CONSTRAINT SATISFACTION PROBLEMS AND THEIR REDUCTION TO DIRECTED GRAPHS

by

Muhanda Stella Mbaka Muzala

B.A.(Hons) Mathematics and Computer Science, York University, 2007

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Applied Mathematics

Toronto, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

CONSTRAINT SATISFACTION PROBLEMS AND THEIR REDUCTION TO DIRECTED GRAPHS

Master of Science 2012

Muhanda Stella Mbaka Muzala

Applied Mathematics

Ryerson University

Abstract

Constraint satisfaction problems present a general framework for studying a large class of algorithmic problems such as satisfaction of Boolean formulas, solving systems of equations over finite fields, graph colourings, as well as various applied problems in artificial intelligence (scheduling, allocation of cell phone frequencies, among others.) CSP (Constraint Satisfaction Problems) bring together graph theory, complexity theory and universal algebra.

It is a well known result, due to Feder and Vardi, that any constraint satisfaction problem over a finite relational structure can be reduced to the homomorphism problem for a finite oriented graph. Until recently, it was not known whether this reduction preserves the type of the algorithm which solves the original constraint satisfaction problem, so that the same algorithm solves the corresponding digraph homomorphism problem.

We look at how a recent construction due to Bulin, Delić, Jackson, and Niven can be used to show that the polynomial solvability of a constraint satisfaction problem using Datalog, a programming language which is a weaker version of Prolog, translates from arbitrary relational structures to digraphs.

Acknowledgements

I would like to thank Ryerson University for allowing me to complete this great milestone in my education. The professors in the Department of Mathematics have been of great support in this journey to complete my graduate studies. I would like to thank Dr. Anthony Bonato, whose brilliant teaching and interest in Graph Theory have captured my attention. Thanks to Professor Katrin Rohlf and Silvana Ilie for being an inspiration as female experts in the field of Mathematics.

My deepest thanks go to my supervisor Dr. Dejan Delić. I believe no one could have been a better candidate to be my mentor. Without his guidance and expertise this thesis would not have been sealed.

I would not close this by omitting to thank my entire family. Thanks to the man of my life, Jean-Robert Djulus, for his ceaseless support and believing in my brain. I sincerely thank my dearest parents, my father, Ambroise Mbaka and my mother Marie-Claire Ngoya who have always been the voice of encouragement from my early childhood.

Thanks to all for investing in me and leaving your footprints on this unforgettable journey.

Contents

1 Introduction **1**

1.1 Graphs and digraphs 3

1.2 Homomorphism and adjacency 4

1.3 Homomorphisms and colourings 7

2 Complexity **11**

2.1 Polynomial time reduction 14

3 CSP **16**

3.1 Primitive positive formulas 18

3.2 More properties of relational structures 19

4 Bounded Width **21**

4.1 Consistency check 25

4.2 Pair consistency check 28

4.3 Majority polymorphism 29

5 CSP Dichotomy **34**

5.1 Reduction digraph 35

5.2 Relational structures 36

5.3 Reduction to digraphs 40

5.3.1 Translation to a bipartite structure 41

5.3.2 Transformation to digraph 42

6 Preservation of Bounded Width **45**

7 Conclusion and Open Problems **49**

7.1 Summary 49

7.2 Open problems 50

7.2.1 Datalog and LFP 51

Bibliography **57**

List of Figures

1.1 A graph with edges and a digraph with arcs. 3

1.2 A mapping f from C_7 to C_5 5

1.3 Each vertex in C_7 has an image in C_5. 5

1.4 Balanced graph G 8

1.5 Balanced graph G with its level function 9

2.1 Classes **P** and **NP**. 13

2.2 A diagram for three known **NP** problems. 15

3.1 A digraph not admitting a weak near unanimity polymorphism. 20

4.1 A 2-colouring of even and odd cycles. 23

4.2 Consistency check performed on two oriented paths. 25

4.3 A digraph admitting the majority polymorphism. 30

4.4 Connectivity of triples built from $0, 1$, and 2. 31

5.1 A minimal path. 35

5.2 Levels in the minimal path from Figure 5.1. 36

5.3 The oriented path N. 38

5.4 Process diagram for reduction to digraph. 40

5.5 Path representing $(x_1, (x_1, y_1))$. 43

5.6 Path representing $(x_1, (y_1, x_1))$. 43

Chapter 1

Introduction

The central topic of this thesis are Constraint Satisfaction Problems and their reduction to digraph homomorphisms. Constraint satisfaction problems play an important role in the area of computer science (in particular, artificial intelligence) and have attracted significant attention in the past 30 years, using methods and tools from logic, universal algebra, and combinatorics.

A paper by Feder and Vardi [7], resulted in a famous conjecture asserting that any constraint satisfaction problem over a finite relational structure is either tractable (solvable in polynomial runtime) or intractable (**NP**-complete). So far, all gathered evidence points to the conjecture being true. In addition, Feder and Vardi showed in the same exposition that every constraint satisfaction problem over a finite structure can be reduced to the homomorphism problem for a finite balanced digraph. We refer the reader to [10]

for a detailed study of graph and digraph homomorphisms.

Recently, algebraic methods have been explored in the study of constraint satisfaction problems and their complexity (see [3], [4]). One of the most important results is the categorization of constraint satisfaction problems that are solvable using a simple class of algorithms, called bounded width problems [2]. A recent result of Bulin, Delić, Jackson, and Niven [5] shows that the original Feder and Vardi reduction to digraphs can be modified so that a variety of polymorphisms are preserved. In particular, in their paper, it is shown that this transformation reduces structures of bounded width to digraphs of bounded width. The main result of this thesis is to give an alternative proof of this result using the Datalog approach to prove bounded width. This is in contrast to a purely algebraic approach used in [5] and [1], to give a similar proof for Feder and Vardi construction.

1.1 Graphs and digraphs

To introduce the notion of CSPs (Constraint Satisfaction Problems) and their reduction to digraphs, we must first consider some definitions related to graphs and digraphs. Our main references for the material of this chapter are [10] and [19].

Definition 1.1. A *graph* G is a non-empty set of vertices denoted by $V(G)$ together with a set of edges $E(G)$, where an edge joins two given vertices u and v. For this reason, we often say that the edge relation is a symmetric binary relation on $V(G)$. If u and v are joined by an edge then these vertices are *adjacent*.

Definition 1.2. A *digraph* G is a non-empty set of vertices denoted by $V(G)$ together with a set of arcs $E(G)$. An *arc* is marked by an orientation of the edge.

Definition 1.3. If u and v are adjacent in the direction from u to v, we say that u is an *in-neighbour* of v and consequently v is an *out-neighbour* of u.

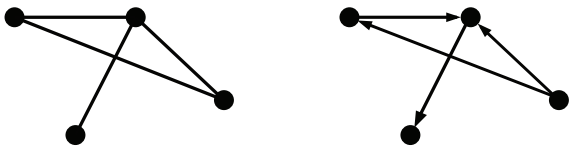

Figure 1.1: A graph with edges and a digraph with arcs.

Definition 1.4. Given two (di)graphs G and H, a mapping f from G to H is called a *homomorphism*, if (arcs) edges of G can be mapped to (arcs) edges in H while preserving

vertex relations. We have:

$$\text{for all } u, v \in E(G),\ f(u)f(v) \in E(H)$$

Homomorphisms will preserve edge relations in a graph but will also preserve the direction of the arc in a digraph.

A homomorphism f between G and H is commonly denoted "$G \to H$", where G is said to be *homomorphic to* H or that G is H-colourable.

1.2 Homomorphism and adjacency

As stated in the previous section, homomorphisms preserve adjacency. We can introduce more definitions related to graphs.

Definition 1.5. Given two vertices u and v of G, a *walk* from u to v is a sequence of adjacent vertices allowing a connection from u to v.

Definition 1.6. In a *closed walk* the starting vertex and the ending vertex are the same.

Definition 1.7. A *path* P from u to v is a walk consisting of distinct vertices. If m edges are traversed in the sequence from u to reach v, then we say that the path P is of length m, denoted P_m.

Definition 1.8. A *cycle* C is a sequence of distinct vertices, with the starting vertex

and the ending vertex being the same. If m edges are traversed in the sequence from u back to u, then we say that the cycle C is of length m, denoted C_m.

Proposition 1.1. (Hell, Nešetřil [10]) A mapping $f : V(C_m) \to V(G)$ is a homomorphism of C_m to G if and only if $f(0), f(1), \ldots, f(m-1)$ is a closed walk in G.

Proof. We refer the reader to [10] for a complete proof. □

In Example 1.1 we will demonstrate that a mapping f between C_7 and C_5 preserves adjacency.

