate every function. Formally we have:

Theorem 4.3.8. *For $F \in \mathcal{B}^p$ the following hold:*

- *If $F \notin \langle \text{NEQ}, \mathcal{B}_1^p \rangle$ then $\text{IMP} \in \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$, and hence $\langle \text{IMP}, \mathcal{B}_1^p \rangle_{\omega,p} \subseteq \langle F, \mathcal{B}_1^p \rangle_{\omega,p}$*

- *If, in addition, $F \notin \text{LSM}$, then $\langle F, \mathcal{B}_1^p \rangle_{\omega,p} = \mathcal{B}^p$*

This also holds for the non-effective version of the theorem, that is the range of the functions can be the non-negative reals (functions in $\mathcal{B}$ instead of $\mathcal{B}^p$), and the pps_ω-definability can be noneffective.

One wonders whether the IMP clone suffices to generate all the lsm functions. The results of [41] state that that is not the case:

Theorem 4.3.9. *Let $\text{LSM}_k = \text{LSM} \cap \mathcal{B}_k$. The following hold:*

- $\langle \text{LSM}_2 \rangle = \langle \text{LSM}_3 \rangle$

- $\langle \text{LSM}_2 \rangle \subsetneq \langle \text{LSM}_4 \rangle$

With a vague picture on the functional clones lattice, we are ready to proceed on the complexity results on boolean #CSP, with constraint functions in $\mathcal{B}^p$, in terms of AP-reductions:

Theorem 4.3.10. *[Bulatov, Dyer, Goldberg, Jerrum [6]] Suppose $\mathcal{F}$ is a finite subset of $\mathcal{B}^p$*

[2] By generate we mean efficiently pps_ω-define.

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

As it can be deduced from the above theorem, what we know of the complexity of #CSP, with constraint functions in $\mathcal{B}^p$, is that we can approximate $\#\mathrm{CSP}F$, if $\mathcal{F}$ is in the clone $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1^p \rangle$, otherwise the problem is at least as hard to approximate as any problem in $\#\mathrm{RH}\Pi_1$. Provided that $\mathcal{F}$ contains some non-lsm functions then the corresponding constraint satisfaction problem is inapproximable under complexity theoretic assumptions.

There are many problems that remain open in therms of the approximation of counting problems: Does the conjecture that #BIS and in general the problems of $\#\mathrm{RH}\Pi_1$, are indeed inapproximable, and under which complexity theoretic assumptions? Which functional clone corresponds to the computational complexity of $\#\mathrm{RH}\Pi_1$, in other words can we make theorem 4.3.10 into "1-1" correspondence with the approximate counting classes? Furthermore as we stated previously the functional clones lattice picture is still incomplete. Can the study of the functional clones be helpful in other areas, can we deduce from it other interesting complexity results for #CSP's?

The ultimate goal for #CSP's, is to have a complete dichotomy theorem for the general case like in [10], but in terms of AP-reductions. It is also interesting to study the complexity of Holant problems under both reductions (Turing and AP), again a series of results from the more restricted cases to the general ones is expected.

Appendix A

Holographic Algorithms

This appendix is a brief introduction to a recent algorithmic technique named holographic algorithms. Holographic Algorithms were introduced by Valiant in [46], and manage to solve "exotic" problems, not previously known to be in FP, in polynomial time. High level speaking, the general idea is to reduce (within polynomial time) a counting problem to the the problem of counting the perfect matchings of a planar graph. The latter problem is known to have a polynomial algorithm (FKT-algorithm) due to Kastelyn and Temperley and Fisher independently [23, 37, 38].

Let us now define formally the problem of counting perfect matchings:

Definition A.0.11. *The problem* #PerfMatch *is the following:*
Input: An undirected graph $G = (V, E, W)$ *with weights on its edges.*
Output:

$$\text{PerfMatch}(G) = \sum_{E'} \prod_{(i,j)\in E'} w_{i,j},$$

where the summation is over all perfect matchings E' *of* G*.*

Note that if for every edge $(v_i, v_j) \in E$, we set $w_{i,j} = 1$, PerfMatch(G) is exactly the number of the perfect matchings of G. The corresponding decision problem is known to be in P, but on the other hand the counting problem #PerfMatch is #P-complete [45].

