Linköping Studies in Science and Technology

Dissertation No. 1091

Complexity Dichotomies for CSP-related Problems

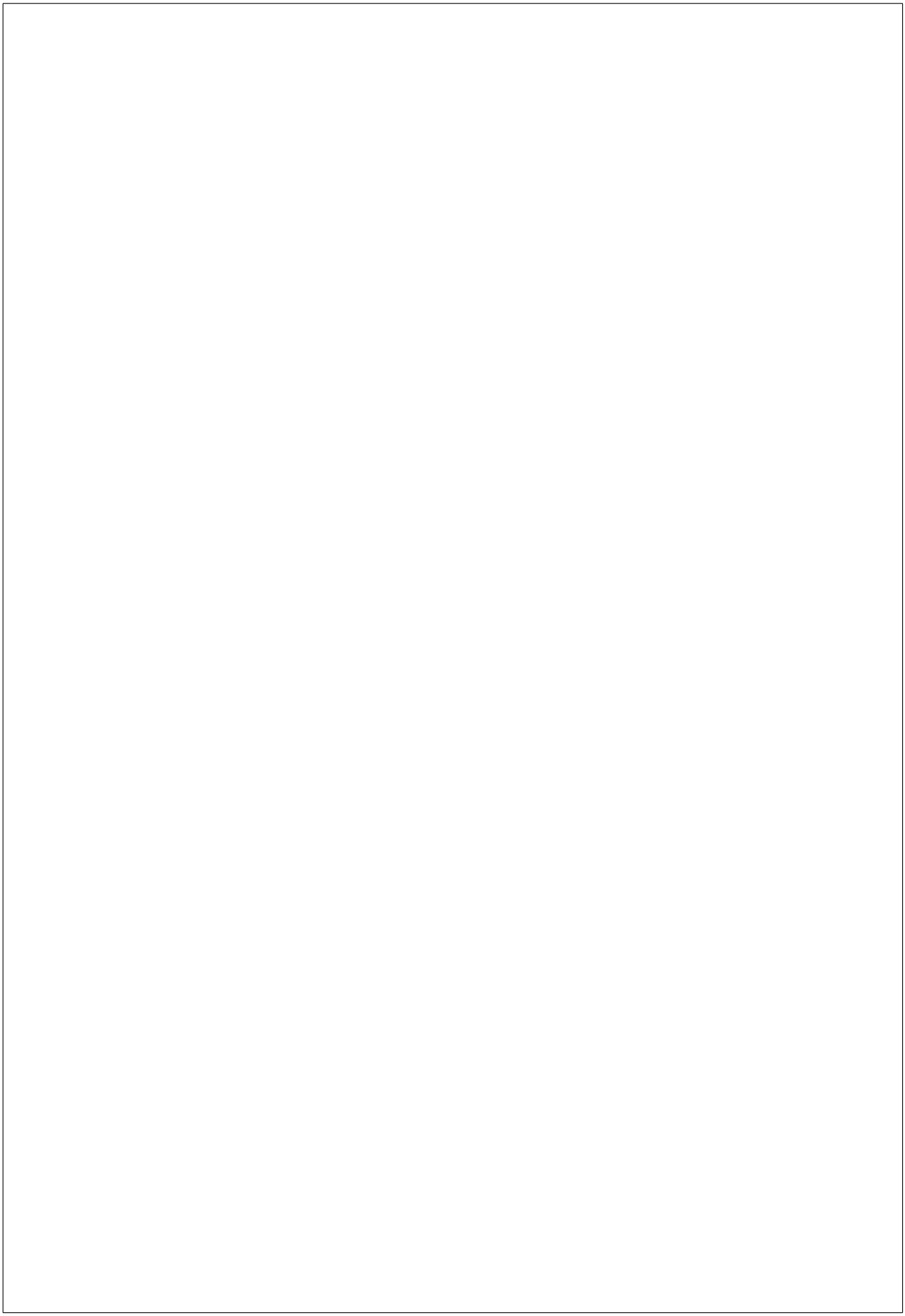
by

Gustav Nordh

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköping universitet
SE-581 83 Linköping, Sweden

Linköping 2007



Abstract

Ladner's theorem states that if $\mathbf{P} \neq \mathbf{NP}$, then there are problems in **NP** that are neither in **P** nor **NP**-complete. $\textsc{Csp}(\Gamma)$ is a class of problems containing many well-studied combinatorial problems in **NP**. $\textsc{Csp}(\Gamma)$ problems are of the form: given a set of variables constrained by a set of constraints from the set of allowed constraints Γ, is there an assignment to the variables satisfying all constraints? A famous, and in the light of Ladner's theorem, surprising conjecture states that there is a complexity dichotomy for $\textsc{Csp}(\Gamma)$; that is, for any fixed finite Γ, the $\textsc{Csp}(\Gamma)$ problem is either in **P** or **NP**-complete.

In this thesis we focus on problems expressible in the $\textsc{Csp}(\Gamma)$ framework with different computational goals, such as: counting the number of solutions, deciding whether two sets of constraints have the same set of solutions, deciding whether all minimal solutions of a set of constraints satisfies an additional constraint etc. By doing so, we capture a host of problems ranging from fundamental problems in nonmonotonic logics, such as abduction and circumscription, to problems regarding the equivalence of systems of linear equations. For several of these classes of problem, we are able to give complete complexity classifications and rule out the possibility of problems of intermediate complexity. For example, we prove that the inference problem in propositional variable circumscription, parameterized by the set of allowed constraints Γ, is either in **P**, **coNP**-complete, or $\mathbf{\Pi_2^P}$-complete. As a by-product of these classifications, new tractable cases and hardness results for well-studied problems are discovered.

The techniques we use to obtain these complexity classifications are to a large extent based on connections between algebraic clone theory and the complexity of $\textsc{Csp}(\Gamma)$. We are able to extend these powerful algebraic techniques to several of the problems studied in this thesis. Hence, this thesis also contributes to the understanding of when these algebraic techniques are applicable and not.

Acknowledgements

I would like to thank all my colleagues at IDA, especially the members of TCSLab, for providing a stimulating research atmosphere. My main supervisor has been Peter Jonsson. It has been a true privilege to collaborate with him and I would like to thank him, in particular, for his enthusiasm and for *always* having time to discuss research matters. The truth is that these discussions have been the most rewarding and enjoyable part of my work.

There are many more who deserve to be thanked. Out of these, Svante Linusson and Andrzej Szalas have been my secondary supervisors, Bruno Zanuttini has co-authored one of the papers in this thesis, Miki Hermann has shown me great hospitality when visiting him in Paris, and Daniel and Erika have given me night shelter and reduced my travelling needs these last hectic weeks.

I would also like to take the opportunity to thank my parents Annika and Roland for all their love and support, and last but not least, Lisa for constantly reminding me that there is more to life than research.

This research work was funded in part by CUGS (the National Graduate School in Computer Science, Sweden). Although, I guess that those who really paid for this, and deserve the biggest thank, is the Swedish people who through their tax-money have paid my salary and travel expenses. *It is my sincere hope that you will be satisfied with what you have paid for.*

Gustav Nordh

Årnäs, Sweden, April 2007

List of Papers

This thesis includes the following five papers:

I. Gustav Nordh and Peter Jonsson. The Complexity of Counting Solutions to Systems of Equations over Finite Semigroups. In *Proceedings of the 10th Annual International Conference on Computing and Combinatorics (COCOON-2004)*, pp. 370-379, Jeju Island, Korea, August, 2004.

II. Gustav Nordh. The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups. *Theoretical Computer Science* 345(2-3): 406-424, 2005.

This article is an extended version of the paper:

> Gustav Nordh. The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups. In *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS-2004)*, pp. 380-391, Prague, Czech Republic, August, 2004.

III. Gustav Nordh and Peter Jonsson. An Algebraic Approach to the Complexity of Propositional Circumscription. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS-2004)*, pp. 367-376, Turku, Finland, July, 2004.

IV. Gustav Nordh. A Trichotomy in the Complexity of Propositional Circumscription. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-2004)*, pp. 257-269, Montevideo, Uruguay, March, 2005.

V. Gustav Nordh and Bruno Zanuttini. Propositional Abduction is Almost Always Hard. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pp. 534-539, Edinburgh, Scotland, UK, August, 2005.

Contents

Introduction

The research area Theoretical Computer Science can be defined as follows[1]: *Theoretical Computer Science (TCS) studies the inherent powers and limitations of computation, that is broadly defined to include both current and future, man-made and naturally arising computing phenomena.*

To be slightly more concrete and to get a feeling for what is meant, let us take an example. Suppose I ask you for help and give you two numbers, say 134578 and 426339, and then ask you to *either* compute their sum $134578 + 426339$ or their product $134578 \cdot 426339$ (the choice is yours, and no, you are not allowed to use a calculator). What would you choose?

My guess is that you would choose to compute the sum because you are lazy and probably find additions easier to compute than multiplications. Anyway, let us assume that you find doing additions of large numbers (by hand) easier than doing multiplications of large numbers (by hand). Have you ever thought about why?

Is it the case that there is some inherent property of the computational problem of multiplication that makes it harder than the problem of addition? How do you know that there is no (yet to be discovered) method of doing multiplication as easily as addition?

The subfield of TCS that studies these types of questions is called *complexity theory*. One of the most important goals in this

[1]Actually, it is defined exactly like this at `www.theorymatters.org`.

research area is to understand which computational problems are easy to solve and which are hard to solve, and of course, why. In this thesis we use the framework of complexity theory to try to classify large classes of computational problems according to the effort (actually, time) required for solving them.

Intuitions about Complexity Theory

In this section we try to give some intuitions about complexity theory. For a more formal treatment, please consult a standard textbook on complexity theory such as [43].

Suppose we want to solve the following task: given a map with n countries, the task is to decide whether or not the countries can be colored by two colors such that no two countries sharing a border are colored by the same color. We want our solution procedure to have the following desirable property: as the number n of countries is increased, the time required for solving the problem should not grow dramatically. More exactly, the time required for solving the problem should not grow faster than a polynomial in n. If such a solution procedure exists we say that the problem is tractable (or solvable in polynomial time). The class of all decision problems having such polynomial time solution procedures is exactly the complexity class **P**.

Here is a solution procedure for the problem of deciding whether a map can be properly colored using two colors. Color one country (which one does not matter). Then color its neighbors with the other color. Color *their* (uncolored) neighbors with the first color, and so on until you have either finished coloring the whole map (demonstrating that the map can be colored by two colors), or reach a position where two neighbouring countries are colored by the same color (in which case the map cannot be properly colored by just using two colors).

By making the reasonable assumption that the time required for carrying out this solution procedure corresponds to the number of times you have to color a country, it is obvious that the

time required to carry out the procedure does not grow faster than a polynomial in n. Hence, the problem of deciding whether or not a map can be colored by two colors is in **P**.

Now, let us alter the map coloring problem slightly by instead asking if a given map can be properly colored by three colors. As it turns out, no one has been able to give a solution procedure running in polynomial time for this problem. In fact, it is strongly *believed* that the problem of properly coloring a map by three colors is intractable (computationally hard), that is, not solvable in polynomial time. The main embarrassment of computational complexity theory is that it is incredibly bad at *proving* that problems are computationally hard, e.g., not solvable in polynomial time. The main reason is that, to prove that a problem is computationally hard, *all possible* efficient solutions procedures must be ruled out.

Again, consider the problem of deciding if a map can be properly colored with three colors; how do you rule out the possibility that in the future, someone/something, somewhere, comes up with a polynomial time solution procedure for solving the problem? Incidentally, the problem of deciding if a map can be properly colored with four colors can be solved in polynomial time as a result of the famous Four Color Theorem[2].

The reason why the problem of deciding if a map can be properly colored with three colors is believed to be hard, is that it is among the hardest problems in the complexity class **NP**. A problem is in the complexity class **NP** if the correctness of solutions can be verified in polynomial time. For example, given a colored map, it is easy to check in polynomial time that no more than three colors are used and that neighbouring countries are colored with different colors. Hence, the problem of deciding if a map can be properly colored with three colors is in **NP**.

Obviously, if a problem can be solved in polynomial time, then solutions can also be verified in polynomial time. Hence,

[2]Every map can be properly colored using four colors [3].

every problem in the complexity class **P** is also in **NP**. Although it is strongly believed that $\mathbf{P} \neq \mathbf{NP}$, no one has been able to prove this (despite the fact that a proof carries a monetary award of $1000 000) and it is currently one of the most important open problems in all of mathematics[3]. The hardest problems in **NP** are the **NP**-complete problems. The **NP**-complete problems all have the interesting property that if any single one of them is solvable in polynomial time, then all of the problems in **NP** are solvable in polynomial time (and, hence, $\mathbf{P} = \mathbf{NP}$). So, the **NP**-complete problems are the hardest problems in **NP** in the sense that they are the ones most unlikely to be solvable in polynomial time.

In addition to **P** and **NP**, we encounter several other complexity classes in this thesis. The most important ones are: **FP**, **#P**, **coNP**, $\mathbf{\Sigma_2^P}$, and $\mathbf{\Pi_2^P}$. Very briefly, **FP** is the class of all problems solvable in polynomial time where the answer is a natural number, and **#P** is the class of all problems where the goal is to count the number of solutions to a problem in **NP**. For example, the problem of adding two numbers is in **FP** and the problem of counting the number of proper three colorings of a map is in **#P** (but probably not in **FP**). **coNP** is the class of problems for which nonexistence of solutions can be proved by polynomial time verifiable proofs. Hence, the problem of deciding whether a map has a proper three coloring would[4] be in **coNP** if, given a map that cannot be properly colored by three colors, there were some way to give a polynomial time verifiable proof of this. $\mathbf{\Sigma_2^P}$ is the class of problems for which correctness of solutions can be verified in polynomial time with the help of an oracle that can solve problems in **NP** instantly. Similarly, $\mathbf{\Pi_2^P}$ is the class of problems for which nonexistence of solutions can

[3]One of the six most important open problems in mathematics according to the Clay Mathematics Institute's list of Millennium Problems: `www.claymath.org/millennium/`.

[4]It is unlikely that this problem is in **coNP** since it would imply that $\mathbf{NP} \subseteq \mathbf{coNP}$.

be proved by proofs that are verifiable in polynomial time with the help of an oracle that can solve problems in **NP** instantly. The reader is referred to [43] for formal definitions and further information about these classes.

The following inclusions among these complexity classes are obvious from their definitions.

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{\Sigma_2^P}, \;\; \mathbf{P} \subseteq \mathbf{coNP} \subseteq \mathbf{\Pi_2^P}$$

Throughout this thesis, *we assume that all the inclusions above are strict*, e.g., that $\mathbf{P} \neq \mathbf{NP} \neq \mathbf{\Sigma_2^P}$. Moreover, we assume that $\mathbf{FP} \neq \mathbf{\#P}$. These assumptions are all non-controversial and widely accepted [43]. For example, it seems unlikely that any physically realizable computational machine (including quantum computers) can solve **NP**-complete problems in polynomial time. It has even been argued that the impossibility to solve **NP**-complete problems in polynomial time in our physical universe is a natural law as profound as the impossibility of superluminal signalling [1].

Again, let us concentrate on the perhaps two most important complexity classes, **P** and **NP**. Given a natural computational problem of your choice in **NP**, it is very likely that your problem is already classified as being either in **P** or **NP**-complete [26] (or that such a classification is an easy exercise[5]). Indeed, one of the greatest success stories in all of TCS is the realization that almost all natural computational problems can be classified (under reasonable assumptions, such as **P**≠**NP**) as either being tractable or hard to solve. Somehow this does not fit in with our (or at least mine) intuitive (pre-complexity theory) experience of the difficulty of solving problems. Problems are not just easy or hard, but instead we intuitively expect that there are problems of intermediate complexity. Indeed, Ladner's theorem [39]

[5]There are exceptional natural problems such as Graph Isomorphism, which are not yet classified and which might be of intermediate complexity, but they are extremely rare.

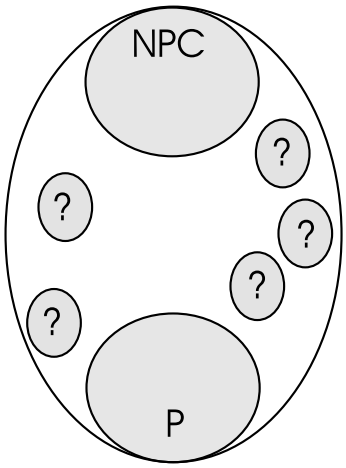


Figure 1: The internal structure of **NP** according to Ladner's theorem (NPC is just an abbreviation of **NP**-complete).

tells us that if $\mathbf{P} \neq \mathbf{NP}$, then there is an infinite number of classes of problems in **NP** that are neither **NP**-complete nor in **P**. See Figure 1 for a simplified pictorial description of the internal structure of the complexity class **NP**, under the assumption that $\mathbf{P} \neq \mathbf{NP}$.