Example 1.1. $f : C_7 \to C_5$

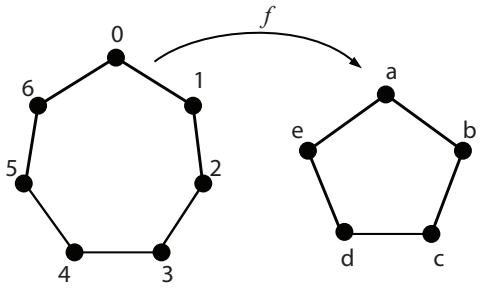

Figure 1.2: A mapping f from C_7 to C_5

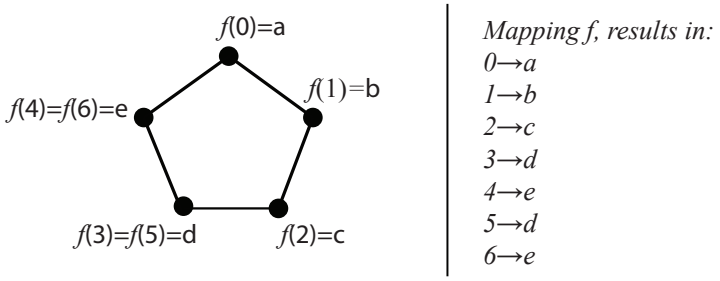

Figure 1.3: Each vertex in C_7 has an image in C_5.

We see the closed walk $f(0), f(1), \ldots, f(6), f(0)$ in C_5. All edge relations in C_7 are preserved in C_5 after performing the mapping.

The above definitions are important tools in the study of homomorphism and adjacency. Similar definitions and properties exist for digraphs, which we proceed to introduce next.

Definition 1.9. Digraphs are often referred to as *oriented* graphs.

Definition 1.10. To distinguish a graph from a digraph, we refer to it as an *undirected* graph.

Definition 1.11. Given an oriented walk of m vertices, adjacent in the following sequence $\{v_1, v_2, \ldots, v_m\}$, a $forward\ arc$ is defined as an arc from v_i to v_{i+1} and a *backward arc* is an arc from v_{i+1} to v_i.

Definition 1.12. When a path or a walk is composed of both forward and backward arcs, we refer to it as an *oriented* path or walk. When it is composed of forward arcs only, we say that the walk or the path is *directed*.

Definition 1.13. A digraph is *connected* if any given two vertices are joined by an oriented walk.

Definition 1.14. If a denotes the number of forward arcs and b the number of backward arcs of an oriented walk, then the *algebraic length* of the walk is calculated as $a - b$.

Proposition 1.2. (Hell, Nešetřil [10]) Let G and H be digraphs, and $f : G \to H$ a homomorphism. If $v_0, v_1, \ldots, v_m$ is a walk in G, then $f(v_0), f(v_1), \ldots, f(v_m)$ is a walk in H of the same algebraic length.

Proof. We refer the reader to [10] for the complete proof. □

Definition 1.15. *An oriented path* and *cycle*, denoted $\overrightarrow{P_m}$ and $\overrightarrow{C_m}$ respectively, are defined similarly as the path P_m and cycle C_m, only that every $v_i v_{(i+1)}$ is an arc and not an edge.

Definition 1.16. A digraph G is *balanced* if any two oriented paths $\overrightarrow{P_1}$ and $\overrightarrow{P_2}$ between vertices u and v have the same algebraic length.

1.3 Homomorphisms and colourings

In the previous section we had mentioned that a homomorphism from G to H is also called an H-colouring of G. One of the most important and extensively studied problems in graph theory is that of graph colouring. It is a constraint satisfaction problem that requires colouring vertices of a given graph G such that the following constraint is satisfied: any two adjacent vertices should not be assigned the same colour.

We can define our mapping f as the assignment of m colours to vertices of G so that no pair of adjacent vertices is assigned the same colour. Following this concept, the graph H is the set of m vertices, where each vertex is labelled as a distinct colour from

its neighbour.

Definition 1.17. A graph of m vertices, is a *complete graph* (denoted K_m) if every pair of distinct vertices are connected by an edge.

Proposition 1.3. (Hell, Nešetřil [10]) A homomorphism $f : G \to K_m$ is the m-colouring of G.

Proof. We refer the reader to [10] for the complete proof. □

Definition 1.18. Given a connected balanced digraph G, starting at a given vertex a, we can label all vertices of G to produce levels. We start by labeling a as 0, then all its out-neighbours incremented by 1 and in-neighbours decremented by 1. If applying this procedure results in some negative labels, then we can add the same positive integer to all labels so that the smallest label is 0. The label of each vertex is called the *level* of the vertex. We call the *height* of G the maximum level of G.

Example 1.2. We start with G, then we proceed with the labeling

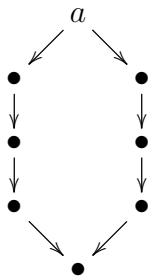

Figure 1.4: Balanced graph G

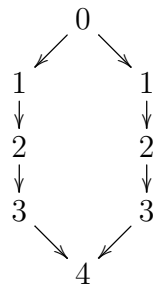

Figure 1.5: Balanced graph G with its level function

Proposition 1.4. (Hell, Nešetřil [10]) If G and H are two balanced digraphs of the same height, then any homomorphism of G to H preserves the levels of vertices.

Proof. We refer the reader to [10] for the complete proof. □

Definition 1.19. If $G \to H$ and $H \to G$, then we say that G and H are *homomorphically equivalent*.

Definition 1.20. Let H be a nonempty subset of $V(G)$. Then, the graph with the vertex set H and the set of edges $E(G) \cap (H \times H)$ is called an *induced subraph* of G, which we will also denote H.

Definition 1.21. Suppose H is an induced subgraph of G, a *retraction* from G to H is a homomorphism $f : G \to H$ such that each vertex u in H is mapped to itself in G.

Definition 1.22. Let H be an induced subgraph of G. An *inclusion* from H to G is a homomorphism $g : H \to G$, which maps each vertex u in H to itself.

By composing the retraction and the inclusion in H and G, we can conclude that G and H are homomorphically equivalent.

Definition 1.23. A *core* is the smallest induced subgraph onto which the graph can retract.

Every finite graph must contain a core. For a proof, we refer the reader to [10].

Chapter 2

Basic Notions from Computational Complexity Theory

Algorithms are built to determine the existence of homomorphisms. If we consider homomorphism problems based on the simplicity or complexity of the problem, then we can classify them as **P** or **NP** problems. Our main reference for the material in this chapter is [17].

Definition 2.1. A problem or a function $f : \mathbb{N} \to \mathbb{N}$ is *computable in polynomial time* (we say that f is in the class **P** or f is *tractable*) if there exists at least one algorithm $\mathcal{A}$ which can solve $f(n)$ in a finite number of steps that is bounded by a polynomial function of n, the length of the input. **P** stands for the class of problems (functions) computable in polynomial time.

Example 2.1. The following are in **P**:

1. Solving a linear equation over the field of real numbers $\mathbb{R}$ ([17]):

   Given $I = \{\{a, b, c\} \in \mathbb{R} | c - b + a = x, x \in \mathbb{R}\}$, the problem of finding all triples (a, b, c) such that $c - b + a = x$ can be solved by a polynomial algorithm, the usual Gaussian elimination;

2. 2-colouring of C_n, with n an even integer ([17]),

   This can be achieved by assigning colours 1 and 2 respectively to adjacent vertices.

Definition 2.2. A function $f : \mathbb{N} \to \mathbb{N}$ is in **NP**, if there exists at least one algorithm $\mathcal{A}$ that verifies in polynomial time whether a proposed solution a is the correct solution to $f(n)$. This algorithm does not calculate the solution to the problem, it simply verifies its correctness. **NP** stands for non-deterministic polynomial time.

Following these definitions, we know for certain that **P** is a subset of **NP**. Indeed, if we can find a solution to a function f in polynomial time, then we can also verify its correctness in polynomial time.

Example 2.2. The following problems are in **NP** but not known to be in **P**:

1. 3-satisfiability in Boolean logic ([17]);

2. Traveling Salesman Problem (TSP) ([17]):

   Given a list of distinct cities $c_1, c_2, \ldots, c_n$, the problem involves finding the shortest

distance, a cycle of length l from c_1 back to c_1, while visiting each city exactly once. Observe that $l \leq \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} d(c_i, c_j)$, where $d(c_i, c_j)$ is the distance in miles between c_i and c_j.

The relationship between the classes **P** and **NP** is one of the central problems of the theory of computational complexity. Though **P** is a subset of **NP**, the question remains to determine if these two classes are identical. **P**=**NP** means that for every problem we can find an algorithm that verifies the correctness of a solution to the problem, but also an algorithm that can find this solution, if unknown, and this performed in polynomial time. The most famous conjecture in this area is the so-called "**P** $\neq$ **NP**", which is still open.

Conjecture 1. (Cook, [6]) **P** $\neq$ **NP**.