When G is a planar graph the we can solve it using the FKT-algorithm:

Theorem A.0.12. *There is a polynomial time computable function* f *that given a planar embedding of a planar graph* $G = (V, E, W)$ *defines* $f : E \to \{-1, 1\}$ *such that for the antisymmetric matrix* M *defined so that for all* $i < j$

- *if* $(i, j) \notin E$ *then* $M_{i,j} = M_{j,i} = 0$*, and*
- *if* $(i, j) \in E$ *then* $M_{i,j} = f(i, j) \cdot w_{i,j}$ *and* $M_{j,i} = -f(i, j) \cdot w_{i,j}$*,*

it is the case that $\text{PerfMatch}(G) = \sqrt{Det(M)}$*.*

In order to solve a counting problems with the help of a holographic algorithm we are going to use the following strategy, named holographic reduction:

- The individual components of an instance I of a counting problem (e.g. nodes and edges) will be replaced by gadgets that we call *matchgates*.
- Therefore, we transform the instance I to an instance Ω of what we call a *matchgrid*.
- The weighted sum of the perfect matchings of Ω will equal the number of solutions of I.

Therefore solving #PerfMatch with the FKT-algorithm we have an algorithm for the initial counting problem.

Let us now provide the definitions for the above:

Definition A.0.13. *A* planar matchgate Γ *is a triple* (G, X, Y) *where* G *is a planar embedding of a planar graph* (V, E, W)*, where* $X \subseteq V$ *is a set of* input *nodes,* $Y \subseteq V$ *is a set of* output *nodes, and* $X \cap Y = \emptyset$*.*

- *The* arity *of the matchgate is* $|X| + |Y|$*.*
- *The* standard signature of Γ with respect to *a set* $Z \subseteq X \cup Y$ *is* $\text{PerfMatch}(G - Z)$*.*
- *The* standard signature *of* Γ *is the* $2^{|X|} \times 2^{|Y|}$ *matrix* $u(\Gamma)$ *whose elements are the standard signatures of* Γ *with respect to* Z *for all possible choices of* Z*.*

Crucial to the holographic reductions is finding a proper basis:

Definition A.0.14. *A* basis *(*of size 1*) is a set of two distinct nonzero vectors, which we call* n *and* p*.*

Furthermore the basis $\mathbf{b0} = [n, p] = [(0, 1), (1, 0)]$ is called the *standard basis*. Note that in general, the vectors in a basis do not need to be independent.

As an example we are going to use the basis $\mathbf{b1} = [n, p] = [(-1, 1), (1, 0)]$

If we have two vectors q, r of length l, m, respectively, then we shall denote the *tensor product* $s = q \otimes r$ to be the vector s of length $l \cdot m$ in which $s_{i \cdot m + j} = q_i \cdot r_j$. Thus, for example, for the basis $\mathbf{b1}$, $n \otimes p = (-1, 0, 1, 0)$.

We are going to use two kinds of matchgates as defined bellow:

Definition A.0.15. *We say that a matchgate is a* generator *if it has zero input nodes and nonzero output nodes.*
Similarly, we say that a matchgate is a recognizer *if it has zero output nodes and nonzero input nodes.*

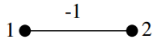

Figure A.1: A generator matchgate

A generator matchgate Γ for basis **b1** with output nodes $\{1, 2\}$ and one edge of weight -1 can be found in figure A.1.

The standard signature $u(\Gamma)$ of the matchgate in figure A.1 is the vector (-1,0,0,1). Therefore, it generates $n \otimes n + n \otimes p + p \otimes n$. The signature of this generator with respect to the basis **b1** will then be (1,1,1,0). For $x \in \{n, p\}^2$ we shall denote by $valG(\Gamma, x)$ the signature element corresponding to x. Thus, for the current example, $valG(\Gamma, n \otimes p) = 1$ and $valG(\Gamma, p \otimes p) = 0$.