This apparent lack of natural computational problems of intermediate complexity is something that we do not yet fully understand. For example, which classes of problems in **NP** have a dichotomy between **P** and **NP**-complete, i.e., do not contain problems of intermediate complexity?

Parameterized Problems

One way to investigate such dichotomy questions is to take a problem and parameterize it in some natural way and study what happens to the complexity of the problem when the parameter is tweaked, e.g., can the parameter be tuned-in so that the resulting

problem is of intermediate complexity? For example, consider the problem of coloring the vertices of a graph with k colors such that no two adjacent vertices receive the same color, i.e., the k-coloring problem[6]. This problem is in **P** when $k \leq 2$ and **NP**-complete for $k \geq 3$. Hence, in this case we could not tweak the parameter to obtain a problem of intermediate complexity (which is not too surprising considering the restricted form of the parameter). The k-coloring problem is a particular case of the more general H-coloring problem, that is, given a graph G determine whether or not there is a homomorphism from G to H (i.e., an edge preserving map from the vertices of G to the vertices of H). Note that a graph G can be k-colored if and only if there is a homomorphism from G to the complete irreflexive graph on k-vertices. Hell and Nešetřil [29] proved that the H-coloring problem is in **P** if H is bipartite or if H contains a looped vertex and that it is **NP**-complete otherwise. So, despite the fact that the parameter H in the H-coloring problem is much more sensitive than the k parameter in the k-coloring problem, it is still not possible to tweak the parameter H to obtain a problem of intermediate complexity.

For another example of a parameterized version of an **NP**-complete problem that on the surface looks very different from the H-coloring problem, consider the satisfiability problem for CNF formulas parameterized by the set of allowed clauses. The CNF-SAT(Γ) problem in propositional logic is the problem of deciding whether or not a conjunction of clauses, all of which must be of the types of allowed clauses specified by Γ, have a variable assignment satisfying all the clauses. For example, the 2-SAT problem is the CNF-SAT(Γ) problem where

$$\Gamma = \{(x \vee y), (\neg x \vee y), (\neg x \vee \neg y)\}$$

[6]The map coloring problem mentioned earlier is just the k-coloring problem on planar graphs.

and the 3-SAT problem is CNF-SAT(Γ) where

$$\Gamma = \{(x \vee y \vee z), (\neg x \vee y \vee z), (\neg x \vee \neg y \vee z), (\neg x \vee \neg y \vee \neg z)\}.$$

We emphasize that Γ contains the set of allowed types of clauses, i.e., $\Gamma = \{(x \vee y), (\neg x \vee y), (\neg x \vee \neg y)\}$, corresponding to the 2-SAT problem, specifies that instances of the CNF-SAT(Γ) problem consists conjunctions of clauses each having two (possibly negated) variables. Hence,

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_1) \wedge (x_2 \vee x_3),$$

where x_1, x_2, x_3 are propositional variables is an instance of the CNF-SAT(Γ) problem where $\Gamma = \{(x \vee y), (\neg x \vee y), (\neg x \vee \neg y)\}$.

As a consequence of a more general result due to Schaefer [47] (which we come back to in the next section), it follows that also the CNF-SAT(Γ) class of problems is either in **P** or **NP**-complete (depending on the set of allowed clauses Γ), but never of intermediate complexity. Both the CNF-SAT(Γ) class of problems and the H-coloring class of problems are special cases of a more general class of problems, namely, Constraint Satisfaction Problems (CSPs).

Constraint Satisfaction Problems

Constraint satisfaction problems have a long and rich history in computer science where they are extensively used to represent problems of a combinatorial nature [20, 50]. An instance of a constraint satisfaction problem consists of a set of variables, a set of possible values for the variables, and a set of constraints that restrict the combination of values that certain tuples of variables may take. The goal is to decide whether there is an assignment of values to the variables such that the given constraints are satisfied.

Due to the importance the CSP problem and the fact that it is an **NP**-complete problem, there has been much research

devoted to finding restricted cases of the CSP problem that are tractable.

One of the dominant and most natural approaches for finding such islands of tractability is to restrict the types of allowed constraints [17]. The set of (types of) allowed constraints is called the *constraint language*. We now formally define the CSP problem parameterized by the set of allowed constraints. The set of all n-tuples of elements from a domain D is denoted by D^n. Any subset of D^n is called an n-ary relation on D. The set of all finitary relations over D is denoted by R_D. A constraint language over a finite set, D, is a finite set $\Gamma \subseteq R_D$. The constraint satisfaction problem over the constraint language Γ, denoted $\textsc{Csp}(\Gamma)$, is defined to be the decision problem with instance (V, D, C), where

- V is a set of variables,
- D is a finite[7] set of values (sometimes called the domain), and
- C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, R_i) where s_i is a list of variables from V of length m_i, called the constraint scope, and R_i is an m_i-ary relation over the set D, belonging to Γ, called the constraint relation.

The question is whether there exists a solution to (V, D, C) or not, that is, a function from V to D such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

Example 1 Let $\neq_D$ denote the binary relation

$$\{(a, b) \mid a, b \in D \text{ and } a \neq b\}.$$

[7]CSPs over infinite domains are also common, but in this thesis the domain will always be a finite set.

Then the $\text{CSP}(\neq_D)$ problem is exactly the $|D|$-coloring problem. To see this, given an instance of the $|D|$-coloring problem (i.e., a graph $G = (V, E)$), then the corresponding $\text{CSP}(\neq_D)$ instance I consists of the constraints $C = \{((v_i, v_j), \neq_D) \mid (v_i, v_j) \in E\}$. That is, two variables v_i, v_j are constrained by a $\neq_D$ constraint if and only if they are adjacent in the graph. It is easy to see that the $\text{CSP}(\neq_D)$ instance I has a solution if and only if the graph G can be $|D|$-colored.

More generally, given a graph H (viewed as a binary symmetric relation), then the $\text{CSP}(H)$ problem is exactly the H-coloring problem. Hence, Hell and Nešetřil's [29] complexity classification of the H-coloring problem characterize the complexity of the $\text{CSP}(\Gamma)$ problem when Γ consists of a single binary symmetric relation.

Now, the research problem that manifests itself is: for which constraint languages Γ is the $\text{CSP}(\Gamma)$ problem tractable and for which constraint languages is it **NP**-complete?

Schaefer proved already in 1978 the following remarkable complexity classification of $\text{CSP}(\Gamma)$ when Γ is constraint language over a two-element domain.

Theorem 2 ([47]) *Let Γ be a finite constraint language over the domain $\{0, 1\}$. The $\text{CSP}(\Gamma)$ problem is in* **P** *if Γ satisfies one of the six conditions below; otherwise $\text{CSP}(\Gamma)$ is* **NP***-complete.*

- *Every relation in Γ contains the tuple $(0, 0, \ldots, 0)$ (also referred to as 0-valid).*
- *Every relation in Γ contains the tuple $(1, 1, \ldots, 1)$ (also referred to as 1-valid).*
- *Every relation in Γ can be expressed by a CNF formula where each conjunct has at most one unnegated variable (also referred to as a Horn formula).*

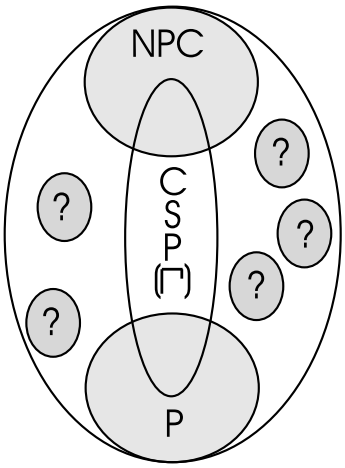


Figure 2: The relationship between **NP** and Csp(Γ) if the Feder-Vardi dichotomy conjecture holds.

- *Every relation in* Γ *can be expressed by a CNF formula where each conjunct has at most one negated variable (also referred to as a dual-Horn formula).*
- *Every relation in* Γ *can be expressed by a CNF formula where each conjunct has at most two variables (also referred to as a bijunctive formula).*
- *Every relation in* Γ *can be expressed by a conjunction of linear equations over the two element group* $\mathbb{Z}_2$ *(also referred to as an affine formula).*

Feder and Vardi [25] conjectured in their famous *dichotomy conjecture* that Csp(Γ) is always either in **P** or **NP**-complete. Moreover, they proved that, in some sense, the Csp(Γ) class of problems might be the largest class of problems in **NP** for which there are no problems of intermediate complexity. Due to the practical as well as theoretical importance of the Csp(Γ) class of problems, the Feder-Vardi dichotomy conjecture has received

a great deal of attention. Several different complementary approaches for attacking the conjecture have been proposed. The original approach, due to Feder and Vardi [25], is based on logical methods from descriptive complexity theory. For more information on this approach, we refer the reader to [37]. It is fair to say that the currently most promising approach towards the conjecture is based on results and methods from universal algebra [12], which we briefly discuss in the next section.

Algebraic Approach

The basis of the algebraic approach to the complexity of $\textsc{Csp}(\Gamma)$ stems from the realization that the complexity of $\textsc{Csp}(\Gamma)$ is completely determined by the presence or absence of certain closure operations on Γ.

An *operation* on a finite set D (the domain) is an arbitrary function $f : D^k \to D$. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows: Let f be a k-ary operation on D and let R be an n-ary relation over D. For any collection of k tuples, $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R$, the n-tuple $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k})$ is defined as follows:

$$f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) =$$

$$(f(\boldsymbol{t_1}[1], \boldsymbol{t_2}[1], \ldots, \boldsymbol{t_k}[1]), \ldots, f(\boldsymbol{t_1}[n], \boldsymbol{t_2}[n], \ldots, \boldsymbol{t_k}[n])),$$

where $\boldsymbol{t_j}[i]$ is the i-th component in tuple $\boldsymbol{t_j}$.

Now, let $R \in \Gamma$. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R$ we have $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R$, then R is *invariant* (or, in other words, closed) under f. If all relations in Γ are invariant under f then Γ is invariant under f. An operation f such that Γ is invariant under f is called a *polymorphism* of Γ. The set of all polymorphisms of Γ is denoted $Pol(\Gamma)$. Given a set of operations F, the set of all relations that are invariant under all the operations in F is denoted $Inv(F)$.

Example 3 A *majority* operation f is a ternary operation satisfying $f(a,a,b) = f(a,b,a) = f(b,a,a) = a$ for all $a, b \in D$; Let $D = \{0,1,2\}$ and let f be the majority operation on D where $f(a,b,c) = a$ if a, b and c are all distinct. Furthermore, let

$$R = \{(0,0,1),(1,0,0),(2,1,1),(2,0,1),(1,0,1)\}.$$

It is then easy to verify that for every triple of tuples, $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in R$, we have $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in R$. For example, if $\boldsymbol{x} = (0,0,1), \boldsymbol{y} = (2,1,1)$ and $\boldsymbol{z} = (1,0,1)$ then

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) =$$
$$\big((f(\boldsymbol{x}[1], \boldsymbol{y}[1], \boldsymbol{z}[1]), f(\boldsymbol{x}[2], \boldsymbol{y}[2], \boldsymbol{z}[2]), f(\boldsymbol{x}[3], \boldsymbol{y}[3], \boldsymbol{z}[3])\big) =$$
$$\big(f(0,2,1), f(0,1,0), f(1,1,1)\big) = (0,0,1) \in R.$$

We can conclude that R is invariant under f or, equivalently, that f is a polymorphism of R.

We continue by defining a closure operation $\langle\cdot\rangle$ on sets of relations: for any set $\Gamma \subseteq R_D$ the set $\langle\Gamma\rangle$ consists of all relations that can be expressed using relations from $\Gamma \cup \{=_D\}$ ($=_D$ is the identity relation on D), conjunction, and existential quantification (see Example 4 below). Intuitively, constraints using relations from $\langle\Gamma\rangle$ are exactly those which can be simulated by constraints using relations from Γ.

Example 4 Let

$$R = \{(0,0,1),(1,0,0),(2,1,1),(2,0,1),(1,0,1)\}.$$

Then the relation

$$S = \{(1,0,0),(1,0,1),(2,1,1)\} \equiv \exists z(((x,y,y),R) \wedge ((x,z,w),R))$$

is in $\langle R\rangle$.

The sets of relations of the form $\langle\Gamma\rangle$ are referred to as *relational clones*. An alternative characterisation of relational clones is given by the following theorem.

Theorem 5 (See, [46]) *For every set* $\Gamma \subseteq R_D$,

$$\langle\Gamma\rangle = Inv(Pol(\Gamma)).$$

The next theorem states that when studying the complexity of $\textsc{Csp}(\Gamma)$, it is sufficient to consider constraint languages that are relational clones.

Theorem 6 ([30]) *Let* Γ *be a constraint language and* $\Gamma' \subseteq \langle\Gamma\rangle$ *finite. Then* $\textsc{Csp}(\Gamma')$ *is polynomial-time reducible to* $\textsc{Csp}(\Gamma)$.

As an easy consequence of the preceding results, if $Pol(\Gamma) \subseteq Pol(\Gamma')$ for a finite constraint language Γ', then $\textsc{Csp}(\Gamma')$ is polynomial-time reducible to $\textsc{Csp}(\Gamma)$. Just note that the fact that $Pol(\Gamma) \subseteq Pol(\Gamma')$ implies that $Inv(Pol(\Gamma')) \subseteq Inv(Pol(\Gamma))$ and, as a consequence of Theorem 5, $\Gamma' \subseteq \langle\Gamma\rangle$ and the reduction follows from Theorem 6. Hence, the complexity of the $\textsc{Csp}(\Gamma)$ problem is completely determined by the polymorphisms of Γ, i.e., $Pol(\Gamma)$. Hence, complexity results for $\textsc{Csp}(\Gamma)$ can be conveniently stated in terms of the operations in $Pol(\Gamma)$. For example, Jeavons et al. [31] prove that if $Pol(\Gamma)$ contains a majority operation (as defined in Example 3), then $\textsc{Csp}(\Gamma)$ is in **P**.

It is well known that the set of all relational clones over a set D form a lattice under set inclusion. Post [45] classified all relational clones over the Boolean domain. The lattice of all Boolean relational clones is usually referred to as Post's lattice and can be found in Figure 3. This lattice is an effective tool for classifying the complexity of $\textsc{Csp}(\Gamma)$-like problems over Boolean constraint languages, see for example Papers IV and V in this thesis. For more information on the Boolean relational clones and Post's lattice we refer the reader to the survey articles [4, 5].

We remark that Schaefer did not use this lattice in the proof of his dichotomy theorem for $\textsc{Csp}(\Gamma)$ over Boolean constraint languages. In fact, it is easy to give a very short proof of Schaefer's dichotomy theorem using Post's lattice. To illustrate the power of the algebraic approach we sketch such a proof.