The following diagram represents the relationship between the classes **P** and **NP**:

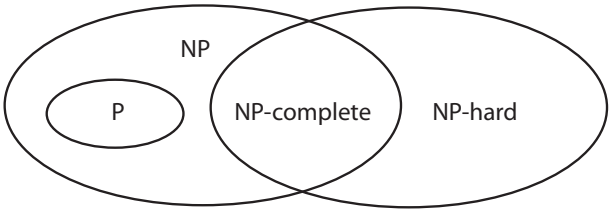

Figure 2.1: Classes **P** and **NP**.

2.1 Polynomial time reduction

Definition 2.3. Given two problems C and D. We say that C is *polynomial time* or *poly-time reducible* to D (denoted by $C \leq D$) if there exists a polynomial time computable function $f : C \to D$ such that for every $x \in C$, we have that $f(x) \in D$.

Definition 2.4. We say that two problems C and D are *poly-time equivalent* if $C \leq D$ and $D \leq C$.

Definition 2.5. Given a class of problems $\mathcal{C}$ we shall say that C_1 (a problem) is $\mathcal{C}$-*hard* if

$$\text{there exists } C_2 \in \mathcal{C} \text{ such as } C_2 \leq C_1.$$

If an algorithm exists that is capable to transform C_2 to C_1 in polynomial time, then C_1 is at least as hard as $C_2 \in \mathcal{C}$.

Definition 2.6. We say that a problem C_1 is **NP**-*complete* if it is **NP**-hard and in **NP** (see Figure 2.1).

There exists several notable problems that are **NP**-complete, one of them being the SAT problem. The SAT problem consists of all satisfiable Boolean formulas using boolean symbols $\wedge, \vee, \neg$ and variables. SAT can be reduced in polynomial time to other problems in order to prove their **NP**-completeness. We state the following fundamental theorems but their proofs are rather involved and can be found in [17]

Theorem 2.1. *(See [17]) SAT is* **NP***-complete.*

Theorem 2.2. *(See [17]) 3-SAT is* **NP***-complete*

Theorem 2.3. *(See [17]) 3-colourability of graphs is* **NP***-complete.*

By definition, 3-SAT is the problem to determine satisfiability of a Boolean formula, which is a conjunction of disjunctions, where each disjunction consists of at most three terms, each being a variable or a negation of a variable.

Figure 2.2 depicts the relationship between SAT, 3-SAT and 3-colourability of a graph.

Figure 2.2: A diagram for three known **NP** problems.

Chapter 3

CSP for relational structures

Definition 3.1. Let A be a nonempty set. A finitary *relation* R on A is any

$$R \subseteq A^n = A \times \cdots \times A$$

for some n greater or equal to 0. In that case, we say that R is n-ary and call n the *arity* of R. A^n is called the n-th power of A or the Cartesian product on A.

Definition 3.2. The *arity* of R is the number of variables on which it is defined. For example, E, the symbol for edge relation is a binary relation so its arity is 2.

Definition 3.3. A *relational structure* $\mathbb{S}$ is a nonempty set S equipped with a collection of relations R_i, with $i = 1, 2, \ldots, k$. We write:

$$\mathbb{S} = (S; R_1, R_2, \ldots, R_k)$$

Definition 3.4. Two relational structures $\mathbb{S}$ and $\mathbb{T}$ are *similar* if they have the same set of relation symbols and arities.

Definition 3.5. Let T be a nonempty subset of S. T, along with the restrictions of relations of $\mathbb{S}$, form a structure $\mathbb{T}$, and we say that $\mathbb{T}$ is a *substructure* of $\mathbb{S}$ or $\mathbb{T}$ is *included* in $\mathbb{S}$. The inclusion mapping $\mathbb{T} \to \mathbb{S}$ is also a homomorphism.

Following are some properties of a homomorphism $f : \mathbb{S} \to \mathbb{T}$:

1. f is an *isomorphism* if there exists an inverse homomorphism $\mathbb{T} \to \mathbb{S}$.

2. f is an *endomorphism* if $\mathbb{S} = \mathbb{T}$.

3. f is an *automorphism* if it is both an endomorphism and an isomorphism.

A relational structure $\mathbb{S}$ is a *core* if all of its endomorphisms are automorphisms.

Definition 3.6. Let $\mathbb{S}$ be a relational structure. We define a *constraint satisfaction problem of* $\mathbb{S}$ as the collection of all relational structures denoted CSP($\mathbb{S}$) which consists of all structures that are homomorphic to $\mathbb{S}$. CSPs are therefore problems of finding homomorphisms between two given general relational structures $\mathbb{S}$ and $\mathbb{T}$.

The constraint satisfaction problem of $\mathbb{S}$ is in **NP** for all $\mathbb{S}$. In fact, any given homomorphism between $\mathbb{S}$ and $\mathbb{T}$ can easily be encoded in a string whose length is linear in $|S|$ and verifying its correctness can be done in polynomial runtime. Thus, CSP($\mathbb{S}$) is in **NP**. There is a famous conjecture by Feder and Vardi which relies on the assumption that **P**$\neq$**NP**.

Conjecture 2. (Feder, Vardi, [7]) Every CSP is either **P** or **NP**-complete.

3.1 Primitive positive formulas

Definition 3.7. An *atomic formula* is a formula of the form $R(x_1, \ldots, x_n)$, where R is an n-ary relation symbol. Given the class of all atomic formulas μ in some fixed signature, we can produce a formula:

$$\exists\alpha_1 \ldots \exists\alpha_n, \phi_1(*, \ldots, *) \wedge \cdots \wedge \phi_n(*, \ldots, *),$$

where the stars are free variables (say, $x_1, \ldots, x_m$) and $\alpha_1 \ldots \alpha_n$ are bound variables. All the ϕ_i belong to μ and are operation symbols.

If the class μ consists of formulas of the form $R(x_1, x_2, \ldots, \alpha_1, \ldots, \alpha_n)$, where R is a k-ary relation, then we call a formula $\exists\alpha_1, \ldots, \exists\alpha_n[\phi_1(*, \ldots, *) \wedge \cdots \wedge \phi_n(*, \ldots, *)]$ *primitive positive*.

Example 3.1. If E is the binary relation of an undirected graph, then an example of a primitive positive formula in the language of graphs is, for instance,

$$\exists x \exists y \exists z (xEy \wedge yEz).$$

3.2 More properties of relational structures

Definition 3.8. Given a relational structure $\mathbb{S} = (S, R)$, a k-ary function f defined on the set S, where $k \geq 1$, is called a *polymorphism* if it preserves all relations in R.

Definition 3.9. We say that an n-ary polymorphism $f : S^n \to S$ is *idempotent* if we have that

$$f(x, x, \ldots, x) = x, \text{ for every } x \in S.$$

Definition 3.10. Let S be a finite relational structure. We say that a polymorphism $f(x_1, \ldots, x_n)$, $n \geq 2$, is a *weak near-unanimity polymorphism* if

1. $f(x, \ldots, x) = x$,

2. $f(x, \ldots, x, y) = f(x, \ldots, x, y, x) = \cdots = f(y, x, \ldots, x)$.

Using a result of Maróti and McKenzie [16], Larose and Zadóri were able to prove the following theorem, which gives a general criterion for a relational structure to have an **NP**-complete CSP.

Theorem 3.1. *(Larose, Zadóri [15]). Let $\mathbb{S}$ be a finite relational structure which does not admit a weak near-unanimity polymorphism. Then there exists a finite relational structure $\mathbb{T}$ whose only idempotent polymorphisms are projections, such that CSP($\mathbb{T}$) is poly-time reducible to CSP($\mathbb{S}$). Therefore, if $\mathbb{T}$ does not have a weak near-unanimity polymorphism, CSP($\mathbb{S}$) is* **NP***-complete.*

Proof. We refer the reader to [15] for the complete proof. □

Due to Theorem 3.1, the Dichotomy Conjecture of Feder and Vardi can be reformulated as follows:

Conjecture 3. (Bulatov, Jeavons and Krokhin [4]) Let $\mathbb{S}$ be a finite relational structure, then CSP($\mathbb{S}$) is in **P** if and only if $\mathbb{S}$ admits a weak near-unanimity polymorphism.

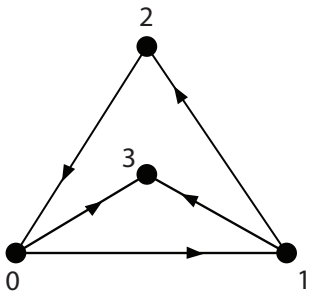

Figure 3.1: A digraph not admitting a weak near unanimity polymorphism.

Example 3.2. It can be shown that the digraph in Figure 3.1 does not admit a weak near unanimity polymorphism and consequently, its homomorphism problem is **NP**-complete [10].

Chapter 4

Datalog and Bounded Width

Definition 4.1. A *Datalog program* is a finite set of rules of the form:

$$t_0 \text{ :- } t_1, t_2, \ldots, t_n$$

where ":-" separates the head and the body of the clause and each t_i, $i = 0, 1, \ldots, n$ is an atomic formula $R(x_{i_1}, \ldots, x_{i_k})$. Then we call t_0 the *head* of the rule, and the sequence t_1, t_2,..., t_n the *body* of the rule.