A recognizer matchgate for basis **b1** with input nodes $v_1, v_2, \cdots, v_5$ and edge weights $w_1, w_2, \cdots, w_5$ can be seen in figure A.2.

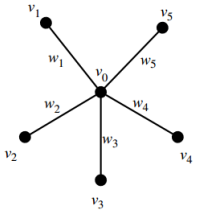

Figure A.2: A recognizer matchgate

Let us consider a recognizer like the above but with k inputs. The purpose of such recognizers is to have PerfMatch take on appropriate values as the inputs range over 2^k possible tensor product values $x = x_1 \otimes x_2 \otimes \cdots \otimes x_k$, where $x_i \in \{n, p\}$. If vector u is the standard signature of Γ, then $valR(\Gamma, x)$ is the inner product of u and x.

The following proposition holds for generator matchgates:

Proposition A.0.16. *For all $k > 0$ and for all $w1, \cdots, w_k$ there exists a $k-$input recognizer matchgate Γ such that on input $x = x_1 \otimes x_2 \otimes \cdots \otimes x_k \in \{n, p\}^k$ over basis **b1**, $valR(\Gamma, x)$ equals:*

- $-(w_1 + \cdots + w_k)$ *if* $x_1 = \cdots = x_k = n$,
- w_i *if* $x_i = p$, *and* $x_j = n$ *for every* $j \neq i$,
- *0 otherwise.*

The above proposition holds for the recognizer of figure A.2, where the number of input vertices is $k = 5$.

We define a *matchgrid* over a basis **b** to be a weighted undirected planar graph G that consists of:

- a set B of g generator matchgates $B_1, \cdots, B_g$,
- a set A of r recognizer matchgates $A_1, \cdots, A_r$, and
- a set C of f connecting edges $C_1, \cdots, C_f$ where each C_i edge has weight one and joins an output node in a generator with an input node of a recognizer.

Consider such a matchgrid $\Omega = (A, B, C)$ and denote by $X = \mathbf{b}^f = (n, p)^f$ the set of 2^f possible combinations of the basis elements n, p that can be transmitted simultaneously along the f connecting edges in the matchgrid.

The *value of the matchgrid at* x is the quantity

$$Holant(\Omega) = \sum_{x \in b^f} [\prod_{1 \le j \le g} valG(B_j, x_j)] \cdot [\prod_{1 \le i \le r} valG(A_i, x)].$$

In the core of holographic reductions, and holographic algorithms, lies the following theorem:

Theorem A.0.17 (Valiant '04 [46])**.** *For any matchgrid* Ω *over any basis* $\boldsymbol{b}$ *if* Ω *has weighted graph* G *then*

$$Holant(\Omega) = PerfMatch(G).$$

Let us now give some examples of these "exotic" problems that now have a polynomial time holographic algorithm.

We begin with the problem #X-Matchings:
Input: A Planar Wighted Bipartite Graph $G = \{V = (V_1, V_2), E, W\}$, where the nodes in V_1 have degree 2.
Output: $\sum_M \text{mass}(M)$, where M is a (not necessarily perfect) matching and

$$\text{mass}(M) = \prod_{(i,j) \in M} w_{i,j} \prod_{i \text{ unsat}} [-\sum_{j \sim i} w_{i,j}]$$

The holographic algorithm for #X-Matchings, constructs a matchgrid H over **b1** by replacing: $v \in V_1$ with the generator matchgate of the example and $u \in V_2$ with the recognizer matchgate of the example of proper degree. For each edge (u, v) connect an output of the generator matchgate of u to an input of the recognizer matchgate for v. It is obvious that $Holant(\Omega_H) = \#\text{X-Matchings}(G)$

Some other problems that have been solved by holographic algorithms are:

- #Pl-3-Nae-Sat: Planar not-all-equal 3 satisfiability (CNF).
- $\#_7$Pl-Rtw-Mon-3-Cnf:
  Input: A planar 3CNF Boolean formula where each variable appears positively and in exactly two clauses (Planar, Read-twice, monotone, 3CNF).
  Output: Number of sat. assignments.