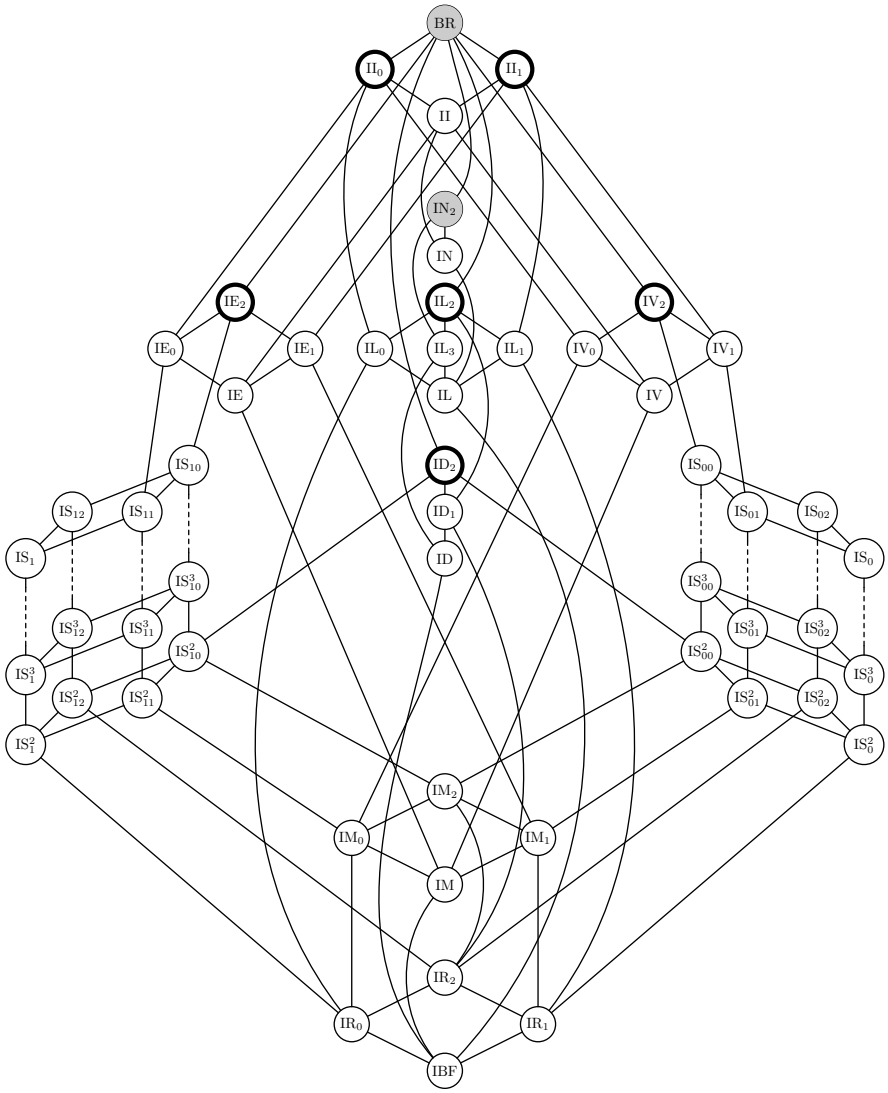


Figure 3: Post's lattice of relational clones.

The six tractable cases in Schaefer's dichotomy theorem are (in Schaefer's own words) all either trivial or previously known. We have indicated the relational clones corresponding to these six tractable cases by making them bold in the lattice. Hence, $\textsc{Csp}(\Gamma)$ is in **P** when $\langle\Gamma\rangle$ is one of these six relational clones, or a relational clone lying below one of these in the lattice. Only two relational clones remain to be classified, namely, BR and IN_2 (colored grey in the lattice). The relational clone IN_2 is exactly $Inv(f)$ where f is the unary operation, $f(0) = 1$, $f(1) = 0$. Let $NAE3$ be the following ternary relation on $\{0, 1\}$:

$$NAE3 = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

It is easy to verify that $NAE3$ is invariant under f (i.e., $NAE3 \in IN_2$) and it is an easy exercise to prove that $\textsc{Csp}(NAE3)$ is **NP**-complete. As a consequence, $\textsc{Csp}(BR)$ and $\textsc{Csp}(IN_2)$ are **NP**-complete, and we are done.

The main technical difficulty (and contribution) in Schaefer's original proof was the result that $\textsc{Csp}(\Gamma)$ is **NP**-complete for *any* Γ which does not fall into one of the six tractable cases. Using the algebraic approach via Post's lattice, we get this result essentially for free.

Unfortunately, the lattices of relational clones over domains of cardinality three or more are much more complicated and their structure is far from well understood. Never the less, Bulatov managed to prove a dichotomy (between **P** and **NP**-complete) for the complexity of $\textsc{Csp}(\Gamma)$ for constraint languages Γ over three-element domains by making heavy use of the algebraic approach via polymorphisms and relational clones [9]. Moreover, Bulatov et al. [12] have presented a conjecture for the complexity of $\textsc{Csp}(\Gamma)$ that exactly describes the borderline between tractability and **NP**-completeness (and hence refines the Feder-Vardi dichotomy conjecture). The hardness part of the conjecture is known to hold [30] and only the tractability part is left to prove. More specifically, to prove the conjecture (and, hence, also

the Feder-Vardi dichotomy conjecture) it would be sufficient [40] to prove that Csp(Γ) is in **P** if there is some k-ary weak near-unanimity (k-WNU) operation f in $Pol(\Gamma)$, where a k-WNU operation is a k-ary operation f that satisfies the identities

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

and

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

Contributions

In this thesis we broaden, in some sense, the Csp(Γ) framework by allowing more general computational goals than just deciding whether or not the given constraints have a solution. The computational goals we study are in turn:

- #Csp(Γ): Count the number of solutions to a Csp(Γ) instance.
- Equiv-Csp(Γ): Decide whether two Csp(Γ) instances are equivalent, i.e., whether they have the same set of solutions.
- Iso-Csp(Γ): Decide whether two Csp(Γ) instances are isomorphic, i.e., whether they can be made equivalent by permuting the variables in one of them.

The next two goals rely on a rather special notion of minimal solutions of Csp(Γ) instances which we need to define. Assume that the domain D is a partial order $(D, \leq)$. Given a Csp(Γ) instance I and a partition of the variables into three sets $(P; Z; Q)$. Then, a solution α to I is a minimal solution if and only if there is no other solution β such that $\alpha(x) = \beta(x)$ for all $x \in Q$, $\beta(x) \leq \alpha(x)$ for all $x \in P$, and $\alpha(x) \neq \beta(x)$ for at least one $x \in P$.

- MIN-CSP(Γ): Decide whether a variable assignment is a minimal solution of a CSP(Γ) instance, i.e., the model checking problem in propositional circumscription.
- MIN-INF-CSP(Γ): Decide whether every minimal solution of a CSP(Γ) instance also is a solution of an additional constraint, i.e., the inference problem in propositional circumscription.
- ABDUCTION(Γ): Given sets of literals M and H and a CSP(Γ) instance with constraints C, decide whether there is a set of literals $E \subseteq H$ such that $C \wedge \bigwedge E$ is satisfiable and $C \wedge \bigwedge E \models \bigwedge M$. This is the problem of deciding whether an explanation exists in propositional logic-based abduction.

As can be seen from the list above we have not tried to come up with new exotic computational goals but instead focused on well-known computational problems that can be recast as CSP(Γ) problems with different computational goals. Also note that the parameterization in terms of constraint languages chosen here is not very controversial. Indeed, the complexity of all these problems have been studied before under these parameterizations, cf. [6, 7, 10, 15, 18, 23, 33, 34, 48].

Applicability of the Algebraic Approach

Taking into account that the algebraic approach for investigating the complexity of CSP(Γ) has proved to be so spectacularly successful we would like to know the range of applicability of these powerful methods. In particular, a relevant question is: for which of the problems above is the algebraic approach applicable? To be a bit more specific: For which of the computational goals above is it the case that the complexity is completely determined by the set of polymorphisms of Γ (i.e., $Pol(\Gamma)$), or equivalently,

for which computational goals is it the case that the complexity of the problem restricted to Γ and Γ' is the same whenever $\langle\Gamma\rangle = \langle\Gamma'\rangle$?

For the #Csp(Γ) problem this has already been proved by Bulatov and Dalmau [10]. The situation for Equiv-Csp(Γ) and Iso-Csp(Γ) is less clear. The complexity of Equiv-Csp(Γ) and Iso-Csp(Γ) has been classified for all Boolean constraint languages Γ by Böhler et al. [6, 7]. By inspecting their results it is easy to see that for finite Boolean constraint languages Γ and Γ' it is the case that Equiv-Csp(Γ) and Equiv-Csp(Γ) have the same complexity. The analogous result hold for Iso-Csp(Γ). For larger domains the problem is still open.

For Min-Csp(Γ) and Min-Inf-Csp(Γ) we show (in Paper III) that if Γ' is a finite subset of $\langle\Gamma\rangle$, then Min-Csp(Γ') (Min-Inf-Csp(Γ')) is polynomial-time reducible to Min-Csp(Γ) (Min-Inf-Csp(Γ)). Hence, the algebraic approach is applicable to these problems.

Similarly, we show (in Paper V) that Abduction(Γ') is polynomial-time reducible to Abduction(Γ) when Γ' is a finite subset of $\langle\Gamma\rangle$. Thus, the algebraic approach is applicable also to the Abduction(Γ) problem.

Complexity Dichotomies

We have so far emphasized the theoretical importance of complexity classifications, but of course they can also be of great practical importance. For example, the identification of new islands of tractability for an important (suitable parameterized) problem can lead to better solution procedures for this problem. By choosing the parameterization of the problem carefully, it is often possible to more systematically study special cases of the problem which has previously been studied in the literature. Taking the Csp(Γ) problem as an example, it offers a common and unified framework where as varied forms of problems as H-coloring and CNF-SAT(Γ) occurs as natural special cases. As

we have seen, the complexity of these two (on the surface) very different problems can be completely determined and uniformly explained just by the presence (or absence) of certain classes of polymorphisms of the corresponding constraint languages.

In light of the long-standing open problem of classifying the complexity of $\textsc{Csp}(\Gamma)$, it seems difficult to classify the complexity of many of the problems studied in this thesis for arbitrary finite constraint languages. Hence, we focus on restricted classes of constraint languages which are particularly interesting.

Linear Equations

For the first three problems $\#\textsc{Csp}(\Gamma)$, $\textsc{Equiv-Csp}(\Gamma)$, and $\textsc{Iso-Csp}(\Gamma)$, we restrict our attention to the case where the constraints are linear equations over some finite semigroup (actually a group for $\textsc{Equiv-Csp}(\Gamma)$ and $\textsc{Iso-Csp}(\Gamma)$). Remember that a semigroup is a set S together with a binary associative operation $\cdot$ (a group is a semigroup with the additional requirements that there is an identity element and every element have an inverse).

Systems of linear equations are the canonical examples of CSPs. A linear equation over a fixed finite semigroup $(S, \cdot)$ is of the form $x_1 \cdot x_2 \cdot \ldots \cdot x_k = y_1 \cdot y_2 \cdot \ldots \cdot y_j$ where each x_i and y_i is either a variable or a constant in S.

The way we parameterize the problems is by fixing the semigroup $(S, \cdot)$ and requiring that all constraints are linear equations over $(S, \cdot)$. Hence, the semigroup $(S, \cdot)$ gives rise to a constraint language Γ_S consisting of all relations expressible as the set of solutions to equations over $(S, \cdot)$. For example, the equation $x \cdot y \cdot z = c$, where x, y, z are variables and c is a constant, gives rise to the following ternary relation $R = \{(x, y, z) \mid x \cdot y \cdot z = c\}$.

The complexity of problems restricted to constraint languages of the form Γ_S has been studied before. The complexity of $\textsc{Csp}(\Gamma_S)$ was classified by Goldmann and Russell [27] in the important special case when S is a group. They prove that the

problem is in **P** if the group is Abelian (i.e., commutative), and **NP**-complete if the group is non-Abelian. Tesson [49] and Klima et al. [35] classify the complexity of $\textsc{Csp}(\Gamma_S)$ for a large class of semigroups S. Moreover, they prove that for each finite constraint language Γ there is a semigroup $(S, \cdot)$ such that $\textsc{Csp}(\Gamma)$ and $\textsc{Csp}(\Gamma_S)$ are polynomial-time equivalent. Hence, giving a complexity classification of $\textsc{Csp}(\Gamma_S)$ for all finite semigroups $(S, \cdot)$ would amount to resolving the Feder-Vardi dichotomy conjecture for CSPs.

We remark in passing that the complexity of several important optimization problems have been classified over constraint languages of the form Γ_G for fixed finite groups $(G, \cdot)$, see for example [24, 28, 38].

In Paper I we classify the complexity of $\#\textsc{Csp}(\Gamma_S)$ for a large class of semigroups S. We prove that $\#\textsc{Csp}(\Gamma_S)$ is tractable when $(S, \cdot)$ is an Abelian group and **#P**-complete when $(S, \cdot)$ is a non-Abelian group. In the case of monoids, $\#\textsc{Csp}(\Gamma_S)$ is **#P**-complete when $(S, \cdot)$ is a monoid that is not a group. Given a semigroup $(S, \cdot)$, we say that an element a in S is divisible if and only if there exists b, c in S such that $b \cdot c = a$. In the case of semigroups where all elements are divisible we show that $\#\textsc{Csp}(\Gamma_S)$ is tractable if $(S, \cdot)$ is a direct product of an Abelian group and a rectangular band, and **#P**-complete otherwise. The class of semigroups where all elements have divisors contains most of the interesting semigroups, in particular regular semigroups.

The equivalence and isomorphism problems for systems of equations over finite groups $(G, \cdot)$, i.e., $\textsc{Equiv-Csp}(\Gamma_G)$ and $\textsc{Iso-Csp}(\Gamma_G)$, are studied in Paper II. Our main results are the following:

- $\textsc{Equiv-Csp}(\Gamma_G)$ is in **P** if G is Abelian, and **coNP**-complete if G is non-Abelian.
- $\textsc{Iso-Csp}(\Gamma_G)$ is in **NP** and at least as hard as graph isomorphism if G is Abelian; and for non-Abelian groups it is in $\mathbf{\Sigma_2^P}$ and **coNP**-hard.

- The Iso-Csp(Γ_G) problem for the restriction where the number of variable occurrences in each equation is bounded by a constant k, is exactly as hard as graph isomorphism (i.e., Graph Isomorphism-complete) for Abelian groups. For non-Abelian groups it is in $\mathbf{P}_{||}^{\mathbf{NP}}$ (which is the class of problems solvable in polynomial time by making nonadaptive queries to an **NP**-oracle) and **coNP**-hard.

- Finally, we prove that the problem of counting the number of isomorphisms between two systems of linear equations over a group G is no harder than deciding whether an isomorphism exists at all.

Nonmonotonic Logics

Much research has been devoted to determine the complexity of reasoning in various nonmonotonic logic formalisms [16]. Two of the most well-studied formalisms are circumscription, which is the topic of Papers III and IV, and abduction which is the topic of Paper V.

Much of the previous research on complexity of reasoning in nonmonotonic logics has focused on identifying islands of tractability (for instance by restricting the constraint language Γ) [5, 7, 26] and determining the complexity in the general case (without restrictions on the constraint language) [11, 12]. In this thesis we take a different perspective and aim at proving complete classifications for circumscription and abduction problems parameterized by the set of allowed constraints Γ.

In Paper III we relate the complexity of Min-Csp(Γ) and Min-Inf-Csp(Γ) to the complexity of Csp(Γ). As a corollary to Bulatov's [9] complexity classifications of Csp(Γ) over three-element domains we get a dichotomy (between **P** and **coNP**-completeness) for the complexity of Min-Csp(Γ) when Γ is a constraint language over a three-element domain. Similarly, we prove a dichotomy between membership in **coNP** and $\mathbf{\Pi_2^P}$-comp-

leteness for Min-Inf-Csp(Γ) over three-element domains.

In Papers IV and V we prove complete complexity classifications for Min-Inf-Csp(Γ) and Abduction(Γ) for all Boolean constraint languages Γ. The exact borderlines for the complexity of Min-Inf-Csp(Γ) are as follows:

- **P:** If Γ is Horn and dual-Horn, width-2 affine, or negative Horn.
- **coNP-complete:** If Γ is Horn, dual-Horn, affine, or bijunctive (and not Horn and dual-Horn, width-2 affine, or negative Horn).
- $\mathbf{\Pi_2^P}$**-complete:** If Γ is neither Horn, nor dual-Horn, nor affine, nor bijunctive.

The borderlines for Abduction(Γ) are as follows:

- **P**: If Γ is affine of width 2.
- **NP**-complete: If Γ is Horn, dual Horn, bijunctive or affine (and not affine of width 2).
- $\mathbf{\Sigma_2^P}$-complete: If Γ is neither Horn, nor dual-Horn, nor affine, nor bijunctive.

Structure of the Thesis

In this introductory chapter of the thesis we have tried to put our results in context with previous work on complexity classifications in general and of Csp(Γ) in particular. We continue with more detailed summaries of the results of the individual papers.

The rest of this thesis consists of five independent papers, each one containing the necessary background and definitions needed for the results. Due to this fact, the introductory parts of the individual papers sometimes overlap and contain repetitions. Hopefully, this should not be too disturbing. All of the

papers are written to be fairly self-contained and, in general, only knowledge of discrete mathematics and some basics of computational complexity is assumed. The papers are presented in the same order as they were written and they can be read in any order. The papers in this thesis are more or less identical to the original published papers, except for minor typographical adjustments.