Definition 4.2. The *signature* of the input of the program is the language of a relational structure. For example, the signature of a graph is a single binary relation.

Definition 4.3. The predicates (atomic formulas) or statements in the heads of the rules are not from the signature and are called $IDBs$ (or *intentional database predicates*)

while all other predicates come from the signature and are called $EDBs$ (or *extentional database predicates*).

One of the IDBs, which is usually nullary, as it does not depend on any variables, is designated as the *goal predicate* of the program. The goal predicate is assumed to be initially set to FALSE and we say that a Datalog program accepts a structure $\mathbb{S}$ if its goal predicate evaluates to TRUE on $\mathbb{S}$.

Example 4.1. The following Datalog program accepts an undirected graph if and only if the graph is not 2-colourable. In essence, the program is searching for the occurence of an odd cycle, as odd cycles are not 2-colourable. Once such a cycle is found, the program evaluates to TRUE.

$$Odd(X, Y) : -E(X, Y)$$

$$Odd(X, Y) : -Odd(X, Z), E(Z, T), E(T, Y)$$

$$non2col : -Odd(X, X)$$

The first clause provides the definition of an odd path between X and Y. The second clause is used to define odd paths recursively; it does so by adding two edges to an odd path, which also results in an odd path. The goal predicate decides whether the path between X and X (a cycle) is either odd or even.

In this example, Odd and $non2col$ are the IDBs and $non2col$ is the program's goal

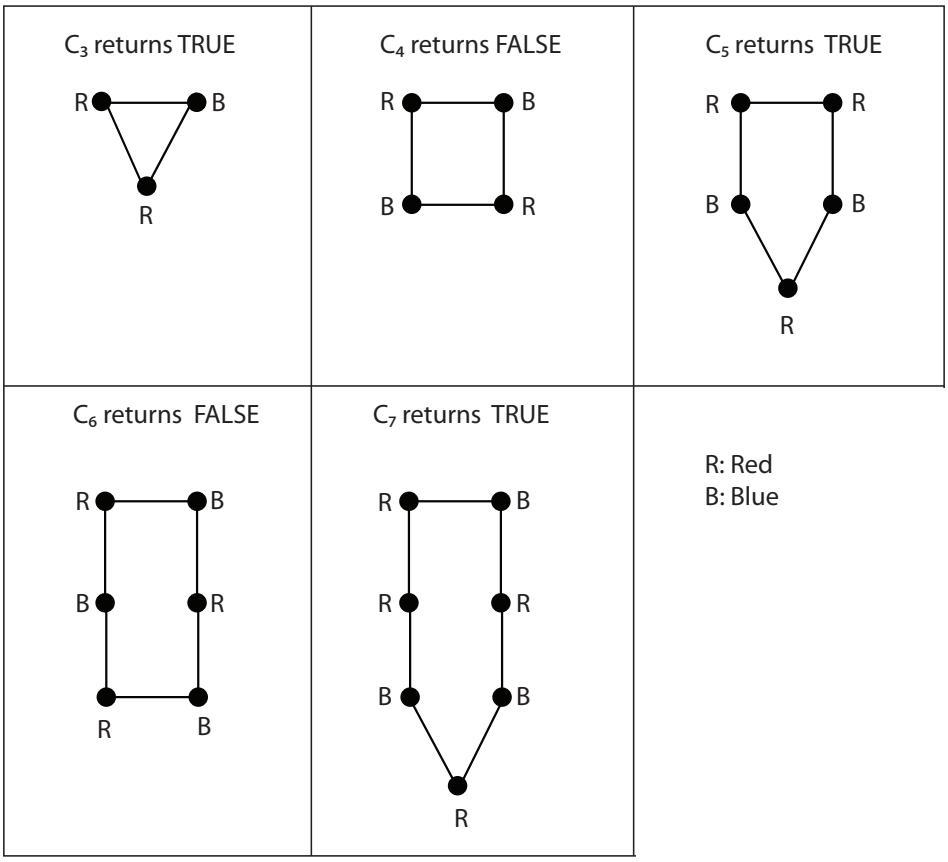

Figure 4.1: A 2-colouring of even and odd cycles.

predicate. The only EDB is the signature predicate E.

EDBs consist of the relations provided by the signature, while IDBs are "new" relations derived from EDBs. An important observation is that each program line or clause is, in fact, a positive primitive formula, which defines an IDB in terms of basic structure relations.

Definition 4.4. Given $\mathbb{S}$ a finite relational structure with a finite signature ρ, we define the class $coCSP(\mathbb{S})$ to consist of all finite structures $\mathbb{T}$ in the same signature as $\mathbb{S}$ such

that $\mathbb{T}$ is not homomorphic to $\mathbb{S}$.

Definition 4.5. We say that a finite relational structure $\mathbb{S}$, with a finite signature ρ, is of *bounded width* if there exists a Datalog program which accepts precisely the structures from coCSP($\mathbb{S}$).

Example 4.2. [10]

1. If $\mathbb{S}$ is a 2-colourable undirected graph or a 2-colourable digraph, then $\mathbb{S}$ is of bounded width.

2. Any finite undirected graph or a digraph containing a loop is of bounded width.

3. For every $k \geq 2$, $\overrightarrow{C_k}$ is of bounded width. It can be shown ([10]) that $G \not\to \overrightarrow{C_k}$ if and only if $\overrightarrow{C_m} \to G$ for some oriented cycle of algebraic length m not divisible by k. So, coCSP($\overrightarrow{C_k}$) consists of all digraphs G which contain oriented cycles whose algebraic length is not divisible by k. The Datalog program which accepts coCSP($\overrightarrow{C_k}$) can be constructed similarly to the one from Example 4.1.

Proposition 4.1. (Hell, Nešetřil [10]) If a finite relational structure $\mathbb{S}$ is of bounded width, then CSP($\mathbb{S}$) is in **P**.

Proof. Fix a finite relational structure $\mathbb{S}$ with a finite signature ρ so that $\mathbb{S}$ is of bounded width. There exists a Datalog program which accepts coCSP($\mathbb{S}$). Suppose $\mathbb{T}$ is a finite relational structure in signature ρ. Since a Datalog program always terminates its run

in a number of steps which is a polynomial function in $|T|$, we can decide in polynomial time whether $\mathbb{T}$ is homomorphic to $\mathbb{S}$ or not. □

For digraphs, the run of a Datalog program can be formulated in terms of a purely graph-theoretic algorithm. Such algorithms are called *consistency checks*.

4.1 Consistency check

Given two digraphs G and H with $u_i \in V(G), i = 1, 2, 3 \ldots, |V(G)|$. We define a vertex-to-vertex assignment, denoted by "$\rightsquigarrow$", where $u_i \rightsquigarrow v_j$, with $v_j \in V(H)$, and $j = 1, 2, 3 \ldots, |V(H)|$. Initially the possible vertices candidates for the assignment to u_i constitute the domain $\rightsquigarrow_{u_i}$, the list of all vertices v_j in H.

This is how the *consistency check* works:

Remove vertex v_j from $\rightsquigarrow_{u_i}$ if for u_{i+1} (adjacent to u_i in a given direction), there is no v_{j+1} in $\rightsquigarrow_{u_{i+1}}$, adjacent to v_j in the same direction as $u_i u_{i+1}$. We say that the domain of each vertex in G is *consistent* if it remains nonempty after possible changes (removals). It is said to be *inconsistent* if the domains become empty, meaning that all vertices were removed from their lists.

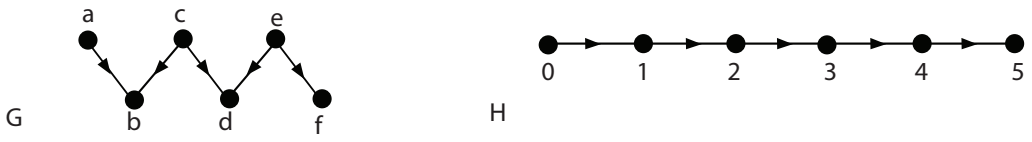

Figure 4.2: Consistency check performed on two oriented paths.

Example 4.3. Given the two oriented paths G and H from Figure 4.2, let us check if they are consistent.