The holographic algorithm for the latter problem was given in [47]. The surprising fact about it is that solving it mod 2 is $\oplus$P-complete!

To sum things up, in order to design a Holographic Algorithm for a combinatorial problem, one has to do the following:

1. Find suitable basis vectors.

2. Find the suitable matchgates.

A series of results [52, 53, 56, 59, 54, 55, 57, 58, 60, 61, 25] explore the realizable signatures, that is the functions for which we can find a matchgate with the proper values under the relevant basis.

A Holographic Algorithm uses planar perfect matchings on a planar matchgrid in order to compute the output. This usually restricts the problem to planar combinatorial instances. Cai, Lu, Xia in [62] extended the Holographic algorithms to non-planar structures by introducing a new gadget in order to realize signatures called Fibonacci gates. Some new counting problems that can be solved in polynomial time are the following:

- Given a 3-regular graph, compute the number of even colorings minus the number of odd colorings. (2-coloring on the edges, even when even edges are colored black).
- Given a Rtw-Cnf formula, compute the number of even lying assignments minus the number of odd lying assignments. (A variable is lying when it is inconsistent).

We conclude by explaining the fact that Valiant's function Holant_{Ω} is the same expression with the counting problem framework $\text{Holant}_{\Omega'}$, where $\Omega' = (G, F, \pi)$, that we used in section 3.1.3: every vertex v of $\text{Holant}_{\Omega'}$ problem can be replaced by a matchgate, with value equal to f_v. The degree of v equals to the number of output nodes of the matchgate. Assignments to edges of G correspond to vectors of the basis used by the matchgrid.

Bibliography

[1] Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *Electronic Colloquium on Computational Complexity (ECCC)*, (100), 2004.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[3] Andrei A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *FOCS*, pages 649–658, 2002.

[4] Andrei A. Bulatov, Martin E. Dyer, Leslie Ann Goldberg, Markus Jalsenius, Mark Jerrum, and David Richerby. The complexity of weighted and unweighted #CSP. *CoRR*, abs/1005.2678, 2010.

[5] Andrei A. Bulatov, Martin E. Dyer, Leslie Ann Goldberg, Markus Jalsenius, and David Richerby. The complexity of weighted boolean #CSP with mixed signs. *Theor. Comput. Sci.*, 410(38-40):3949–3961, 2009.

[6] Andrei A. Bulatov, Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. Log-supermodular functions, functional clones and counting csps. In *STACS*, pages 302–313, 2012.

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

[8] Jin-Yi Cai. Holographic algorithms: guest column. *SIGACT News*, 39(2):51–81, 2008.

[9] Jin-Yi Cai and Xi Chen. A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In *FOCS*, pages 437–446, 2010.

[10] Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. In *STOC*, pages 909–920, 2012.

[11] Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph homomorphisms with complex values: A dichotomy theorem. In *ICALP (1)*, pages 275–286, 2010.

[12] Jin-Yi Cai, Xi Chen, and Pinyan Lu. Non-negatively weighted #CSP: An effective complexity dichotomy. In *IEEE Conference on Computational Complexity*, pages 45–54, 2011.

[13] Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. In *ISAAC (1)*, pages 253–265, 2010.

[14] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In *STOC*, pages 715–724, 2009.

[15] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems of boolean domain. In *SODA*, pages 1714–1728, 2011.

[16] Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.

[17] Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. The complexity of weighted boolean #csp. *CoRR*, abs/0704.3683, 2007.

[18] Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for boolean #csp. *J. Comput. Syst. Sci.*, 76(3-4):267–277, 2010.

[19] Martin E. Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6), 2007.

[20] Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3-4):260–289, 2000.

[21] Martin E. Dyer and David Richerby. On the complexity of #CSP. In *STOC*, pages 725–734, 2010.

[22] Tomás Feder and Moshe 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.

[23] Michael Fisher and Harold Temperley. Dimer problem in statistical mechanics-an exact result. *Philosophical Magazine*, 6:1061–1063, 1961.