Summary of the Papers

In this section we give summaries of the topics and results of the papers constituting this thesis. We try to do this at a sufficiently abstract level to avoid the need for formal definitions.

Paper I: The Complexity of Counting Solutions to Systems of Equations over Finite Semigroups

In this paper we study the complexity of counting the number of solutions to systems of linear equations over finite semigroups. The problem is parameterized by the semigroup S, and our goal is to classify the complexity of counting the number of solutions to systems of equations over each finite semigroup S.

The problem is naturally viewed as a special case of the problem of counting the number of solutions of a $\textsc{Csp}(\Gamma)$ instance (i.e., the $\#\textsc{Csp}(\Gamma)$ problem), by identifying a constraint language Γ_S with each finite semigroup S. The constraint language Γ_S is the set of all relations expressible by linear equations over S.

Creignou and Hermann [18] prove a dichotomy theorem (between membership in **FP** and **#P**-completeness) for the complexity of the $\#\textsc{Csp}(\Gamma)$ problem over the Boolean domain. Bulatov and Dalmau [10] prove that the complexity of $\#\textsc{Csp}(\Gamma)$ is completely determined by the polymorphisms of Γ and using

this result they give some very general tractability and hardness results for #Csp(Γ) over arbitrary finite domains.

By making extensive use of the results for #Csp(Γ), due to Bulatov and Dalmau [10], we manage to classify the complexity of #Csp(Γ_S) for a large class of semigroups. In particular, we prove a dichotomy for the complexity of counting the number of solutions to systems of linear equations over any fixed finite regular semigroup. More specifically, we prove that the problem is in **FP** if the regular semigroup S is a direct product of a rectangular band and an Abelian group, and that it is **#P**-complete otherwise. An easy consequences of this result is that counting the number of solutions to systems of linear equations over a fixed finite group is in **FP** if the group is Abelian, and **#P**-complete if the group is non-Abelian.

Finally, we remark that Klíma et al. [36], building upon the results in this paper and [11], completely classify the complexity of counting the number of solutions to systems of equations over any finite semigroup.

Paper II: The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups

Isomorphism problems are notoriously known to be hard to classify from a complexity point of view. The graph isomorphism problem plays a special role in complexity theory [32] since it is the most famous candidate for a natural problem in **NP** that might be of intermediate complexity, i.e., being neither in **P** nor **NP**-complete.

In this paper we study the complexity of the isomorphism problem for systems of linear equations over finite groups. We say that two systems of linear equations (over the same group G) are equivalent they have the same set of solutions. Two systems of linear equations (over the same group G) are isomorphic if the

variables in one of them can be permuted so that the resulting systems of equations are equivalent. Just as in Paper I we parameterize the problem by fixing the underlying group G. That is, our goal is to classify the complexity of the isomorphism problem for systems of linear equations over G, for each finite group G. We also study a slightly restricted variant of the problem in which the total number of variable occurrences in each equation is bounded by a constant k.

Our main results are the following:

- The equivalence problem for systems of linear equations (over the finite group G) is in $\mathbf{P}$ if G is Abelian, and **coNP**-complete if G is non-Abelian.
- The isomorphism problem for systems of linear equations is in **NP** and at least as hard as graph isomorphism if G is Abelian; and for non-Abelian groups it is in $\mathbf{\Sigma_2^P}$ and **coNP**-hard.
- The isomorphism problem for the restriction where the number of variable occurrences in each equation is bounded by a constant k is exactly as hard as graph isomorphism (i.e., Graph Isomorphism-complete) for Abelian groups. For non-Abelian groups it is in $\mathbf{P}_{||}^{\mathbf{NP}}$ and **coNP**-hard.
- Finally, we prove that the problem of counting the number of isomorphisms between two systems of linear equations over a group G is no harder than deciding whether an isomorphism exists at all.

Interestingly, many of the proofs and proof techniques used for proving these results closely follows proofs for other isomorphism problems such as the formula isomorphism problem [2] and isomorphism problems for Boolean constraints [6, 7]. This seems to suggest that a more unified treatment of these and similar isomorphism problems is possible.

Paper III: An Algebraic Approach to the Complexity of Propositional Circumscription

Circumscription, first introduced by McCarthy [41], is one of the most important and well-studied formalisms in the realm of non-monotonic reasoning. The key intuition behind circumscription is that by focusing on minimal models of formulas (instead of all models), we arrive at a formalism which is closer to how common sense reasoning is performed by humans.

The notion of a minimal model can be defined in different ways, we use one of the most general definitions. Given a propositional (Boolean) formula T and a partition of the variables in T into three (possibly empty) disjoint subsets $(P; Z; Q)$, we define a partial order on satisfying models (extending the order $0 \leq 1$ on truth values) as follows. Let α, β be two models of T, then $\alpha \leq \beta$ if α and β assign the same value to the variables in Q and for every variable p in P, $\alpha(p) \leq \beta(p)$. Moreover, if there exists a variable p in P such that $\alpha(p) \neq \beta(p)$, we write $\alpha < \beta$. A minimal model of a formula T is a satisfying model α such that there exist no satisfying model β where $\beta < \alpha$.

Every logical formalism gives rise to two fundamental problems: model checking and inference. In the case of propositional circumscription, the model checking and inference problems can be formulated as follows.

- **Model checking:** Given a propositional formula T, a partition of the variables in T into three disjoint subsets $(P; Z; Q)$ and a truth assignment α, is α a minimal model of T?
- **Inference:** Given two propositional formulas T and T' and a partition of the variables in T into three disjoint (possibly empty) subsets $(P; Z; Q)$, is T' true in every minimal model of T?

We parameterize the problems by requiring that the theory T is built over a fixed finite (Boolean) constraint language Γ. That is,

the theory T consists of a conjunction of clauses/constraints, all of which must come from the constraint language Γ. We denote the parameterized versions of the problem by Min-Csp(Γ) and Min-Inf-Csp(Γ), respectively.

The problems Min-Csp(Γ) and Min-Inf-Csp(Γ) over Boolean constraint languages Γ are very well-studied from the computational complexity perspective [13, 14, 15, 21, 22, 33, 34]. In particular, Kirousis and Kolaitis prove a dichotomy (between membership in **P** and **coNP**-completeness) for the complexity of Min-Csp(Γ) where Γ is a Boolean constraint language.

In this paper we study the Min-Csp(Γ) and Min-Inf-Csp(Γ) problems for arbitrary constraint languages over finite partially ordered domains (i.e., the domain D is a partial order $(D, \leq)$ and the notion of minimal models/solutions is defined in terms of this partial order $\leq$).

Our first result shows that the algebraic approach to CSP is applicable to Min-Csp(Γ), i.e., that the complexity of the Min-Csp(Γ) problem is completely determined by the relational clone corresponding to Γ. Using this approach we give a very short and simple proof of Kirousis and Kolaitis complexity dichotomy for Min-Csp(Γ) over the Boolean domain. We also prove a tight correspondence between the complexity of Csp(Γ) and Min-Csp(Γ). As a corollary of Bulatov's classification of Csp(Γ) over three-element domains [9] we get a complexity dichotomy (between **P** and **coNP**-completeness) for Min-Csp(Γ) over three-element domains.

The story is essentially the same for the Min-Inf-Csp(Γ) problem. Using the algebraic approach together with Schaefer's and Bulatov's classifications of Csp(Γ) over Boolean and three-element domains, respectively, we are able to prove dichotomies (between membership in **coNP** and $\mathbf{\Pi_2^P}$-completeness) for the complexity of Min-Inf-Csp(Γ) over Boolean and three-element domains.

Paper IV: A Trichotomy in the Complexity of Propositional Circumscription

This second paper on propositional circumscription studies the special case of the inference problem from paper III where the background theory T consists of constraints over the Boolean domain. Hence, the parameter Γ is a constraint language over $\{0, 1\}$. Our approach in this paper is to use Post's lattice as a tool for proving a trichotomy (between **P**, **coNP**-completeness, and $\mathbf{\Pi_2^P}$-completeness) for the Min-Inf-Csp(Γ) problem (where Γ is a Boolean constraint language).

The dichotomy between membership in **coNP** and $\mathbf{\Pi_2^P}$-completeness is proved in Paper III, so all that remains is to classify the complexity of the Min-Inf-Csp(Γ) problems that reside within **coNP**. Cadoli and Lenzerini [15] study the complexity of the Min-Inf-Csp(Γ) problem extensively and prove several hardness results. Durand and Hermann prove an additional important hardness result in [21]. Until now, only one tractable case has been known for the Min-Inf-Csp(Γ) problem, namely the case where the theory T consists only of clauses having at most one positive and at most one negative literal (i.e., clauses that are both Horn and dual-Horn) [15]. We identify two additional tractable classes of constraint languages, namely, width-2 affine and Horn clauses only containing negative literals. Then we observe that Min-Inf-Csp(Γ) is **coNP**-complete for all remaining classes of constraint languages Γ. More specifically, by using the algebraic approach and Post's lattice, it is easy to realize that these three tractable classes together with known hardness results from the literature [15, 21], gives a complete classification for the complexity of the Min-Inf-Csp(Γ) problem over the Boolean domain. The exact borderlines for the complexity of Min-Inf-Csp(Γ) are as follows:

- **P:** If Γ is Horn and dual-Horn, width-2 affine, or negative Horn.

- **coNP-complete:** If Γ is Horn, dual-Horn, affine, or bijunctive (and not Horn and dual-Horn, width-2 affine, or negative Horn).
- **$\Pi_2^{\mathbf{P}}$-complete:** If Γ is neither Horn, nor dual-Horn, nor affine, nor bijunctive.

Paper V: Propositional Abduction is Almost Always Hard

Abduction is the fundamental reasoning process which consists of explaining observations by plausible causes taken from a given set of hypotheses. For instance, it is the problem of trying to derive diseases from observed symptoms, according to known rules relating both. This process was extensively studied by Peirce [8], and its importance to Artificial Intelligence was first emphasized by Morgan [42] and Pople [44].

We are interested here in propositional logic-based abduction, i.e., the background knowledge is represented by a propositional theory. More specifically, we study the computational complexity of the basic problem of deciding whether an explanation exists for a set of manifestations. More formally, given a propositional theory T formalizing a particular application domain, a set M of literals describing a set of manifestations, and a set H of literals containing possible hypotheses, decide whether M can be explained, that is, is there a set $E \subseteq H$ such that $T \cup E$ is consistent and logically entails M? We denote the ABDUCTION problem, as described above, where the theory T is restricted to the finite Boolean constraint language Γ, by ABDUCTION(Γ).

Eiter and Gottlob [23] prove (among other things) that ABDUCTION(Γ) is $\mathbf{\Sigma_2^P}$-complete in the general case. Moreover, many constraint language restrictions yielding ABDUCTION(Γ) problems of lower complexity are known (see for example [48, 51]).

In this paper we again use the algebraic approach via Post's lattice to give a complete classification of the complexity of the

ABDUCTION(Γ) problem for all Boolean constraint languages Γ. More precisely, we prove that ABDUCTION(Γ) is:

- In **P** if Γ is affine of width 2,
- Otherwise, **NP**-complete if Γ is Horn, dual Horn, bijunctive or affine,
- Otherwise, $\mathbf{\Sigma_2^P}$-complete.

As far as we know, the polynomial case and the minimal **NP**-hard languages that we exhibit are all new results.

We remark that our formalization of the ABDUCTION(Γ) problem is quite general in that we allow the possible hypotheses to be any set of literals and the manifestations to be any set of literals. The corresponding ABDUCTION(Γ) problem in which the manifestation is a single literal and the possible hypotheses H are of the form $H = \{v, \neg v \mid v \in V' \subseteq Vars(T)\}$ (i.e., a literal is in H if and only if its negation is) was recently classified by Creignou and Zanuttini in [19]. Their results together with the results in this paper shows that even minor changes of the form of allowed possible hypotheses and manifestations have a strong influence on the complexity of the problem.

References

[1] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, 36(1):30–52, 2005.

[2] M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM Journal on Computing*, 30(3):990–1009, 2000.

[3] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[4] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part I: Post's lattice with applications to complexity theory. *SIGACT News*, 34(4):38–52, 2003.

[5] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part II: Constraint satisfaction problems. *SIGACT News*, 35(1):22–35, 2004.

[6] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *Proc. Computer Science Logic (CSL 2002))*, pages 412–426., 2002.

[7] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. The complexity of boolean constraint isomorphism. In *Proc. 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, pages 164–175, 2004.

[8] J. Buchler, editor. *Philosophical Writings of Peirce*. Dover, New York, 1955.

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

[10] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, 2003.

[11] A. Bulatov and M. Grohe. The complexity of partition functions. *Theoretical Computer Science*, 348(2-3):148–186, 2005.

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

[13] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 42:113–118, 1992.

[14] M. Cadoli. *Tractable Reasoning in Artificial Intelligence*. Number 941 in Lecture Notes in Artificial Intelligence. Springer Verlag Berlin Heidelberg, 1995.

[15] M. Cadoli and M. Lenzerini. The complexity of closed world reasoning and circumscription. *Journal of Computer and System Sciences*, 48(2):255–301, 1994.

[16] M. Cadoli and M. Schaerf. A survey of complexity results for nonmonotonic logics. *Journal of Logic Programming*, 17(2/3&4):127–160, 1993.

[17] D. Cohen and P. Jeavons. *Handbook of Constraint Programming*. Chapter: The Complexity of Constraint Languages. Elsevier, 2006.

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

[19] N. Creignou and B. Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM Journal on Computing*, 36(1):207–229, 2006.

[20] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[21] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is coNP-complete. In *Proc. of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2003)*, pages 451–462, 2003.

[22] T. Eiter and G. Gottlob. Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[23] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):3–42, 1995.

[24] L. Engebretsen, J. Holmerin, and A. Russell. Inapproximability results for equations over finite groups. *Theoretical Computer Science*, 312(1):17–45, 2004.

[25] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1999.

[26] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[27] M. Goldmann and A. Russell. The complexity of solving equations over finite groups. *Information and Computation*, 178(1):253–262, 2002.

[28] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[29] P. Hell and J. Nešetřil. On the complexity of H-colouring. *Journal of Combinatorial Theory B*, 48:92–110, 1990.

[30] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.

[31] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[32] H. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, Boston, 1993.

[33] L. Kirousis and P. Kolaitis. The complexity of minimal satisfiability problems. *Information and Computation*, 187(1):20–39, 2003.

[34] L. Kirousis and P. Kolaitis. A dichotomy in the complexity of propositional circumscription. *Theory of Computing Systems*, 37(6):695–715, 2004.

[35] O. Klima, P. Tesson, and D. Therien. Dichotomies in the complexity of solving systems of equations over finite semigroups. *Theory of Computing Systems*, 40(3):263–297, 2007.

[36] O. Klíma, B. Larose, and P. Tesson. Systems of equations over finite semigroups and the #CSP dichotomy conjecture. In *Proc. of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS 2006)*, pages 584–595, 2006.

[37] P. Kolaitis and M.Y. Vardi. *Finite Model Theory and Its Applications*, chapter A Logical Approach to Costraint Satisfaction. Springer, 2007.

[38] F. Kuivinen. Tight approximability results for the maximum solution equation problem over Z_p. In *Proc. of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005)*, pages 628–639, 2005.

[39] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[40] R. McKenzie and M. Maróti. Existence theorems for weakly symmetric operations. Submitted, 2006.

[41] J. McCarthy. Circumscription - a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[42] C. Morgan. Hypothesis generation by machine. *Artificial Intelligence*, 2:179–187, 1971.

[43] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1995.

[44] H. Pople. On the mechanization of abductive logic. In *Proc. of the 3rd International Joint Conference on Artificial Intelligence (IJCAI 1973)*, pages 147–152, 1973.

[45] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

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

[47] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226, 1978.

[48] B. Selman and H. Levesque. Abductive and default reasoning: A computational core. In *Proc. of the 8th National Conference on Artificial Intelligence (AAAI 1990)*, pages 343–348, 1990.

[49] P. Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, School of Computer Science, McGill University, Montreal, 2003.

[50] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1993.

[51] B. Zanuttini. New polynomial classes for logic-based abduction. *Journal of Artificial Intelligence Research*, 19:1–10, 2003.