Initially the domains are:

$\rightsquigarrow_a = \{0, 1, 2, 3, 4, 5\}$

$\rightsquigarrow_b = \{0, 1, 2, 3, 4, 5\}$

$\rightsquigarrow_c = \{0, 1, 2, 3, 4, 5\}$

$\rightsquigarrow_d = \{0, 1, 2, 3, 4, 5\}$

$\rightsquigarrow_e = \{0, 1, 2, 3, 4, 5\}$

$\rightsquigarrow_f = \{0, 1, 2, 3, 4, 5\}$

We now consider $\rightsquigarrow_a$ and check for a possible change. Since there exists an arc ab we must remove any vertex from the domain that does not offer the same adjacency in H, thus, 5 needs to be removed. After applying the check to each domain, the lists are nonempty and consequently consistent:

$\underset{\downarrow}{\rightsquigarrow_a} = \{0, 1, 2, 3, 4, \not{5}\}$

$\underset{\uparrow}{\rightsquigarrow_b} = \{\not{\emptyset}, 1, 2, 3, 4, 5\}$

$\underset{\downarrow}{\rightsquigarrow_c} = \{0, 1, 2, 3, 4, \not{5}\}$

$\underset{\uparrow}{\rightsquigarrow_d} = \{\not{\emptyset}, 1, 2, 3, 4, 5\}$

$\underset{\downarrow}{\rightsquigarrow_e} = \{0, 1, 2, 3, 4, \not{5}\}$

$\underset{\uparrow}{\rightsquigarrow_f} = \{\not{\emptyset}, 1, 2, 3, 4, 5\}$

The arrow index below each vertex of G shows the direction of the arc.

Lemma 4.1. *(Hell, Nešetřil [10]) If G is H-colourable, then the consistency check will succeed.*

Proof. We refer the reader to [10] for the complete proof. □

Example 4.4. We have previously established that any even cycle is 2-colourable. Let us consider C_6 and K_2.

Initially the domains are :

$\rightsquigarrow_a = \{red, blue\}$, $\rightsquigarrow_b = \{red, blue\}$, $\rightsquigarrow_c = \{red, blue\}$, $\rightsquigarrow_d = \{red, blue\}$, $\rightsquigarrow_e = \{red, blue\}$, $\rightsquigarrow_f = \{red, blue\}$.

After applying the consistency check, the list remains intact as neither *red* nor *blue* needs to be removed, because the edge relation constraint is satisfied at every step of the check.

Theorem 4.1. *(Hell, Nešetřil [10]) If H is an oriented path, then H is of bounded width.*

Proof. Example 4.3 confirms this theorem, but we will prove it in its generality.

A theorem from Hell, Nešetřil [10] states that a given H is of bounded width if and only if there exists G such that $G \to H$ when the consistency check performed on G succeeds. Using this notion on H, an oriented path, we will use the consistency check. If it succeeds, then we will define an H-colouring and thus prove that H is of bounded width.

Let us label all the vertices of the path H as $1, 2, 3, \ldots, |V(H)|$. If indeed the consistency check succeeds on H, then $\rightsquigarrow_u$, for all $u \in V(G)$ will be nonempty but will contain at least one vertex of H. And the list being consistent means that for any given vertex

v_1 in $\rightsquigarrow_{u_i}$, there must exist its adjacent vertex v_2 in $\rightsquigarrow_{u_{i+1}}$. This is important to note for consistency, in order to ensure arc relations between $\rightsquigarrow_{u_i}$ and $\rightsquigarrow_{u_{i+1}}$. We claim that we can find an homomorphism f, by assigning to each $f(u)$ the corresponding smallest labeled vertex residing in the consistent domain $\rightsquigarrow_u$. This is our H-colouring. □

4.2 Pair consistency check

Pair consistency check applies to pairs of vertices. In the pair consistency check procedure, the arc-to-arc assignment is $(u_1, u_2) \rightsquigarrow (v_1, v_2)$, where u_1 is taken to v_1 and u_2 is taken to v_2 following arc adjacency and direction constraint.

Initially possible pairs of vertices, candidates for any assignment are all tuples $(v_1, v_2) \in V(H)$ with connecting arcs.

This is how the *pair consistency check* works:

Remove any pairs (v_1, v_2) from $\rightsquigarrow_{u_1,u_2}$, if there is no v_α in the tuples of the domain $\rightsquigarrow_{(u_1,u_3)}$ and $\rightsquigarrow_{(u_2,u_3)}$, assuming there exists an arc between vertices (u_1, u_2), (u_1, u_3) and (u_2, u_3) such that (v_1, v_α) belongs to $\rightsquigarrow_{(u_1,u_3)}$ and (v_2, v_α) belongs to $\rightsquigarrow_{(u_2,u_3)}$. We say that the domain of each pair of vertices is *consistent* if the list remains nonempty after possible changes (removals). It is said to be *inconsistent* if the domains become empty.

4.3 Majority polymorphism

Let $\mathbb{S}$ be a finite relational structure with a finite signature. We say that $\mathbb{S}$ admits a *majority polymorphism* if there exists a ternary polymorphism $m : \mathbb{S}^3 \to \mathbb{S}$ such that

$$m(x, x, y) = m(x, y, x) = m(y, x, x) = x, \text{ for all } x, y \in \mathbb{S}.$$

Theorem 4.2. *(Hell, Nešetřil, [10]) If H admits a majority polymorphism, then H has bounded width.*

Proof. The proof [10] uses the fact that if a given digraph admits a majority polymorphism, then the pair consistency check will be successful, and a homomorphism can be defined based on the success of the pair consistency check. □

Corollary 1. (Hell, Nešetřil, [10]) If H admits a majority polymorphism, then the H-colouring problem is polynomial time solvable.

Example 4.5. Using the digraph in Figure 4.3, let us define a majority polymorphism. We have the following possible scenarios

$$xyz = \begin{cases} (1) & xyz \text{ are some permutations of } \{0,1,2\}; \\ (2) & xyz \text{ are in } \{0,1,2\} \text{ but at least two terms are equal}; \\ (3) & xyz \text{ are all different but contains one instance of } 3; \\ (4) & xyz \text{ contain } 3 \text{ but are not all different}; \end{cases}$$

We are giving special attention to vertex 3 as it does not have any out-neighbours, it is

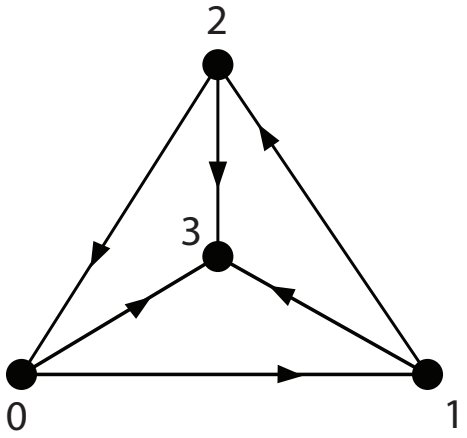

Figure 4.3: A digraph admitting the majority polymorphism.

a dead end. We will try to avoid going to 3.

(1) In consideration for the length of this thesis, we will consider only few permutations of $\{0, 1, 2\}$. The same operation can be applied to all other permutations.

Considering the order 012, we study out-neighbours of each vertex 0,1 and 2.

From $\boxed{0}$, we can either go to 1 or 3. Since the rule is to avoid the dead end, we choose $\boxed{1}$.

From $\boxed{1}$, we can either go to 2 or 3, we avoid 3 by choosing $\boxed{2}$.

From $\boxed{2}$, we will choose $\boxed{0}$.

Figure 4.4 represents a fragment of the Cartesian product graph G^3, which consists of the triples in $\{0, 1, 2\}^3$ all of whose coordinates are distinct:

We quickly notice the following pattern $\boxed{0} \to \boxed{1} \to \boxed{2} \to \boxed{0} \to \boxed{1} \to \boxed{2} \to \ldots$

Hence, the majority polymorphism here is defined as it would be in $\vec{C_3}$.

(2) Since two coordinates are equal, we take the majority element, $m(112) = 1$.

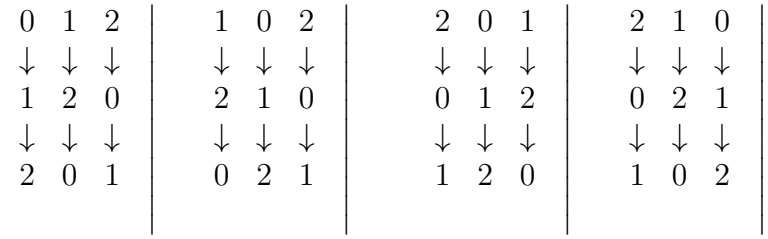

Figure 4.4: Connectivity of triples built from $0, 1$, and 2.

(3) With all xyz different but including 3, we decide to take $m(xyz) = 3$.

(4) Since two coordinates are equal, we take the majority element $m(223) = 2$.

The existence of a majority polymorphism for a given structure, among many other polymorphisms, is sufficient to prove the CSP can be solved in polynomial time.

Definition 4.6. Let $\mathbb{S}$ be a relational structure of finite signature. We say that a k-ary polymorphism $f : S^k \to S$ is a k-ary *near-unanimity* polymorphism on $\mathbb{S}$ if

$$f(xx \dots xy) = f(xx \dots yx) = \cdots = f(yx \dots xx) = x, \text{ for all } x, y \in \mathbb{S}$$

.