[24] Michael Freedman, László Lovász, and Alexander Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. 2007.

[25] Zhiguo Fu and Jin-Yi Cai. Holographic algorithms on domain size k $>$ 2. In *TAMC*, pages 346–359, 2012.

[26] Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. *CoRR*, abs/0804.1932, 2008.

[27] Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the tutte polynomial. *Inf. Comput.*, 206(7):908–929, 2008.

[28] Pavol Hell and Jaroslav Nesetril. On the complexity of h-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.

[29] Francois Jaeger, Dirk L. Vertigan, and Dominic J. A. Welsh. On the computational complexity of the jones and tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

[30] Peter Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artif. Intell.*, 101(1-2):251–265, 1998.

[31] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

[32] Peter Jeavons and Martin C. Cooper. Tractable constraints on ordered domains. *Artif. Intell.*, 79(2):327–339, 1995.

[33] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[34] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[35] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[36] Richard M. Karp and Michael Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64, 1983.

[37] Pieter Kasteleyn. Dimer statistics and phase transitions. *Journal of Mathematical Physics*, 4:287–293, 1963.

[38] Pieter Kasteleyn. Graph theory and crystal physics. In *F. Harrary: Graph Theory and Theoretical Physics*, page 43–110. New York: Academic Press, 1967.

[39] Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[40] Pinyan Lu. Complexity dichotomies of counting problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:93, 2011.

[41] Colin McQuillan. Lsm is not generated by binary functions. *CoRR*, abs/1110.0461, 2011.

[42] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[43] Christos M. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[44] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.

[45] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[46] Leslie G. Valiant. Holographic algorithms (extended abstract). In *FOCS*, pages 306–315, 2004.

[47] Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

[48] Dominic J. A. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[49] Tomoyuki Yamakami. Approximate counting for complex-weighted boolean constraint satisfaction problems. In *WAOA*, pages 261–272, 2010.

[50] Tomoyuki Yamakami. Approximation complexity of complex-weighted degree-two counting constraint satisfaction problems. In *COCOON*, pages 122–133, 2011.

[51] Tomoyuki Yamakami. Optimization, randomized approximability, and boolean constraint satisfaction problems. In *ISAAC*, pages 454–463, 2011.

[52] Jin yi Cai and Vinay Choudhary. Some results on matchgates and holographic algorithms. In *ICALP (1)*, pages 703–714, 2006.

[53] Jin yi Cai and Vinay Choudhary. Valiant's holant theorem and matchgate tensors. In *TAMC*, pages 248–261, 2006.

[54] Jin yi Cai, Vinay Choudhary, and Pinyan Lu. On the theory of matchgate computations. In *IEEE Conference on Computational Complexity*, pages 305–318, 2007.

[55] Jin yi Cai and Pinyan Lu. Bases collapse in holographic algorithms. In *IEEE Conference on Computational Complexity*, pages 292–304, 2007.

[56] Jin yi Cai and Pinyan Lu. Holographic algorithms: from art to science. In *STOC*, pages 401–410, 2007.

[57] Jin yi Cai and Pinyan Lu. Holographic algorithms: The power of dimensionality resolved. In *ICALP*, pages 631–642, 2007.

[58] Jin yi Cai and Pinyan Lu. On block-wise symmetric signatures for matchgates. In *FCT*, pages 187–198, 2007.

[59] Jin yi Cai and Pinyan Lu. On symmetric signatures in holographic algorithms. In *STACS*, pages 429–440, 2007.

[60] Jin yi Cai and Pinyan Lu. Holographic algorithms with unsymmetric signatures. In *SODA*, pages 54–63, 2008.

[61] Jin yi Cai and Pinyan Lu. Signature theory in holographic algorithms. In *ISAAC*, pages 568–579, 2008.

[62] Jin yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by fibonacci gates and holographic reductions for hardness. In *FOCS*, pages 644–653, 2008.

[63] Jin yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms with matchgates capture precisely tractable planar #csp. In *FOCS*, pages 427–436, 2010.