Paper I

The Complexity of Counting Solutions to Systems of Equations over Finite Semigroups

Gustav Nordh and Peter Jonsson

Abstract

We study the computational complexity of counting the number of solutions to systems of equations over a fixed finite semigroup. We show that if the semigroup is a group, the problem is tractable if the group is Abelian and **#P**-complete otherwise. If the semigroup is a monoid (that is not a group) the problem is **#P**-complete. In the case of semigroups where all elements have divisors we show that the problem is tractable if the semigroup is a direct product of an Abelian group and a rectangular band, and **#P**-complete otherwise. The class of semigroups where all elements have divisors contains most of the interesting semigroups e.g. regular semigroups. These results are proved by the use of powerful techniques from universal algebra.

1 Introduction

The computational complexity of deciding whether systems of equations over a fixed finite semigroup are solvable has been intensively studied in the past. In [3], it is proved that when the semigroup is a group, the problem is tractable if the group is Abelian and **NP**-complete otherwise. This line of research continued in [6, 7], where the corresponding problem for finite monoids was given a complete solution. Some partial results in the general case of finite semigroups have been proved in [7]. Note that even the restricted problem of determining the computational complexity of solving systems of equations over a fixed regular semigroup is still open.

In this paper we study the computational complexity of counting solutions to systems of equations over a fixed finite semigroup. This problem has not been given the same attention as the corresponding decision problem. A system of equations over a fixed finite semigroup $(S, \cdot)$ is a collection of equations of the form $w_1 \cdot w_2 \cdot \cdots \cdot w_k = y_1 \cdot y_2 \cdot \cdots \cdot y_j$ where each w_i and y_i is either a variable or a constant in S. We denote the problem of counting the number of solutions to systems of equations over the semigroup $(S, \cdot)$ by $\#EQN_S^*$. The notation $\#EQN_S^*$ is used in order to comply with the notation in [3, 6, 7]. The presence of the * in $\#EQN_S^*$ is motivated by the need to distinguish this problem from the problem of counting the number of solutions to a single equation over $(S, \cdot)$ (denoted by $\#EQN_S$). The complexity of $\#EQN_S^*$ is measured in the size of the systems of equations (the size of $(S, \cdot)$ is fixed and does not matter).

We prove that $\#EQN_G^*$ is tractable when $(G, \cdot)$ is an Abelian group and **#P**-complete when $(G, \cdot)$ is a non-Abelian group. In the case of monoids, $\#EQN_M^*$ is **#P**-complete when $(M, \cdot)$ is a monoid that is not a group. Given a semigroup $(S, \cdot)$, we say that an element a in S is divisible if and only if there exists b, c in S such that $b \cdot c = a$. In the case of semigroups where all elements are divisible we show that $\#EQN_S^*$ is tractable if $(S, \cdot)$ is a direct

product of an Abelian group and a rectangular band, and **#P**-complete otherwise. The class of semigroups where all elements have divisors contains most of the interesting semigroups, in particular regular semigroups. Moreover we prove that if $(S, \cdot)$ is a semigroup with zero, then $\#EQN_S^*$ is tractable if $(S, \cdot)$ is such that for all elements x, y in S, $x \cdot y = 0$, and **#P**-complete otherwise. Important to note is that all the conditions we impose on our semigroups to ensure the tractability (or **#P**-completeness) of $\#EQN_S^*$ can be checked in polynomial time (in the size of the semigroups).

Our approach for obtaining dichotomy theorems for $\#EQN_S^*$ relies heavily on the use of powerful algebraic techniques that have previously shown to be very useful in proving dichotomy theorems for Constraint Satisfaction Problems (CSPs) [1, 2]. It is our belief that similar algebraic techniques can also be used in a unified approach to determine the complexity of the decision problem.

The paper is organized as follows. In Section 2 we give the basic definitions and results needed in the rest of this paper. Section 3 contains all our tractability proofs for $\#EQN_S^*$. In Section 4 we prove the dichotomy for the computational complexity of $\#EQN_S^*$ when $(S, \cdot)$ is a group or monoid. In Section 5 we prove the dichotomy for the computational complexity of $\#EQN_S^*$ when $(S, \cdot)$ is a semigroup with zero or is in the class of semigroups where all elements are divisible.

2 Preliminaries

2.1 CSPs

In this section we introduce the notation and basic results on CSPs that we will use in the rest of this paper.

Let D be a finite set. The set of all n-tuples of elements of D is denoted by D^n. Any subset of D^n is called an n-ary relation

on D. The set of all finitary relations over D is denoted by R_D.

Definition 7 A constraint language over a finite set, D, is an arbitrary set $\Gamma \subseteq R_D$.

Definition 8 The counting constraint satisfaction problem over the constraint language Γ, denoted $\#CSP(\Gamma)$, is defined to be the counting problem with instance (V, D, C), where

- V is a finite set of variables,
- D is a set of values (sometimes called a domain), and
- C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation over the set D, belonging to Γ, called the constraint relation.

We ask for the number of solutions to (V, D, C), that is, the number of functions from V to D such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

The corresponding decision problem where we are interested in the existence of a solution is denoted $CSP(\Gamma)$. In the context of this paper D is the elements of a finite semigroup $(S, \cdot)$ and Γ is the set of relations expressible as equations over $(S, \cdot)$ (denoted by Γ_S). For example the equation $x \cdot y = c$ where x, y are variables and c is a constant gives rise to the following relation $R = \{(x, y) | x \cdot y = c\}$.

Next we consider operations on the domain D. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows.

Definition 9 Let f be a k-ary operation on D and let R be an n-ary relation over D. For any collection of k tuples, $t_1, t_2, \ldots, t_k \in$

R, the n-tuple $f(t_1, t_2, \ldots, t_k)$ is defined as follows:

$$f(t_1, t_2, \ldots, t_k) =$$

$$(f(t_1[1], t_2[1], \ldots, t_k[1]), \ldots, f(t_1[n], t_2[n], \ldots, t_k[n]))$$

where $t_j[i]$ is the i:th component in tuple t_j.

A technique that has shown to be useful in determining the computational complexity of $CSP(\Gamma)$ is that of investigating whether Γ is closed under certain families of operations [5].

Definition 10 For any constraint relation $\varrho_i \in \Gamma$ if f is an operation such that for all $t_1, t_2, \ldots, t_k \in \varrho_i$: $f(t_1, t_2, \ldots, t_k) \in \varrho_i$, then ϱ_i is closed under f. If all constraint relations in Γ are closed under f then Γ is closed under f. An operation f such that Γ is closed under f is called a polymorphism of Γ. The set of all polymorphisms of Γ is denoted $Pol(\Gamma)$.

In this paper we will only be concerned with a particular family of ternary operations called Mal'tsev operations. An operation M is a *Mal'tsev operation* if and only if for all x, y, $M(x, y, y) = M(y, y, x) = x$.

The following result provides us with the key to prove the #**P**-completeness of constraint languages.

Theorem 11 ([2]) *If Γ is a finite constraint language that is closed under no Mal'tsev operation then $\#CSP(\Gamma)$ is #**P**-complete.*

Throughout this paper we will use the following notation for the application of an arbitrary Mal'tsev operation M on tuples in a relation R.

$$\begin{array}{rll} & (a,b) & \in R \\ & (b,b) & \in R \\ M & (b,a) & \in R \\ \hline & (a,a) & \end{array}$$

In the example above $(a,b),(b,b)$ and (b,a) are three tuples in a relation R. If $(a,a) \notin R$ we have shown that R is closed under no Mal'tsev operation, since $(M(a,b,b),\ M(b,b,a)) = (a,a)$ for *any* Mal'tsev operation M.

All of our **#P**-completeness proofs for $\#EQN_S^*$ follows this outline: First we choose a relation R in Γ_S i.e., R is expressed by a specific equation E over $(S,\cdot)$. Then we show that R is closed under no Mal'tsev operation, thus proving the **#P**-completeness of $\#CSP(\Gamma_S)$. It should be clear that this implies the **#P**-completeness of $\#EQN_S^*$. It is also easy to realize that any system of equations over a finite semigroup $(S,\cdot)$ can be (parsimoniously) reduced in polynomial time to a system of equations over $(S,\cdot)$ such that no equation in this system of equations contains more than three variables. Equations containing more that three variables can be split into smaller parts by introducing new variables. Hence we can safely assume that all constraint languages we consider are finite.

The following theorem is very useful when it comes to proving tractability results for $\#CSP(\Gamma)$.

Theorem 12 ([2]) *A Mal'tsev operation $M(x,y,z) = x*y^{-1}*z$ where $*$ and $^{-1}$ are the operations of an Abelian group is called an affine operation. If $M \in Pol(\Gamma)$ then $\#CSP(\Gamma)$ is tractable.*

2.2 Definitions and Results from Semigroup Theory

For the convenience of the reader we here give the definitions and results from semigroup theory that we use in the rest of this paper. The proofs are available in (or easily deducible from) any standard reference on semigroup theory such as [4].

A *semigroup* $(S,\cdot)$ is a set S with an associative binary operation $\cdot$. A *semigroup with zero* is a semigroup that contains an element 0 such that for all x in S, $0 \cdot x = x \cdot 0 = 0$. A *monoid*

is a semigroup that has an identity element. A *band* is a semigroup where all elements are idempotent (that is for all x in S, $x \cdot x = x$). A *rectangular band* is a band where for all a, b in S, $a \cdot b \cdot a = a$. A *regular semigroup* is a semigroup where for all a in S there exists an element b in S such that $a \cdot b \cdot a = a$. A *right zero semigroup* is a semigroup where for all a, b in S, $a \cdot b = b$ (*left zero semigroups* are defined in an analogous way). If two semigroups $(S, \cdot_s)$ and $(T, \cdot_t)$ are *isomorphic* we denote this by $(S, \cdot_s) \cong (T, \cdot_t)$. If $(S, \cdot_s)$ and $(T, \cdot_t)$ are semigroups, then the Cartesian product $(S, \cdot_s) \times (T, \cdot_t)$ becomes a semigroup if we define $(s, t) \cdot (s', t') = (s \cdot_s s', t \cdot_t t')$. We refer to this semigroup as the *direct product* of $(S, \cdot_s)$ and $(T, \cdot_t)$. Let $Div(S)$ denote the set of divisible elements in S where $(S, \cdot)$ is a finite semigroup (a is divisible if and only if there exists b, c in S such that $b \cdot c = a$).

Theorem 13 *If $(S, \cdot)$ is a finite semigroup and a an arbitrary element in S, then a^m is idempotent for some m.*

Theorem 14 *If $(M, \cdot)$ is a finite monoid with the identity element as the only idempotent then $(M, \cdot)$ is a group.*

Theorem 15 *Let $(S, \cdot)$ be a rectangular band, then there exist a left zero semigroup $(L, \cdot_l)$ and a right zero semigroup $(R, \cdot_r)$ such that $(S, \cdot) \cong (L, \cdot_l) \times (R, \cdot_r)$.*

The following theorem is folklore.

Theorem 16 *A regular semigroup $(S, \cdot)$ whose idempotents form a rectangular band is isomorphic to a direct product of a group and a rectangular band.*

We give some hints on how to prove this theorem. By using Green's relations it can be proved that $(S, \cdot)$ is a completely simple semigroup, and since $(S, \cdot)$ is orthodox it follows from the Rees-Suschkewitch Theorem [4] that $(S, \cdot)$ is a direct product of a group and the rectangular band of its idempotents.

3 Tractability Results

Due to the similar structure of the tractability proofs for the tractable cases of $\#EQN_S^*$ we have gathered all of these proofs in this section to make a unified approach possible. In the rest of this section assume that all equations in the systems of equations are of the following two types: equations only containing variables, $x_1 \cdot \cdots \cdot x_n = \hat{x}_1 \cdot \cdots \cdot \hat{x}_m$ or equations with a single variable on the left hand side and a constant on the right hand side, $x = a$. We can make this assumption because it is easy to reduce (parsimoniously and in polynomial time) any system of equations over a finite semigroup into a system of equations of this form.

Let A be finite set and let $f : A^n \to A$ and $g : A^m \to A$ be two operations. If $f(a, \ldots, a) = a$ for all $a \in A$, then f is *idempotent*. We say that g *commutes* with f if for all $a_{11}, \ldots, a_{1m}, \ldots, a_{nm} \in A$,

$$g(f(a_{11}, \ldots, a_{n1}), \ldots, f(a_{1m}, \ldots, a_{nm})) =$$
$$f(g(a_{11}, \ldots, a_{1m}), \ldots, g(a_{n1}, \ldots, a_{nm}))$$

The following lemma is the foundation for our tractability results.

Lemma 17 *Let $(S, \cdot)$ be a finite semigroup. If $f : S^k \to S$ is an idempotent operation and $\cdot$ commutes with f, then $f \in Pol(\Gamma_S)$.*

Proof: Arbitrarily choose a relation $R \in \Gamma_S$. If R is a unary relation (i.e. $R(x)$ holds if and only if $x = a$ for some $a \in S$), then R is closed under f since f is idempotent. Otherwise, $R = \{(x_1, \ldots, x_n, \hat{x}_1, \ldots, \hat{x}_m) \mid x_1 \cdot \cdots \cdot x_n = \hat{x}_1 \cdot \cdots \cdot \hat{x}_m\}$ and we arbitrarily choose k tuples $t_1, \ldots, t_k$ in R. Now, R is closed under f if the following holds:

$$\prod_{i=1}^{n} f(t_1[i], \ldots, t_k[i]) = \prod_{i=n+1}^{n+m} f(t_1[i], \ldots, t_k[i]) \qquad (*)$$

We know that $\cdot$ commutes with f so

$$\prod_{i=1}^{n} f(t_1[i], \ldots, t_k[i]) = f\left(\prod_{i=1}^{n} t_1[i], \ldots, \prod_{i=1}^{n} t_k[i]\right)$$

and

$$\prod_{i=n+1}^{n+m} f(t_1[i], \ldots, t_k[i]) = f\left(\prod_{i=n+1}^{n+m} t_1[i], \ldots, \prod_{i=n+1}^{n+m} t_k[i]\right).$$

Since $t_1, \ldots, t_k$ are tuples in R, it follows that

$$\prod_{i=1}^{n} t_j[i] = \prod_{i=n+1}^{n+m} t_j[i]$$

for all j and $(*)$ holds. □

Lemma 18 *Let $(S, \cdot)$ be a finite semigroup. If f is an affine operation and $\cdot$ commutes with f, then $\#EQN_S^*$ is tractable.*

Proof: We see that f is idempotent since $f(a, a, a) = a * a^{-1} * a = a$ so $f \in Pol(\Gamma_S)$ by Lemma 17. Consequently, Theorem 12 implies that $\#EQN_S^*$ is tractable. □

Our tractability results for $\#EQN_S^*$ are contained in the following theorem.

Theorem 19 *$\#EQN_S^*$ is tractable when*

1. *$(S, \cdot)$ is an Abelian group.*
2. *$(S, \cdot)$ is a left zero or right zero semigroup.*
3. *$(S, \cdot)$ is a semigroup with zero such that for all x, y in S, $x \cdot y = 0$.*

4. $(S,\cdot) \cong (T,*) \times (T',*')$ *and* $\#EQN_T^*$, $\#EQN_{T'}^*$ *are tractable.*

Proof: If $(S,\cdot)$ is an Abelian group, then let $M(x,y,z) = x \cdot y^{-1} \cdot z$. We show that $\cdot$ commutes with M and tractability follows from Lemma 18. Let a,b,c be arbitrary tuples in S^2, then $M(a[1],b[1],c[1]) \cdot M(a[2],b[2],c[2]) = a[1] \cdot b[1]^{-1} \cdot c[1] \cdot a[2] \cdot b[2]^{-1} \cdot c[2] = a[1] \cdot a[2] \cdot (b[1] \cdot b[2])^{-1} \cdot c[1] \cdot c[2] = M(a[1] \cdot a[2], b[1] \cdot b[2], c[1] \cdot c[2])$.

If $(S,\cdot)$ is a right-zero group (the proof for left-zero groups is analogous), then let $M(x,y,z) = x * y^{-1} * z$ where the operation $*$ is chosen such that $(S,*)$ is an Abelian group (such a choice is always possible). Let a,b,c be arbitrary tuples in S^2, then $M(a[1],b[1],c[1]) \cdot M(a[2],b[2],c[2]) = M(a[2],b[2],c[2]) = M(a[1] \cdot a[2], b[1] \cdot b[2], c[1] \cdot c[2])$ and tractability follows from Lemma 18.