Theorem 4.3. *(Barto, Kozik [2]) Any finite relational structure which admits a k-ary near-unanimity polymorphism is of bounded width.*

Proof. We refer the reader to [2] for the complete proof. □

We are now in position to sketch the proof of the homomorphism dichotomy for undirected graphs, due to Hell and Nešetřil [10].

Theorem 4.4. *(Hell, Nešetřil, [10]) Let H be a graph admitting loops.*

(1) If H is bipartite or contains a loop, then the H-colouring problem is in $\boldsymbol{P}$.

(2) Otherwise, the H-colouring problem is $\boldsymbol{NP}$-complete.

Proof. (1) follows directly from Example 4.1 at the beginning of this section.

(2) Suppose $H = (V, E)$ is an undirected graph without any loops and containing an odd cycle. Let C be an odd cycle in H. The heart of the proof consists in showing that C cannot be compatible with a k-ary weak near-unanimity of any arity $k \geq 2$. The details of the proof are rather technical and involved and, therefore, omitted. □

Theorem 4.3 shows that if a finite relational structure admits a near-unanimity polymorphism, then it is of bounded width. This naturally leads us to the question whether the property of being of bounded width (having a set of homomorphism recursively defined by a Datalog program) can be characterized by the existence of a particular polymorphism. This conjecture, originally posed by Larose and Zadóri [15] was finally resolved recently through a very deep result of Barto and Kozik ([2]).

Theorem 4.5. *(Barto, Kozik [2]) For a finite relational structure $\mathbb{S}$, the following are equivalent.*

1. *$\mathbb{S}$ is of bounded width.*

2. $\mathbb{S}$ *admits a pair of weak near unanimity polymorphisms f,g of consecutive arities* $n, n+1$, *for some* $n \geq 3$.

3. $\mathbb{S}$ *admits a ternary weak near unanimity polymorphism* f *and a* 4*-ary weak near unanimity polymorphism* g, *such that* $f(xxy) = g(xxxy)$.

Proof. We refer the reader to [2] for the complete proof. □

Chapter 5

CSP Dichotomy Conjecture to Digraphs

Feder and Vardi [7] assert that, for every finite relational structure $\mathbb{S}$, CSP($\mathbb{S}$) is polynomial time equivalent to the homomorphism problem for a suitable balanced digraph $D_{\mathbb{S}}$, where $D_{\mathbb{S}}$ is such that the following equivalences are true: $CSP(\mathbb{S}) \Leftrightarrow CSP(D_{\mathbb{S}})$, $coCSP(\mathbb{S}) \Leftrightarrow coCSP(D_{\mathbb{S}})$, $RET(\mathbb{S}) \Leftrightarrow RET(D_{\mathbb{S}})$. RET is the retraction transformation.

We are interested in the representation of a finite relational structure in a language that is purely graph-theoretical followed by the translation of the finite relational structure to a suitable digraph to ensure optimal polymorphism preservation. Our main reference for the material in this chapter is [5]. In that paper, a simplification of the original reduction

of Feder and Vardi is given, and will be the main tool used to obtain the original result of the next chapter.

5.1 A suitable digraph for reduction

We mention essentially the results that were revealed by the work of [5] in their proof of the reduction of CSPs to digraphs. Let us contemplate some of the identified properties of a suitable digraph $D_{\mathbb{S}}$.

(1) $D_{\mathbb{S}}$ must be built from a collection of minimal paths.

Definition 5.1. A path P from vertex a to b is called *minimal* if P starts with two forward arcs from vertex a and ends with two forward arcs at b. The vertex a is called the *initial vertex* and b is called the *terminal verted* of P, denoted $init(P)$ and $term(P)$, respectively. Figure 5.1 is an example of a minimal path.

$$a\bullet \to \bullet \to \bullet \leftarrow \bullet \to \bullet \to \bullet \to \bullet \leftarrow \bullet \to \bullet \to \bullet\ b$$

Figure 5.1: A minimal path.

Furthermore, if we use the direction of the arcs to create levels for P, with a at level 0 and b at level 5, then in a minimal path when going from a to b we never revisit level 0 or reach level 5 before we get to b.

Forward arcs are usually counted as "1" and backward arcs as "0". Hence, P can be

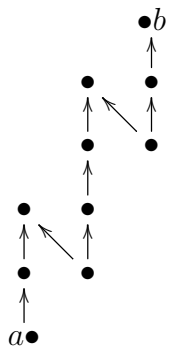

Figure 5.2: Levels in the minimal path from Figure 5.1.

annotated with the string 110111011.

(2) $D_{\mathbb{S}}$ **must be balanced and** h**-bipartite.**

Recall that a balanced digraph G is one for which any given two paths P_1 and P_2 between a and b have the same algebraic length.

Definition 5.2. A balanced digraph is h-bipartite if all its paths are minimal and have algebraic length h.

(3) **In** $D_{\mathbb{S}}$**, intersections of minimal paths are not permitted.**

As long as $a \in V(D_{\mathbb{S}})$ is neither an initial vertex nor a terminal vertex to all minimal paths P_i in $D_{\mathbb{S}}$, a must solely belong to a unique P_i.

5.2 Relational structures and algebra

Given f and g, two functions, they form *a linear identity* if at most one function symbol appears on either side of the equation, $f(x, y, z) = g(x, z, x, x)$.

Definition 5.3. A linear identity is *balanced* if the same variables occur in f and g, $f(x, y, z, y) = g(x, y, y, y, z)$.

Definition 5.4. A linear identity f is *idempotent* if $f(x, x, x \ldots, x) = x$.

Definition 5.5. A given relational structure $\mathbb{S}$ is said to $satisfy$ a set of identities Σ if, when these identities are evaluated in S, they are all true.

Definition 5.6. A relational structure $\mathbb{T}$ is $primitive\ positive\ definable$ ($pp-definable$) by a relational structure $\mathbb{S}$, if the set T of $\mathbb{T}$ form a relation on $\mathbb{S}$ that is directly pp-definable from $\mathbb{S}$ and each relation of $\mathbb{T}$ defined on the set of T is pp-definable from $\mathbb{S}$.

We will show that every relational structure $\mathbb{S}$ is *pp-definable* by a balanced digraph $D_{\mathbb{S}}$ where $CSP(D_{\mathbb{S}})$ is polynomial time equivalent to CSP($\mathbb{S}$).

Definition 5.7. $Maltsev\ conditions$ are conjunctions of equations which polymorphisms may satisfy in a given relational structure.

For more details on important Maltsev conditions in universal algebra, see [11]. When Maltsev conditions are satisfied in a relational structure, it implies applicability of certain algorithms which solve the CSP of the relational structure. Let N denote the oriented path 101:

The path N satisfies most of the important Maltsev conditions relevant to the algebraic theory of CSPs and will later on be used in the reduction of CSP to a digraph.

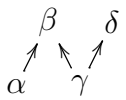

Figure 5.3: The oriented path N.

Definition 5.8. We say that a finite relational structure $\mathbb{S}$ is *congruence* 3-*permutable*, if it admits two ternary polymorphisms f and g such that

$$f(x, y, y) = x$$

$$f(x, x, y) = g(x, y, y)$$

$$g(x, x, y) = y,$$

for all $x, y \in S$.

Lemma 5.1. *(Bulin, Delić, Jackson and Niven, [5]) The path N admits a majority polymorphism, congruence 3-permutability polymorphisms, and it satisfies any balanced set of identities.*

Proof. The three polymorphisms listed in Lemma 5.2.1 imply most of the Maltsev conditions that have been studied in applications of universal algebra in the study of CSPs.

(1) N **admits a majority polymorphism.**

We can define the majority operation as described in the previous chapter to be the majority element of $\{x, y, z\}$ when at least two elements are equal. In the event that

$x \neq y \neq z$, we can choose to have $m(x, y, z) = \alpha$. The majority operation is a polymorphism because it preserves relations in N.

(2) N **admits congruence 3-permutability polymorphisms.**

Two linear polymorphisms f and g witnessing 3-permutability can be defined as follows:

$$f(x,y,z) = \begin{cases} \beta & \text{if } \beta \in \{x,y,z\} \text{ and } y \neq z; \\ \gamma & \text{if } \gamma \in \{x,y,z\} \text{ and } \beta \notin \{x,y,z\} \text{ and } y \neq z; \\ x & \text{otherwise.} \end{cases}$$

$$g(x,y,z) = \begin{cases} \beta & \text{if } \beta \in \{x,y,z\} \text{ and } x \neq y; \\ \gamma & \text{if } \gamma \in \{x,y,z\} \text{ and } \beta \notin \{x,y,z\} \text{ and } x \neq y; \\ z & \text{if } x = y; \\ x & \text{otherwise.} \end{cases}$$

If the tuple (x_1, y_1, z_1) is sent to (x_2, y_2, z_2) coordinate-wise, then in N, (x_1, y_1, z_1) must belong to $\{\alpha, \gamma\}$ and (x_2, y_2, z_2) consequently must belong to $\{\beta, \delta\}$. This satisfies the definition of f and g. Also $f(x_1, y_1, y_1) = x_1$ and $g(x_1, x_1, y_1) = y_1$ can be observed directly from the definition of f and g. In f when variables $y = z$, by definition the value assigned to the operation is x, in our case x_1. Then in g, when variables $x = y$, the value

assigned to the operation is z, in our case y_1.

The identity $f(x_1, x_1, y_1) = g(x_1, y_1, y_1) = x_1$ can also be proven by assuming $x_1 \neq y_1$. In f when $x = y$, the value of the operation is x, in our case x_1, and in g, when $y = z$, the value of the operation is x, in our case x_1. Hence, both functions f and g yield the same results x_1.

(3) N **satisfies any balanced set of identities.**

Let Σ denote the balanced set of identities. For any given operations applied to N from Σ, we can define them as the vertex that is closest to α. □

5.3 Reduction to digraphs

We have identified some prerequisites on relational structures and digraphs necessary for the reduction transformation. The process of the transformation is illustrated in Figure 5.4.

Figure 5.4: Process diagram for reduction to digraph.

Definition 5.9. A *bipartite relational structure* is a structure $\mathbb{S} = (S, R)$, where R is a binary relation, and $S = A \dot{\bigcup} B$ (S is a disjoint union of A and B), and $R \subseteq A \times B$.

5.3.1 Translation to a bipartite structure

In the bipartite structure translated from the relational structure the set of vertices are connected to the set of relations where they appear as coordinates. Here is how we can ensure this:

Let $\mathbb{S}$ be a relational structure, we define two components of $\mathbb{S}$, the set S and the set of relations $\overline{R_1}, \overline{R_2}, \ldots, \overline{R_n}$. Each of the $\overline{R_i}$ relations have arity k_i $(1 \leq i \leq n)$. We define the transversal relation $R = \overline{R_1} \times \overline{R_2} \times \cdots \times \overline{R_n}$.

Now let us define the edge relation R_i between the set S and the relations component R. In our bipartite structure, R_i will be defined as the edge connecting vertex $a \in S$ to the relation which has a as the ith coordinate of the tuple:

$$R_i = \{(x_i, (x_1, \ldots, x_m)) \mid (x_1, \ldots, x_m) \in R\}.$$

or using primitive positive formulas, we can define R:

$$R = \{(x_1, \ldots, x_k) \mid \exists y \ ((x_1, y) \in R_1 \wedge \cdots \wedge (x_k, y) \in R_k)\}.$$

Following the definition of pp-definable structures introduced at the beginning of this section, we can conclude that the relational structure $\mathbb{S}$ is pp-definable by the bipartite structure, denoted $B_{\mathbb{S}}$.

Theorem 5.1. *(Bulin, Delić, Jackson and Niven, [5]) Given a relational structure* $\mathbb{S}$ *and*

its corresponding bipartite structure $B_{\mathbb{S}}$. $\mathbb{S}$ satisfies a set of identities Σ, if and only if $B_{\mathbb{S}}$ satisfies Σ.

Proof. We refer the reader to [5] for the complete proof. □

5.3.2 Transformation to digraph

We will begin to transform our bipartite graph $B_{\mathbb{S}}$ to the desired balanced digraph $D_{\mathbb{S}}$. Let us replace every edge $e \in R_i$ by an oriented path P_e (of algebraic length $2k+1$) defined as

$$P_e = 1 + P_e^1 + 1 + P_e^2 + 1 + \cdots + 1 + P_e^k + 1,$$

$$P_e^i = \begin{cases} 1 & \text{if } e \in R_i \\ 101 & \text{else.} \end{cases}$$

where 101 is the path N, discussed earlier.

Example 5.1. Given the following tuples representing a relation from a bipartite structure $B_{\mathbb{S}}$,

(a) $(x_1, (x_1, y_1))$

(b) $(x_1, (y_1, x_1))$

we will build the corresponding P_e. Since $k = 2$, $P_e = 1 + P_e^1 + 1 + P_e^2 + 1$,

(a) Because $x_1 \in R_1$, $P_e^1 = 1$, P_e is

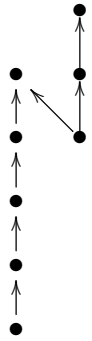

Figure 5.5: Path representing $(x_1, (x_1, y_1))$.

(b) Because $x_1 \in R_2$, $P_e^2 = 1$, P_e is

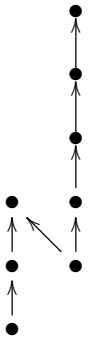

Figure 5.6: Path representing $(x_1, (y_1, x_1))$.

Note that the algebraic length of both paths in **(a)** and **(b)** is 5 (by definition $2k+1$).

After replacing every edge e by P_e, we obtain a $(2k+1)$-bipartite balanced digraph with minimal paths and

$$|V(D_{\mathbb{S}})| = |S| + |R| + 4|S||R|k - 2|R|k$$

and

$$|E(D_{\mathbb{S}})| = (|S||R|(4k+1) - 2|R|k$$

.

Lemma 5.2. *(Bulin, Delić, Jackson, Niven [5]) The relational structure* $\mathbb{S}$ *is pp-definable in* $\mathbb{D}_{\mathbb{S}}$ *on the set* S*.*

Proof. In [8], Häggkvist, Hell, Miller and Neumann established the existence of a minimal path of height h that is homomorphic to the set of minimal paths in the balanced digraph $D_{\mathbb{S}}$ (of height h). Let P_m denote such a minimal path that maps to all P_e. This means there is a trajectory P_m that connects the vertices in S to the relation component. Hence, the set S is pp-defined by $D_{\mathbb{S}}$:

$$S = \{\exists b \in D_{\mathbb{S}} | a \overset{P_m}{\to} b\}$$

□

Lemma 5.3. *(Bulin, Delić, Jackson, Niven [5]) CSP(*$D_{\mathbb{S}}$*) reduces in polynomial time to CSP(*$\mathbb{S}$*).*

Proof. We refer the reader to [5] for the complete proof. □

Chapter 6

Preservation of Bounded Width

The content of this section revolves around a new proof by Bulin, Delić, Jackson and Niven on the reduction of a finite relational structure to a balanced digraph. According to Theorem 4.5, the property of a CSP being of bounded width can be expressed by a conjunction of two balanced identities. Since the construction from the previous chapter preserves the validity of balanced identities, we have the following theorem:

Theorem 6.1. *(Bulin, Delić, Jackson and Niven, [5]) Given a finite relational structure* $\mathbb{S}$*=(S,R), where R is k-ary relation (k ≥ 2), CSP(*$\mathbb{S}$*) is of bounded width if and only if CSP(*$D_{\mathbb{S}}$*) is of bounded width.*

Proof. The complete proof can be found in [5]. It is largely algebraic and does not truly reflect the algorithmic reasons for such an equivalence. Namely, we would like to gain a better insight into how expressibility in Datalog gets preserved under this

construction. □

We now proceed to give an alternative proof of Theorem 6.1 working strictly within the Datalog-based definition of bounded width. Theorem 6.1 can be restated in the following way:

Theorem 6.2. *coCSP($\mathbb{S}$) is expressible in Datalog if and only if coCSP($D_{\mathbb{S}}$) is expressible in Datalog.*

Proof. It is a well-known fact that if a finite relational structure $\mathbb{S}$ is positive-primitive definable within a finite relational structure $\mathbb{T}$, then if coCSP ($\mathbb{T}$) is expressible in Datalog, so is coCSP($\mathbb{S}$) [7]. For that reason, we only need to show that if coCSP($\mathbb{S}$) is expressible in Datalog, so is coCSP($D_{\mathbb{S}}$). First, we need to establish two claims:

Claim 6.1. *Given $m \geq 1$, the set of all finite oriented paths whose algebraic length is greater or equal to $m+1$ is definable in Datalog.*

Proof. An oriented path $\vec{P}$ is of algebraic length greater or equal to $m+1$ if and only if $\vec{P} \not\to \vec{P}_m$ (no homomorphism exists between $\vec{P}$ and $\vec{P}_m$). Since $\vec{P}_m$ is of bounded width, for every m ≥ 1, the claim follows. □

Claim 6.2. *For every finite oriented path $\vec{P}$, coCSP($\vec{P}$) is definable in Datalog.*

Proof. This follows directly from the fact that every finite oriented path is of bounded width ([10]). This was proven in Chapter 4. □

Suppose $\mathbb{S} = (S, R)$ is a finite relational structure such that R is a k-ary ($k \geq 2$) relation, and assume that $coCSP(\mathbb{S})$ is definable in Datalog. Since the height of the associated balanced digraph $D_{\mathbb{S}}$ is $2k + 1$, we can construct the Datalog program which will accept coCSP($D_{\mathbb{S}}$) as follows:

(1) Include the clauses which will guarantee acceptance of any digraph which contains a path of algebraic length at least $2k + 2$. This can be done using the reasoning in Claim 6.1.