If $(S,\cdot)$ is a semigroup with zero such that for all x,y in S, $x \cdot y = 0$, then let $M(x,y,z) = x * y^{-1} * z$ where the operation $*$ is chosen such that $(S,*)$ is an Abelian group. Let a,b,c be arbitrary tuples in S^2, then $M(a[1],b[1],c[1]) \cdot M(a[2],b[2],c[2]) = 0 = M(a[1] \cdot a[2], b[1] \cdot b[2], c[1] \cdot c[2])$ and tractability follows from Lemma 18.

If $(S,\cdot) \cong (T,*) \times (T',*')$, then a system of equations over $(S,\cdot)$ can be split into two independent systems of equations, one over $(T,*)$ and one over $(T',*')$. The number of solutions to the system of equations over $(S,\cdot)$ equals the product of the number of solutions to the systems of equations over $(T,*)$ and $(T',*')$ respectively. Hence if $\#EQN_T^*$ and $\#EQN_{T'}^*$ are tractable, then so is $\#EQN_S^*$. □

4 Finite Groups and Monoids

In this section we prove our dichotomies for the complexity of the counting problem over finite groups and monoids. Note

that the decision dichotomy proved in [6] and the counting dichotomy we prove differs in the case of monoids, for example it is tractable to decide whether systems of equations over a commutative band are solvable, but to count the number of solutions is **#P**-complete.

Theorem 20 *Let $(G,\cdot)$ be a finite group, then $\#EQN^*_G$ is tractable if $(G,\cdot)$ is Abelian, and **#P**-complete otherwise.*

Proof: We begin by proving the **#P**-completeness of $\#EQN^*_G$ when $(G,\cdot)$ is a non-Abelian group. Because $(G,\cdot)$ is a non-Abelian group there exists two elements a, b in G such that $a\cdot b \neq b\cdot a$. Consider the following relation $R = \{(x,y)|x\cdot y = y\cdot x\}$. It is easy to see that R is closed under no Mal'tsev operation.

$$\begin{array}{rlll} & (a,1) & \in R & a\cdot 1 = 1\cdot a \\ & (1,1) & \in R & 1\cdot 1 = 1\cdot 1 \\ M & (1,b) & \in R & 1\cdot b = b\cdot 1 \\ \hline & (a,b) & \notin R & a\cdot b \neq b\cdot a \end{array}$$

Hence $\#EQN^*_G$ is **#P**-complete for non-Abelian groups. The tractability part is proved in Theorem 19. □

Next we consider the case where the semigroup is a finite monoid.

Theorem 21 *Let $(M,\cdot)$ be a finite monoid that is not a group, then $\#EQN^*_M$ is **#P**-complete.*

Proof: According to Theorem 14 we know that there exist an element a in M such that $a \neq 1$ and $aa = a$. Consider the relation $R = \{(x,y)|x\cdot y = a\}$. It is easy to see that R is closed under no Mal'tsev operation.

$$\begin{array}{rlll} & (1,a) & \in R & 1\cdot a = a \\ & (a,a) & \in R & a\cdot a = a \\ M & (a,1) & \in R & a\cdot 1 = a \\ \hline & (1,1) & \notin R & 1\cdot 1 = 1 \neq a \end{array}$$

Thus, $\#EQN^*_M$ is **#P**-complete. □

5 Finite Semigroups

The computational complexity of deciding whether systems of equations over a fixed finite semigroup are solvable has been studied in [7], where some evidence is given that a dichotomy in the decision case would be very hard to prove. Even the restricted problem of proving a dichotomy for regular semigroups is open. We prove that for semigroups where all elements have divisors the counting problem is tractable if the semigroup is a direct product of an Abelian group and a rectangular band, and **#P**-complete otherwise. Moreover we prove that for semigroups with zero, the problem is tractable if the semigroup is such that for all elements x, y $x \cdot y = 0$, and **#P**-complete otherwise.

The following lemma will be very useful in the proofs that follows.

Lemma 22 *Let $(S, \cdot)$ be a semigroup and I be the set of idempotents in S. A necessary condition for $\#EQN_S^*$ to be tractable is that the following holds: for all a in I and for all b, c in S, $b \cdot c = b \cdot a \cdot c$. Otherwise $\#EQN_S^*$ is **#P**-complete.*

Proof: Assume that there exists a in I and that there exists b, c in S and $b \cdot c \neq b \cdot a \cdot c$. Consider the relation $R = \{(x, y) | x \cdot y = b \cdot a \cdot c\}$. R is closed under no Mal'tsev operation.

	$(b, a \cdot c)$	$\in R$	$b \cdot a \cdot c = b \cdot a \cdot c$
	$(b \cdot a, a \cdot c)$	$\in R$	$b \cdot a \cdot a \cdot c = b \cdot a \cdot c$
M	$(b \cdot a, c)$	$\in R$	$b \cdot a \cdot c = b \cdot a \cdot c$
	(b, c)	$\notin R$	$b \cdot c \neq b \cdot a \cdot c$

Hence $\#EQN_S^*$ is **#P**-complete. □

We continue by proving a dichotomy in the case where the semigroup is a band.

Theorem 23 *If $(S, \cdot)$ is a band, then $\#EQN_S^*$ is tractable if $(S, \cdot)$ is a rectangular band and **#P**-complete otherwise.*

Proof: It follows directly from Lemma 22 that if S is a band that is not a rectangular band then $\#EQN_S^*$ is **#P**-complete. If $(S, \cdot)$ is a rectangular band we know from Theorem 15 that there exist a left zero semigroup $(L, \cdot_l)$ and a right zero semigroup $(R, \cdot_r)$ such that $(S, \cdot) \cong (L, \cdot_l) \times (R, \cdot_r)$. Thus the tractability of $\#EQN_S^*$ when $(S, \cdot)$ is a rectangular band follows from Theorem 19. □

The following lemma gives us an important necessary condition for the tractability of $\#EQN_S^*$.

Lemma 24 *Given a semigroup $(S, \cdot)$, it is a necessary condition for $\#EQN_S^*$ to be tractable that $(Div(S), \cdot)$ form a regular semigroup. Otherwise $\#EQN_S^*$ is **#P**-complete.*

Proof: Arbitrarily choose $a \in Div(S)$ and assume that $a = b \cdot c$ for some b, c. Since S is finite, we know from Theorem 13 that b^n and c^m are idempotent for some m, n. From Lemma 22 we know that $a = b \cdot c = b \cdot c^m \cdot b^n \cdot c = a \cdot c^{m-1} \cdot b^{n-1} \cdot a$, otherwise $\#EQN_S^*$ is **#P**-complete. Thus it is a necessary condition for $\#EQN_S^*$ to be tractable that all elements in $Div(S)$ are regular. □

Now we can prove a dichotomy $\#EQN_S^*$ when $(S, \cdot)$ is a regular semigroup.

Theorem 25 *Let $(S, \cdot)$ be a finite regular semigroup. Then, $\#EQN_S^*$ is tractable if $(S, \cdot)$ is a direct product of an Abelian group and a rectangular band. Otherwise, $\#EQN_S^*$ is **#P**-complete.*

Proof: If the idempotents in $(S, \cdot)$ do not form a rectangular band then by Lemma 22, $\#EQN_S^*$ is **#P**-complete. If the idempotents in $(S, \cdot)$ do form a rectangular band then by Theorem 16, $(S, \cdot)$ is a direct product of a group and a rectangular band. We conclude from Theorem 19 that $\#EQN_S^*$ is tractable if $(S, \cdot)$ is a direct product of an Abelian group and a rectangular band.

Now assume that $(S, \cdot)$ is isomorphic to the direct product of a non-Abelian group by a rectangular band i.e. $(S, \cdot) \cong (G, \cdot_g) \times (RB, \cdot_b)$ where $(G, \cdot_g)$ is a non-Abelian group and $(RB, \cdot_b)$ is a rectangular band. Consider the following relation $R = \{(x, y) | x \cdot y = y \cdot x\}$. We show that R is closed under no Mal'tsev operation. The elements below are chosen as follows, 1 is the identity in $(G, \cdot_g)$, since $(G, \cdot_g)$ is non-Abelian we know that there exists elements a, b in G such that $a \cdot_g b \neq b \cdot_g a$, and c is an arbitrary element in $(RB, \cdot_b)$.

$$
\begin{array}{cll}
 & ((a,c),(1,c)) & \in R \quad (a,c)\cdot(1,c) = (1,c)\cdot(a,c) \\
 & ((1,c),(1,c)) & \in R \quad (1,c)\cdot(1,c) = (1,c)\cdot(1,c) \\
M & ((1,c),(b,c)) & \in R \quad (1,c)\cdot(b,c) = (b,c)\cdot(1,c) \\
\hline
 & ((a,c),(b,c)) & \notin R \quad (a,c)\cdot(b,c) \neq (b,c)\cdot(a,c)
\end{array}
$$

Hence $\#EQN_S^*$ is $\#\mathbf{P}$-complete when $(S, \cdot)$ is isomorphic to the direct product of a non-Abelian group by a rectangular band. □

The following corollary follows directly from Lemma 24 and Theorem 25.

Corollary 26 *If $(S, \cdot)$ is a finite semigroup where all elements have divisors, then $\#EQN_S^*$ is tractable if $(S, \cdot)$ is a direct product of an Abelian group and a rectangular band, and $\#\mathbf{P}$-complete otherwise.*

The following theorem proves a dichotomy for $\#EQN_S^*$ when $(S, \cdot)$ is a semigroup with zero.

Theorem 27 *Let $(S, \cdot)$ be a finite semigroup with zero, then $\#EQN_S^*$ is tractable if for all x, y in S, $x \cdot y = 0$, and $\#\mathbf{P}$-complete otherwise.*

Proof: The tractability part is proved in Theorem 19. We continue by proving the $\#\mathbf{P}$-completeness of $\#EQN_S^*$ when $(S, \cdot)$ is a semigroup with zero and there exist two elements a, b in S such that $a \cdot b \neq 0$. Consider the following relation $R =$

$\{(x, y)|x \cdot y = 0\}$. It is easy to see that R is closed under no Mal'tsev operation.

$$\begin{array}{cccc} & (a,0) & \in R & a \cdot 0 = 0 \\ & (0,0) & \in R & 0 \cdot 0 = 0 \\ M & (0,b) & \in R & 0 \cdot b = 0 \\ \hline & (a,b) & \notin R & a \cdot b \neq 0 \end{array}$$

Hence $\#EQN_S^*$ is $\#\mathbf{P}$-complete when $(S, \cdot)$ is a semigroup with zero and there exist two elements x, y in S such that $x \cdot y \neq 0$. □

It is easy to realize that all the conditions we impose on our semigroups to ensure the tractability (or $\#\mathbf{P}$-completeness) of $\#EQN_S^*$ can be checked in polynomial time (in the size of the semigroups), except maybe for the condition in Theorem 25. To check whether a semigroup, isomorphic to a direct product of a group and a rectangular band, is in fact isomorphic to a direct product of an Abelian group and a rectangular band, we need a way to extract the group part from the semigroup. Choose an idempotent x in $(S, \cdot)$. Then we know that x is of the form $(1, x')$ where 1 is the identity in $(G, \cdot_g)$. Consider the set xSx, elements in this set is of the form $(1, x') \cdot (a, y) \cdot (1, x') = (a, x')$. This implies that $(xSx, \cdot) \cong (G, \cdot_g)$. Now it is easy to check whether $(G, \cdot_g)$ is Abelian.

Acknowledgments

The authors want to thank Pascal Tesson for interesting discussions and remarks on the topics in this paper, Andrei Bulatov for many valuable remarks on an earlier version of this paper and Jorge Almeida for his explanations of some topics in the realm of semigroups.

References

[1] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. 43rd IEEE Symposium on Foundations of Computer Science, FOCS'02*, pages 649–658, 2002.

[2] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *Proc. 44th IEEE Symposium on Foundations of Computer Science, FOCS'03*, 2003.

[3] M. Goldmann and A. Russell. The complexity of solving equations over finite groups. *Inform. and Comput.*, 178(1):253–262, 2002.

[4] J.M. Howie. *Fundamentals of Semigroup Theory*. Claredon Press, Oxford, 1995.

[5] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[6] C. Moore, P. Tesson, and D. Thérien. Satisfiability of systems of equations over finite monoids. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science, MFCS'01*, pages 537–547, 2001.

[7] P. Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, School of Computer Science, McGill University, Montreal, 2003.

Paper II

The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups

Gustav Nordh

Abstract

We study the computational complexity of the isomorphism and equivalence problems on systems of equations over a fixed finite group. We show that the equivalence problem is in **P** if the group is Abelian, and **coNP**-complete if the group is non-Abelian. We prove that if the group is non-Abelian, then the problem of deciding whether two systems of equations over the group are isomorphic is **coNP**-hard. If the group is Abelian, then the isomorphism problem is Graph Isomorphism-hard. Moreover, if we impose the restriction that all equations are of bounded length, then we prove that the isomorphism problem for systems of equations over finite Abelian groups is Graph Isomorphism-complete. Finally we prove that the problem of counting the number of isomorphisms of systems of equations is no harder than deciding whether there exist any isomorphisms at all.

1 Introduction

The computational complexity of deciding whether a system of equations over a fixed finite group is solvable has been studied in the past. Goldmann and Russell [6] proved that the problem is in **P** if the group is Abelian and **NP**-complete otherwise. This line of research continued in [8, 11], where the corresponding problem for finite monoids was given a complete solution. Moreover, some very interesting results in the general case of finite semigroups have been proved by Klíma et al. in [8]. Note that even the restricted problem of determining the computational complexity of solving systems of equations over a fixed regular semigroup is still open. The problem of deciding whether systems of equations over a fixed finite group $(G, \cdot)$ are solvable is denoted by $\textsc{Eqn}^*_G$ in the literature.

The computational complexity of counting solutions to systems of equations over a fixed finite semigroup has been studied in [12], where it is proved that if the semigroup is an Abelian group, then the problem is in **FP**, and if the semigroup is a non-Abelian group, then the problem is #**P**-complete. This problem is denoted $\#\textsc{Eqn}^*_G$.

In this paper we study the computational complexity of deciding whether systems of equations over a fixed finite group are equivalent/isomorphic. More specifically, the equivalence problem is the problem of deciding whether two systems of equations have the same set of solutions and the isomorphism problem is the problem of deciding whether two systems of equations can be made equivalent by permuting the variables in one of them. We also study the problem of counting the number of isomorphisms. These fundamental problems have as far as we know eluded previous investigations from a computational complexity perspective, except for some results on the Boolean constraint equivalence and isomorphism problems, due to Böhler et al. [2, 4], that are also relevant in our setting. More specifically, the equivalence problem for systems of equations over the two element group $\mathbb{Z}_2$

where each equation has a bounded number of variable occurrences is in **P**, and the corresponding isomorphism problem is Graph Isomorphism-complete. Note that in [2, 4] Böhler et al. only study the equivalence and isomorphism problems for Boolean constraints over fixed finite constraint languages, and hence the arity of all constraints involved can be assumed to be bounded by a constant. This motivates us to additionally study the complexity of our problems under the restriction that the number of variables in each equation is bounded by a constant.

The computational complexity of several other isomorphism and equivalence problems have been intensively studied in the past, most notably the Graph Isomorphism problem [7], the formula isomorphism problem [1], and the isomorphism problem for branching programs [13]. Although there are not many results in the literature having direct implications for the equivalence and isomorphism problems for systems of equations, many of the constructions and proof techniques from [1, 2, 4, 10] can be reused.

1.1 Definitions and Summary of Results

A system of equations over a fixed finite group $(G, \cdot)$ is a collection of equations of the form $x_1 \cdot x_2 \cdot \cdots \cdot x_k = x_{k+1} \cdot \cdots \cdot x_n$, where each x_i is either a variable or a constant in G. We will often use G as a shorthand for $(G, \cdot)$ and in the case of Abelian groups we will denote the group operation by $+$. We also assume that all equations have been simplified so that they do not contain subexpressions of the type $a \cdot b$ where a and b are group constants.

The length of an equation is defined to be the number of variable occurrences in it. A system of equations is said to be of length k if k is the length of the longest equation in the system.

Example 1 Let S_1 be the following system of equations over

$\mathbb{Z}_3$:

$$\begin{aligned} x + x + y &= 2 + z \quad (1) \\ x + 2z &= 1 \quad (2) \\ 2 + w + 1 + y &= z + 2 \quad (3) \end{aligned}$$

Equation (1) is of length 4 (x occurs twice, y and z occur once each), (2) is of length 3 ($2z$ amounts to two occurrences of z since $2z$ is just another way of writing $z + z$), and (3) is of length 3. Hence S_1 is of length 4.