(2) For each path $\vec{P}_1, \ldots, \vec{P}_k$ used in the construction of $D_{\mathbb{S}}$, include clauses that will guarantee the acceptance of all $coCSP(\vec{P}_i)$, $i = 1, \ldots, k$. This can be done using Claim 6.2.

(3) For each clause of the program which accepts $coCSP(\mathbb{S})$ in

$$t :- t_0, t_1, \ldots, t_m$$

t_i is a relation that can be written as

$$R(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

for some elements $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, we replace it by a conjunction of the form

$$\exists y(\text{“}x_{i_1} \xrightarrow{P_1} y\text{”} \wedge \cdots \wedge \text{“}x_{i_k} \xrightarrow{P_k} y\text{”})$$

so that

$$\text{“}x_{i_j} \overset{P_j}{\rightarrow} y\text{”}$$

is a positive-primitive formula asserting that x_{i_j} and y are connected by an oriented path isomorphic to P_j. Finally, we claim that the Datalog program constructed in this way accepts a digraph G if and only if

$$G \not\rightarrow D_{\mathbb{S}}.$$

If G is accepted by clauses from (1), then it contains a path of algebraic length greater or equal to $2k + 2$, so it cannot be homomorphic to $D_{\mathbb{S}}$.

If G is accepted by clauses (2), then it is either such that all the paths in G are of algebraic length at most $2k$ with G not homomorphic to $D_{\mathbb{S}}$, or G contains a path of algebraic length at least 2k+2. In the latter case, the clauses from (3) will accept G if it is not homomorphic to $D_{\mathbb{S}}$.

Similarly, if $G \not\rightarrow D_{\mathbb{S}}$, then one of the sets of clauses (1)-(3) will guarantee the acceptance of G. □

Chapter 7

Conclusion and Open Problems

7.1 Summary

In Chapter 1, we gave basic definitions involving the concepts of homomorphisms on graphs and digraphs and established elementary consequences of the existence of a homomorphism between two graphs or digraphs. Chapter 2 presented basic notions from the theory of computational complexity, such as problems being in complexity classes **P** and **NP**. In addition, we found out that a well-known **NP**-complete problem, such as SAT, can be polynomially reduced to 3-SAT and 3-colouring problems for graphs.

In Chapter 3, we introduced the notion of the constraint satisfaction problem for a finite relational structure and investigated the role played by positive-primitive first-order formulas in the polynomial reduction of one CSP to another.

The concept of bounded width CSPs was investigated in Chapter 4, where we studied

several aspects in which it can be viewed using definability in Datalog, an algebraic problem asking whether a certain structure admits particular polymorphisms or, in the context of digraphs, whether particular intuitive algorithms are adequate to solve the CSP in question.

The original work in this thesis lies in a new method of reducing CSPs for relational structure to homomorphism problems for balanced digraphs, based on a recent work of Bulin, Delić, Jackson, and Niven. The exposition of Chapter 5 follows closely the content of [5].

Finally, Chapter 6 consists of original work, based on the techniques developed in the previous chapter, to give an original proof that CSPs of bounded width transform to homomorphism problems for digraphs which are also of bounded width. Unlike the result of Atserias ([1]) and those of Bulin, Delić, Jackson, and Niven ([5]), our proof does not rely on involved techniques from model theory or validity of algebraic conditions but simply using Datalog-based definitions.

7.2 Open problems

As stated in Chapter 3, the main open problem in the area is the so-called Dichotomy Conjecture, due to Feder and Vardi.

Conjecture 4. (Dichotomy Conjecture, [7]). Every CSP with a finite relational structure is either in **P** or it is **NP**-complete.

So far, all known evidence points to the affirmative answer to the conjecture. The two major results in the area ([2],[3]) indicate that the majority of CSPs over finite relational structures which are tractable seems to be so for two reasons:

1. They are of bounded width, or

2. They can be solved using a generalization of Gaussian elimination ("few subpowers algorithm" [3]). It is known that there are finite structures (see [7]) which are solvable using the few subpowers algorithm but are not of bounded width, and conversely.

In Chapter 5, we showed that if a relational structure $\mathbb{S}$ is such that coCSP($\mathbb{S}$) is expressible in Datalog, so is coCSP($D_{\mathbb{S}}$).

7.2.1 Datalog and LFP

The logic LFP (Least Fixed Point) is an extension of first-order logic that has an operator allowing to define least fixed points of a given formula. The LFP is formed by closing first-order logic under the rule: if μ is a formula, positive in the relational symbol R, then so is

$$lfp_{R,\vec{x}}\mu(\vec{t})$$

We interpret this to mean that the tuple $\vec{t}$ is the least fixed point of the operator that maps R to $\mu(R, \vec{x})$.

Example 7.1. The formula $\forall u \forall v [lfp_{T,xy}(x = y \vee \exists z(E(x,z) \wedge T(z,y)))](u,v)$ is satisfied in a graph $G = (V, E)$ if and only if G is connected.

This example shows that a Datalog program can be written as a single formula in the LFP logic. One can show that, similarly, every Datalog program can be written as a single LFP formula. However, there are examples (complicated) that show that the converse is not true: there are queries which can be expressed using formulas in the LFP logic, but which cannot be verified using Datalog programs. Thus, Datalog is a proper subset (in terms of what it can express) of the LFP logic. Hence, our result from Chapter 6 naturally leads to the following two problems:

Problem 1. If a finite relational structure $\mathbb{S}$ is such that coCSP($\mathbb{S}$) is definable in LFP, then is the same true for coCSP($D_{\mathbb{S}}$)?

Problem 2. Is there an extension of the logic LFP such that, for a finite relational structure $\mathbb{S}$, coCSP($\mathbb{S}$) is definable in this extension if and only if CSP($\mathbb{S}$) is in **P**?

We notice that if such an extension of LFP indeed exists and if given a finite structure $\mathbb{S}$ and a formula μ in this logic, we can decide whether

$$\mathbb{S} \models \mu.$$

That means if μ is true in $\mathbb{S}$ in runtime which is a polynomial function of $|\mathbb{S}|$, the

Dichotomy Conjecture is true. For that reason, we believe that further investigation of these problems may play an important role in settling the Dichotomy Conjecture and it will be the focus of our future research.

Bibliography

[1] A. Atserias, *On digraph coloring problems and treewidth duality,* European J. Combin. **29** (2008), 796–820.

[2] L. Barto and M. Kozik, *Constraint satisfaction problems of bounded width*, 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009).

[3] J. Berman, P. Idziak, P. Marković, R. McKenzie, M. Valeriote and R. Willard, *Varieties with few subalgebras of powers.* Trans. Amer. Math. Soc. **362** (2010), 1445–1473.

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

[5] J. Bulin, D. Delić, M. Jackson, T. Niven, *On the Reduction of the CSP Conjecture to Digraphs*, submitted to SIAM Journal of Computing.

[6] S. A. Cook, *The Complexity of Theorem Proving Procedures*, Proceedings Third Annual ACM Symposium on Theory of Computing, May 1971, 151- 158

[7] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory,* SIAM J. Comput., **28** (1), 1998, 57–104.

[8] R. Häggkvist, P. Hell, D.J. Miller and V. Neumann Lara, *On multiplicative graphs and the product conjecture,* Combinatorica **8** (1988), 63–74.

[9] P. Hell and J. Nešetřil, *On the complexity of H-colouring,* J. Combin. Theory Ser. B **48** (1990), 92–110.

[10] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford Lecture Series in Mathematics and its Applications 28, Oxford University Press, 2004.

[11] D. Hobby and R. McKenzie *The structure of finite algebras.* Contemporary Mathematics, 76. American Mathematical Society, Providence, RI, 1988. xii+203 pp. ISBN: 0-8218-5073-3.

[12] P. Idziak, P. Marković, R. McKenzie, M. Valeriote and R. Willard *Tractability and learnability arising from algebras with few subpowers,* SIAM J. Comput. 39 (2010), no. 7., 3023–3037.

[13] R. Ladner, *On the Structure of Polynomial Time Reducibility,* J. ACM **22** (1975), 155–171.

[14] B. Larose and P. Tesson *Universal algebra and hardness results for constraint satisfaction problems.* Theoret. Comput. Sci. 410 (2009), no. 18.

[15] B. Larose and L. Zadóri *Bounded width problems and algebras,* Algebra Universalis, 56 no. 3–4, 439–466, 2007.

[16] M. Maróti and R. McKenzie, *Existence theorems for weakly symmetric operations,* 2006. (to appear in Algebra Universalis)

[17] C. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.

[18] T. J. Schaefer, *The Complexity of Satisfiability Problems,* STOC (1978), pp. 216–226.

[19] D.B. West, *Introduction to Graph Theory*, 2nd edition, Prentice Hall, 2001.