Definition 2 Let S be a system of equations on variables X and let π be a permutation of X. By $\pi(S)$ we denote the system of equations that results when we replace each variable x in S by $\pi(x)$.

- EQUIV-EQN$^*_G$ is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over G are equivalent, i.e., whether for every assignment of values in G to the variables in X, S_1 is satisfied if and only if S_2 is satisfied. Note that when we say that S_1 and S_2 are systems of equations on variables X we only mean that X is the union of the variables in S_1 and S_2, hence all variables in X need not to occur both in S_1 and S_2. If S_1 is equivalent to S_2 we denote this by $S_1 \equiv S_2$.

- ISO-EQN$^*_G$ is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over G are isomorphic, i.e., whether there exists a permutation π of the variables in X such that $\pi(S_1) \equiv S_2$. If S_1 is isomorphic to S_2 we denote this by $S_1 \cong S_2$.

- ISO-B-EQN$^*_{G,k}$ and EQUIV-B-EQN$^*_{G,k}$ are the restricted forms of ISO-EQN$^*_G$ and EQUIV-EQN$^*_G$ respectively, where the systems of equations have been bounded to have length at most k.

- #ISO-EQN$^*_G$ is the counting version of ISO-EQN$^*_G$, i.e., the problem of counting the number of permutations π of the variables in X such that $\pi(S_1) \equiv S_2$.

The complexity of EQUIV-EQN$^*_G$ (ISO-EQN$^*_G$, #ISO-EQN$^*_G$) is measured in the size of the systems of equations (the size of G is fixed and does not matter). Note that in all of the problems described above, each group gives rise to a distinct problem. Hence we are trying to classify the complexity of infinite classes of problems. Moreover, the additional restriction that the systems of equations must have length at most k gives rise to infinite classes of problems for each fixed group. This framework makes it possible to give a very fine grained analysis for the complexity of the equivalence and isomorphism problems for systems of equations over finite groups. Also note that all the problems above become trivial if the group G is the (trivial) one element group. Hence when stating our hardness results for problems over Abelian groups we will always assume that the group involved is different from the trivial one.

The main results of the paper can be summarized as follows. We prove that if the group is non-Abelian, then the equivalence problem is **coNP**-complete and if the group is Abelian, then the equivalence problem is in **P**. As for the isomorphism problem we prove that if the group is non-Abelian, then the isomorphism problem is **coNP**-hard and in $\mathbf{\Sigma_2^P}$. If the group is Abelian, then the isomorphism problem is GRAPH ISOMORPHISM-hard and in **NP**. For the restriction of the problems where the length of all equations are bounded by k, we are able to give a very detailed picture for the complexity of ISO-B-EQN$^*_{G,k}$ and EQUIV-B-EQN$^*_{G,k}$. EQUIV-B-EQN$^*_{G,k}$ is **coNP**-complete when G is non-Abelian and $k \geq 3$, otherwise EQUIV-B-EQN$^*_{G,k}$ is in **P**. ISO-B-EQN$^*_{G,k}$ is in **P** when $k \leq 2$ (for all groups G). If $k \geq 3$, then ISO-B-EQN$^*_{G,k}$ is GRAPH ISOMORPHISM-complete when G is Abelian, and **coNP**-hard and in $\mathbf{P}_{\|}^{\mathbf{NP}}$ for all non-Abelian

groups G. Our results on the complexity of EQUIV-B-EQN$^*_{G,k}$ and ISO-B-EQN$^*_{G,k}$ can be seen as a first attempt to extend the complexity classification of the equivalence and isomorphism problems for constraints over fixed finite Boolean constraint languages to larger domains.

For the problem of counting the number of isomorphisms, we give an algorithm that shows that it is no harder to count the number of isomorphisms than to decide whether any isomorphisms exist at all. As a corollary to this algorithm we obtain the result that ISO-EQN$^*_G$ for Abelian groups is powerless as an oracle to **PP**, and that ISO-EQN$^*_G$ for non-Abelian groups is no more powerful than an **NP**-oracle for **PP**. These results indicate that ISO-EQN$^*_G$ for Abelian groups is not **NP**-complete, and that ISO-EQN$^*_G$ for non-Abelian groups is not $\mathbf{\Sigma_2^P}$-complete.

The paper is organized as follows. In Section 2 we prove our results concerning the equivalence problem, Section 3 deals with the isomorphism problem, Section 4 treats the problem of counting the number of isomorphisms, and finally in Section 5 we present our conclusions and some ideas for future research.

2 Equivalence

In this section we investigate the computational complexity of EQUIV-EQN$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are equivalent. We prove that EQUIV-EQN$^*_G$ is **coNP**-complete if G is non-Abelian, and that EQUIV-EQN$^*_G$ is in **P** if G is an Abelian group.

First note that it is easy to see that the equivalence problem is in **coNP**.

Theorem 3 EQUIV-EQN$^*_G$ *is in* **coNP**.

The following theorem states that if it is hard to decide whether systems of equations over a group G are solvable, then

it is also hard to decide whether systems of equations over the same group are equivalent.

Theorem 4 *If* EQN^*_G *is* **NP***-complete, then* EQUIV-EQN^*_G *is* **coNP***-complete.*

Proof: If EQN^*_G is **NP**-complete, it follows that it is **coNP**-complete to decide whether a system of equations over G is insoluble. Since a system of equations is insoluble if and only if it is equivalent to an insoluble system of equations (for example a system of equations containing the equation $a = b$ where a and b are distinct constants in G), it follows that EQUIV-EQN^*_G is **coNP**-complete. □

The previous theorem and the fact that EQN^*_G is **NP**-complete when G is a non-Abelian group [6] immediately implies the following corollary.

Corollary 5 EQUIV-EQN^*_G *is* **coNP***-complete when* G *is a non-Abelian group.*

Next we prove that if it is easy to count the number of solutions to systems of equations over a group G, then it is also easy to decide whether systems of equations over the same group are equivalent.

Theorem 6 EQUIV-EQN^*_G *is in* **P** *if* $\#\text{EQN}^*_G$ *is in* **FP***.*

Proof: Let S_1 and S_2 be two systems of equations. Count the number of solutions to S_1 and S_2: if they have different number of solutions they are not equivalent. Thus we can assume that S_1 and S_2 have the same number of solutions. Count the number of solutions to the system of equations consisting of the union of S_1 and S_2. If the number of solutions to this system of equations equals the number of solutions to S_1, then we know that S_1 and S_2 have the same set of solutions and hence S_1 is equivalent to

S_2, otherwise S_1 and S_2 have different sets of solutions and thus are inequivalent. □

The following corollary follows directly from Theorem 6 and the fact that $\#\text{EQN}^*_G$ is in **FP** when G is an Abelian group [12].

Corollary 7 EQUIV-EQN$^*_G$ *is in* **P** *when* G *is an Abelian group.*

It should be clear that Theorems 4 and 6 also hold when generalized to systems of equations over a fixed finite semigroup G. Hence interesting results on the complexity of EQUIV-EQN$^*_G$, where G is a finite semigroup, can be deduced from the results for $\#\text{EQN}^*_G$ and EQN$^*_G$ proved in [8, 11, 12]. We collect these results in the following corollary.

Corollary 8 EQUIV-EQN$^*_G$ *is* **coNP***-complete when*

- G *is a monoid but is not commutative [11].*
- G *is a monoid but is not a union of Abelian groups [11].*
- G *is a regular semigroup but is not a strong normal band of Abelian groups [8].*

EQUIV-EQN$^*_G$ *is in* **P** *when*

- G *is a direct product of an Abelian group and a rectangular band [12],*
- G *is a semigroup with zero, such that for all elements* x, y, $x \cdot y = 0$ *[12].*

2.1 Bounded Length

If we restrict the problem and require that all equations are of bounded length, then we can prove an even tighter correspondence between the complexity of the equivalence problem and the solvability problem.

Theorem 9 *If* $\text{EQN}^*_{G,k}$ *is in* **P**, *then* EQUIV-B-$\text{EQN}^*_{G,k}$ *is in* **P**, *and if* $\text{EQN}^*_{G,k}$ *is* **NP**-*complete, then* EQUIV-B-$\text{EQN}^*_{G,k}$ *is* **coNP**-*complete (where* $\text{EQN}^*_{G,k}$ *denotes the solvability problem restricted to systems of equations of length at most* k*).*

Proof: For the tractability part we give a polynomial-time Turing reduction from EQUIV-B-$\text{EQN}^*_{G,k}$ to $\text{EQN}^*_{G,k}$. The reduction is the same as the one in the proof of [2, Lemma 7]. Let $S \to E$ denote that the equation E can be inferred from the system of equations S, i.e., there exists no assignment to the variables in S such that S is satisfied and E is not satisfied. Let S_1 and S_2 be two systems of equations where all equations are of length at most k for some constant k. S_1 and S_2 are equivalent if and only if $S_1 \to E$ for every equation E in S_2, and $S_2 \to E'$ for every equation E' in S_1. Note that it is easy to check whether $S_1 \to E$ with at most $|G|^k$ queries to $\text{EQN}^*_{G,k}$. For every assignment to the variables in E that does not satisfy E, we check whether this partial assignment of the variables can be extended to a satisfying assignment for S_1. Of course, $S_1 \to E$ if and only if none of these assignments can be extended to a satisfying assignment to S_1.

The **coNP**-completeness part follows from Theorem 4. □

Note that the preceding theorem also holds when generalized to systems of equations over a fixed finite semigroup G.

It is easy to see that equations of length more than 3 can always be split into equations of length 3 in a solvability preserving manner by introducing new variables. Hence, EQN^*_G is polynomial-time reducible to the restricted form where each equation have at most 3 variable occurrences, i.e., EQN^*_G is polynomial-time reducible to $\text{EQN}^*_{G,3}$. This observation together with the preceding theorem and the fact that EQN^*_G is **NP**-complete for non-Abelian groups [6], gives us the following result.

Corollary 10 EQUIV-B-$\text{EQN}^*_{G,k}$ *is* **coNP**-*complete for all non-Abelian groups* G *when* $k \geq 3$.

Next we prove that the equivalence problem for systems of equations of length at most 2 is in **P** for all groups G.

Theorem 11 EQUIV-B-EQN$^*_{G,k}$ *is in* **P** *for all groups* G *when* $k \leq 2$.

Proof: We prove that EQN$^*_{G,2}$ is in **P**. Hence, the desired result then follows from Theorem 9. First of all, equations of the form $a = a$, where a is a constant from G, are redundant and can be removed, and if there exists an equation in the system of equations of the form $d = e$, where d and e are distinct constants from G, then clearly the system has no solution. Hence we can assume that all equations contain at least one variable.

Represent the system of equations S as a graph G with one vertex for each variable in S and an edge between any pair of vertices whose corresponding variables occur in the same equation in S. It should be clear that the connected components in G corresponds to independent subsystems of S. Obviously S has a solution if and only if each of these independent subsystems of equations have a solution. Hence each of these independent subsystems can be tested for solvability in isolation.

Let S' be such a subsystem corresponding to a connected component in G. Choose an arbitrary variable in S', say x. If we assign a value $a \in G$ to x, then each equation where x occurred will now be either in the form $b \cdot y = c$ or $d = e$ where b, c, d, and e are (not necessarily distinct) constants from G and y is a variable. Note that since we are working over a group, equations of the form $b \cdot y = c$ always have a unique solution. Hence y will be forced to take a value from G and the propagation can continue in the same manner until all variables in S' have been assigned. Now if there is an equation in the resulting system of equations of the form $d = e$, where d and e are distinct constants from G, then clearly there exists no solution of S' such that a is assigned to x. Otherwise S' has a solution (where a is assigned to x). Obviously S' has a solution if and only if for at least one

element $a \in G$, there exists a solution where a is assigned to x. □

3 Isomorphism

In this section we investigate the computational complexity of ISO-EQN$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are isomorphic. We prove that ISO-EQN$^*_G$ is **coNP**-hard if G is non-Abelian, and that ISO-EQN$^*_G$ is GRAPH ISOMORPHISM-hard if G is an Abelian group. If we restrict the problem and demand that all equations are of bounded length, then ISO-EQN$^*_G$ in the Abelian case becomes GRAPH ISOMORPHISM-complete, and ISO-EQN$^*_G$ in the non-Abelian case is in $\mathbf{P}_{||}^{\mathbf{NP}}$, i.e., the class of problems solvable in polynomial time with parallel access to an **NP**-oracle.

We begin by giving upper bounds for the complexity of the isomorphism problem.

Theorem 12 ISO-EQN$^*_G$ *is in* **NP** *when* G *is an Abelian group and* ISO-EQN$^*_G$ *is in* $\mathbf{\Sigma_2^P}$ *when* G *is a non-Abelian group.*

Proof: The **NP** upper bound for ISO-EQN$^*_G$ when G is an Abelian group follows from the results in the previous section on the equivalence problem. We know from Corollary 7 that $\pi(S_1) \equiv S_2$ can be decided in polynomial time when G is an Abelian group, hence ISO-EQN$^*_G$ is in **NP** when G is an Abelian group.

For the $\mathbf{\Sigma_2^P}$ upper bound we nondeterministically choose a permutation π of the variables in X and use an **NP**-oracle to check whether $\pi(S_1) \equiv S_2$. Hence ISO-EQN$^*_G$ is in $\mathbf{NP^{NP}} = \Sigma_2^{\mathrm{P}}$. □

The following theorem states that if it is hard to decide whether systems of equations over a group G are solvable, then

it is also hard to decide whether systems of equations over the same group are isomorphic.

Theorem 13 *If* EQN^*_G *is* **NP***-complete then* ISO-EQN^*_G *is* **coNP***-hard.*

Proof: If EQN^*_G is **NP**-complete, then it follows that it is **coNP**-hard to decide whether a system of equations over G is insoluble. Since a system of equations is insoluble if and only if it is isomorphic to an insoluble system of equations, e.g., a system of equations containing the equation $a = b$ where a and b are distinct constants in G, it follows that ISO-EQN^*_G is **coNP**-hard. □

The previous theorem and the fact that EQN^*_G is **NP**-complete when G is a non-Abelian group [6] immediately implies the following corollary.

Corollary 14 ISO-EQN^*_G *is* **coNP***-hard when G is a non-Abelian group.*

The following theorem indicates that ISO-EQN^*_G for Abelian groups perhaps is not in **P**, or at least that it is hard to prove that ISO-EQN^*_G is in **P** for Abelian groups.

Theorem 15 *Let G be a finite Abelian group (different from the trivial one), then* GRAPH ISOMORPHISM *is polynomial-time many-one reducible to* ISO-EQN^*_G.

Proof: We first show the result for the case where G is a cyclic group (that is groups of the form $\mathbb{Z}_m$, i.e., the integers modulo m under addition), then we show how to extend this result to all finite Abelian groups.

Since the group is Abelian, we denote the group operation by $+$. As usual we denote the elements of $\mathbb{Z}_m$ as $\{0, 1, 2 \ldots, m-1\}$, where 0 is the identity element and 1 is the group generator. It

is known that Graph Isomorphism is polynomial-time equivalent to the Graph Isomorphism problem restricted to bipartite graphs. To reduce the general Graph Isomorphism problem to the Graph Isomorphism problem restricted to bipartite graphs; we simply split each edge by introducing a new vertex as a middle point, and attach a sufficiently large even cycle to each of these new vertices (forcing middle points to be mapped to middle points). The resulting bipartite graphs are isomorphic if and only if the original graphs are isomorphic. To simplify the proof we assume from now on that all graphs are bipartite and contain no isolated vertices.

Let T be the following transformation from a graph $H = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$, to a system of equations $T(H)$ over G:

$$T(H) = \{x_i + x_j + y_k = 1 \mid e_k = \{i, j\}\} \cup \{x_i + z_i + z'_i = 1 \mid i \in V\}.$$

The variables x_i and y_k correspond to vertex i and edge e_k respectively. The z and z' variables are used to distinguish x variables from y variables. This transformation is based on the construction in the proof of [2, Theorem 25].

Let $T(H) \rightarrow a + b + c = 1$ (where a, b, c are variables occurring in $T(H)$) denote that the equation $a + b + c = 1$ can be inferred from the system of equations $T(H)$, i.e., there exists no assignment to the variables in $T(H)$ such that $T(H)$ is satisfied and $a + b + c = 1$ is not satisfied. Note that since G is Abelian, we (for example) consider $a + b + c = 1$ as being identical to $b + a + c = 1$. A maximum set of equations is a set of equations S such that if $S \rightarrow E$ then $E \in S$.

The proof of the theorem relies on the following lemma, which shows that $T(H)$ is a maximum set of equations of the type $a + b + c = 1$, where a, b, c are distinct variables. More precisely, the lemma shows that there exists no equation E of length 3 having exactly 3 distinct variables (and no constants) on the left-hand side and only the constant 1 on the right-hand side, such that $T(H) \rightarrow E$ and $E \notin T(H)$.

The importance of the property that $T(H)$ is a maximum set of equations (of type $a+b+c=1$), lies in the observation that two maximum sets of equations are equivalent if and only if the sets are equal. We will see later that $T(H)$ would not necessarily be a maximum set of equations of type $a+b+c=1$ for non-bipartite graphs H.

Lemma 16 *The following holds for every triple a, b, c of distinct variables in $T(H)$: If $T(H) \rightarrow a+b+c=1$, then $a+b+c=1 \in T(H)$.*

Proof: The proof is a minor modification of the proof of Lemma 27 in [3] and consists of a careful case analysis. Let $X = \{x_i \mid i \in V\}$, $Y = \{y_i \mid e_i \in E\}$, and $Z = \{z_i, z'_i \mid i \in V\}$. To give the proof more structure we assume that $a \leq b \leq c$, where $\leq$ is the following order on $X \cup Y \cup Z$:

$$x_1 < \cdots < x_n < y_1 < \cdots < y_m < z_1 < \cdots < z_n < z'_1 < \cdots < z'_n.$$

With the aim of reaching a contradiction we assume that there exists a triple a, b, c of distinct variables in $T(H)$ such that $T(H) \rightarrow a+b+c=1$ and $a+b+c=1 \notin T(H)$. In each of the 10 possible cases that arise we show that there exists an assignment to the variables such that $T(H)$ is satisfied but $a+b+c=1$ is not satisfied.

1. If a, b, and c are in X, then set all variables in X to 0, and all variables in Y to 1. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a+b+c=1$.

2. If a, b are in X, and c is in Y, suppose that $c = y_k$ and let $e_k = \{i, j\}$. By the assumption that $a+b+c=1$ is not in $T(H)$, at least one of a and b is not in $\{x_i, x_j\}$. Without loss of generality, let $a \notin \{x_i, x_j\}$. Set a to 1 and set $X \setminus \{a\}$ to 0. Note that such an assignment will force $y_k = c$ to be set to 1. This assignment can be extended to an assignment satisfying $T(H)$ but not $a+b+c=1$.

3. If a, b are in X, and c is in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G| - 1$ (i.e., $m - 1$ if the group is $\mathbb{Z}_m$) to every variable in Y. This assignment satisfies $T(H)$ but not $a + b + c = 1$.

4. If a is in X, and b, c are in Y, then set all variables in X to 0, and all variables in Y to 1. This can be extended to a satisfying assignment to $T(H)$ but not to $a + b + c = 1$.

5. If a is in X, b is in Y, and c is in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G| - 1$ to every variable in Y. This assignment satisfies $T(H)$ but not $a + b + c = 1$.

6. If a is in X, and b, c are in Z, then consider the assignment that assigns 0 to a, b, and c. We know by assumption that $a + b + c = 1 \notin T(H)$, hence this assignment can be extended to an assignment on $X \cup Z$ satisfying all equations in $T(H)$ of the form $x_i + z_i + z_i' = 1$. Now since all remaining equations in $T(H)$ contain a variable from Y and no variable in Y occurs in more than one equation, it is easy to see that this assignment can be extended to an assignment that satisfies $T(H)$ but not $a + b + c = 1$.

7. If a, b, and c are in Y, then consider the assignment that assigns 0 to all variables in Y. This assignment can be extended to a satisfying assignment to $T(H)$ but not to $a + b + c = 1$. Note that the fact that the assignment can be extended follows from the bipartiteness of the graph. A two coloring of the graph gives an assignment that assigns 1 to exactly one of $\{x_i, x_j\}$ for each $e_k = \{i, j\}$ and 0 to all other elements of X.

8. If a, b are in Y, and c is in Z, then consider the assignment that assigns 2 to c (0 to c if the group is $\mathbb{Z}_2$), 1 to every

variable in X, and $|G| - 1$ to every variable in Y. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a + b + c = 1$.

9. If a is in Y, and b, c is in Z, let $a = e_k$ where $e_k = \{i, j\}$. If $\{b, c\} = \{z_l, z'_l\}$ for some l, then set x_l to 1. In all cases, set b and c to 0, and for all $e_k = \{i, j\} \in Y$ set exactly one of $\{x_i, x_j\}$ to 1 (this could be x_l). Set all other elements of X to 0, and all elements of Y to 0. This assignment can be extended to an assignment that satisfies $T(H)$ but not $a + b + c = 1$. Note again that the existence of such an assignment to the variables in X follows from the fact that the graph is bipartite.

10. If a, b, and c are in Z, then consider the assignment that assigns 0 to every variable in Z, 1 to every variable in X, and $|G|-1$ to every variable in Y. This assignment satisfies $T(H)$ but not $a + b + c = 1$.

Hence if $T(H) \rightarrow a + b + c = 1$, then $a + b + c = 1 \in T(H)$. □

The importance of the fact that $T(H)$ is a maximum set of equations of the form $a + b + c = 1$ lies in the observation that $T(H_1) \equiv T(H_2)$ if and only if $T(H_1) = T(H_2)$. Note again that since G is Abelian we consider $a+b+c = 1$ as being identical to $b + c + a = 1$, and so on. Hence $T(H_1)$ is isomorphic to $T(H_2)$ if and only if there exists a permutation ρ of the variables in $T(H_1) \cup T(H_2)$ such that $\rho(T(H_1)) = T(H_2)$.

Let $H_1 = (V, E)$ and $H_2 = (V, E')$ be two bipartite graphs, with an equal number of vertices and edges, where $V = \{1, 2, \ldots, n\}$, $E = \{e_1, e_2, \ldots, e_m\}$, and $E' = \{e'_1, e'_2, \ldots, e'_m\}$. We will prove that there exists an isomorphism π from H_1 to H_2 if and only if there exists an isomorphism ρ from $T(H_1)$ to $T(H_2)$.

Given an isomorphism π from H_1 to H_2, we construct an isomorphism ρ from $T(H_1)$ to $T(H_2)$ as follows. For each i in V, let $\rho(x_i) = x_{\pi(i)}$, $\rho(z_i) = z_{\pi(i)}$, and $\rho(z'_i) = z'_{\pi(i)}$. For each $e_k = \{i, j\}$ in E, let $\rho(y_k) = y_l$ where $e'_l = \{\pi(i), \pi(j)\}$.

To prove the remaining implication, we first need to make some observations about the different types of variables. If we are given an isomorphism ρ from $T(H_1)$ to $T(H_2)$, then as we have already observed $\rho(T(H_1)) = T(H_2)$. Now consider the following distinguishing properties of the variables.

- Variables from X are the only variables that occur at least twice. So, ρ will be a bijection on X.
- Variables from Z are those variables that occur exactly once and that occur together with another variable that occurs exactly once. Hence, ρ will be a bijection also on Z.
- Since ρ is a bijection on both X and Z, it will of course have to be a bijection also on the variables in Y.

So, given ρ we let $\pi(i) = j$ if and only if $\rho(x_i) = x_j$. Now we must prove that $\{i, j\} \in E$ if and only if $\{\pi(i), \pi(j)\} \in E'$. If $e_k = \{i, j\}$, then we know that $x_i + x_j + y_k = 1$ is in $T(H_1)$. Hence, $\rho(x_i) + \rho(x_j) + \rho(y_k) = 1$ is in $T(H_2)$, i.e., $x_{\pi(i)} + x_{\pi(j)} + \rho(y_k) = 1$ is in $T(H_2)$. Thus we must have $\rho(y_k) = y_l$, where $e'_l = \{\pi(i), \pi(j)\}$, so $\{\pi(i), \pi(j)\}$ is an edge in E'. On the other hand, if $\{\pi(i), \pi(j)\}$ is an edge in E', then $x_{\pi(i)} + x_{\pi(j)} + y_l = 1$ is in $T(H_2)$ (where $e'_l = \{\pi(i), \pi(j)\}$). This gives us that $x_i + x_j + \rho^{-1}(y_l) = 1$ must be in $T(H_1)$. Since $\rho^{-1}(y_l) = y_k$, where $e_k = \{i, j\}$, we conclude that $\{i, j\} \in E$. Thus π is an isomorphism from H_1 to H_2 if and only if ρ is an isomorphism from $T(H_1)$ to $T(H_2)$. This completes our proof that Graph Isomorphism is polynomial-time many-one reducible to Iso-Eqn$^*_G$ where G is a finite cyclic group.

By using the fundamental theorem of finitely generated Abelian groups it is possible to extended this result to all finite Abelian groups. Note that all finite Abelian groups are of course finitely generated.

Lemma 17 (See, [9]) *Every finite Abelian group G is isomorphic to a direct product of cyclic groups in the form $\mathbb{Z}_{p_1^{r_1}} \times \mathbb{Z}_{p_2^{r_2}} \times$*

$\cdots \times \mathbb{Z}_{p_n^{r_n}}$, *where the* p_i*'s are (not necessarily distinct) primes, and* $|G| = p_1^{r_1} \cdot \cdots \cdot p_n^{r_n}$.

Thus given a system of equations S_1 over a finite Abelian group G, we can use the fundamental theorem of finitely generated Abelian groups to view it as n independent systems of equations, each over a cyclic group $\mathbb{Z}_{p_i^{r_i}}$. Given a group $G \cong \mathbb{Z}_{p_1^{r_1}} \times \mathbb{Z}_{p_2^{r_2}} \times \cdots \times \mathbb{Z}_{p_n^{r_n}}$, denote the element $1_{p_1^{r_1}} \times 1_{p_2^{r_2}} \times \cdots \times 1_{p_n^{r_n}}$ in G by 1_G, where $1_{p_i^{r_i}}$ denotes the element 1 in $\mathbb{Z}_{p_i^{r_i}}$. Let $T(H_1)'$ and $T(H_2)'$ denote the systems of equations over G obtained from $T(H_1)$ and $T(H_2)$ by replacing all occurrences of 1 by 1_G.

We have proved above that there exists an isomorphism π from H_1 to H_2 if and only if there exists an isomorphism ρ from $T(H_1)$ to $T(H_2)$ regardless of which finite cyclic group the systems of equations are defined over. Moreover, if ρ is an isomorphism from $T(H_1)$ to $T(H_2)$ then $\rho(T(H_1)) = T(H_2)$, and of course $\rho(T(H_1)') = T(H_2)'$ (i.e., $T(H_1)' \cong T(H_2)'$). Conversely, if there exists an isomorphism ρ' from $T(H_1)'$ to $T(H_2)'$ (i.e., $\rho'(T(H_1)') \equiv T(H_2)'$), then $\rho'(T(H_1)) \equiv T(H_2)$ (for all $\mathbb{Z}_{p_i^{r_i}} \in \{\mathbb{Z}_{p_1^{r_1}}, \mathbb{Z}_{p_2^{r_2}}, \ldots, \mathbb{Z}_{p_n^{r_n}}\}$, and hence all finite cyclic groups).

Hence given a finite Abelian group G and two graphs H_1 and H_2, then H_1 is isomorphic to H_2 if and only if the two systems of equations $T(H_1)'$ and $T(H_2)'$ over G are isomorphic. □

Despite intensive research Graph Isomorphism is not known to be in **coNP**. Hence, in light of the previous theorem, it seems hard to prove that Iso-Eqn$^*_G$ is in **coNP** when G is an Abelian group.

3.1 Bounded Length

If we restrict the isomorphism problem and require that all equations are of bounded length, then we can prove an even tighter correspondence between the complexity of Iso-Eqn$^*_{G,k}$ and Graph Isomorphism.

Theorem 18 ISO-B-EQN$^*_{G,k}$ *is* GRAPH ISOMORPHISM*-complete under polynomial-time many-one reductions when* G *is an Abelian group (different from the trivial one) and* $k \geq 3$.

Proof: It follows from the proof of Theorem 15 that GRAPH ISOMORPHISM is polynomial-time many-one reducible to ISO-B-EQN$^*_{G,k}$ for Abelian groups G and $k \geq 3$. Hence what remains is to give a polynomial-time many-one reduction from ISO-B-EQN$^*_{G,k}$ to GRAPH ISOMORPHISM. We will actually give a reduction from ISO-B-EQN$^*_{G,k}$ to the GRAPH ISOMORPHISM-complete problem vertex colored graph isomorphism (VCGI) [7]. Our reduction is a minor modification of the reduction in the proof of Claim 22 from Böhler et al. [2] and our proof closely follows theirs.

Definition 19 VCGI is the problem of, given two vertex-colored graphs $H_1 = (V_1, E_1, \chi_1)$, $H_2 = (V_2, E_2, \chi_2)$, with χ_1, χ_2: $V \to \mathbb{N}$, to determine whether there exists an isomorphism from H_1 to H_2 that preserves colors, i.e., whether there exists a bijection $\pi : V_1 \to V_2$ such that $\{v, v'\} \in E_1$ if and only if $\{\pi(v), \pi(v')\} \in E_2$, and $\chi_1(v) = \chi_2(\pi(v))$ for all $v \in V_1$.

Let S_1 and S_2 be two systems of equations of bounded length over an Abelian group G on variables X. The case where at least one of S_1 and S_2 are not solvable is trivially reducible to GRAPH ISOMORPHISM. Hence we can assume that both S_1 and S_2 are solvable. We will first bring S_1 and S_2 into normal form. We choose an approach similar to that in Theorem 15 and reduce S_1 and S_2 to the maximum set of equations of length at most k that can be inferred from S_1 and S_2, respectively. Since both S_1 and S_2 are solvable we can assume that all equations only containing constants have been removed. Moreover, we assume that all equations have been reduced in such a way that they contain no constants from G except for in the right-most position on the right-hand side (if an equation should lack a constant we add 0 to it, where 0 denotes the identity in G). Since the group is Abelian

this reduction is trivial and the resulting system of equations is of course equivalent to the original one. We call equations of this particular form reduced equations.

Let $\langle S_1 \rangle$ denote the set of all reduced equations E of length at most k such that all of E's variables occur in X and $S_1 \rightarrow E$. It should be clear that S_1 is equivalent to $\langle S_1 \rangle$, since $S_1 \subseteq \langle S_1 \rangle$ and $S_1 \rightarrow \langle S_1 \rangle$. We define $\langle S_2 \rangle$ analogously. Note that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time (in $|S_1| + |S_2|$), since there exist at most $O(|X|^k)$ reduced equations E of length at most k such that all of E's variables occur in X, and since G is Abelian we can use the polynomial-time algorithm for EQN^*_G to decide whether $S \rightarrow E$. Note that $S \rightarrow E$ if and only if for every assignment to the variables in E such that E is not satisfied, this assignment applied to S makes S insoluble. There is at most $O(|G|^k)$ possible assignments of the variables in E. Testing whether or not such an assignment makes S insoluble can be done in polynomial time (by the result in [6]). So, it follows that $S \rightarrow E$ can be decided in polynomial time. It should be clear that if $\langle S_1 \rangle$ is equivalent to $\langle S_2 \rangle$ then $\langle S_1 \rangle = \langle S_2 \rangle$, hence if π is a permutation of the variables in X such that $\pi(\langle S_1 \rangle) \equiv \langle S_2 \rangle$, then $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$.

Now we proceed and reduce $\langle S_1 \rangle$ and $\langle S_2 \rangle$ to vertex-colored graphs H_1 and H_2 such that $\langle S_1 \rangle \cong \langle S_2 \rangle$ if and only if $(H_1, H_2) \in VCGI$.

Let $\langle S_1 \rangle$ consist of the equations $A_1, \ldots, A_m$, i.e.,

$$\begin{array}{rcll} x_{11} + x_{12} + \cdots + x_{1j_1} & = & x_{1j_1+1} + \cdots + x_{1k_1} + c_1 & (A_1) \\ x_{21} + x_{22} + \cdots + x_{2j_2} & = & x_{2j_2+1} + \cdots + x_{2k_2} + c_2 & (A_2) \\ & \vdots & & \\ x_{m1} + x_{m2} + \cdots + x_{mj_m} & = & x_{mj_m+1} + \cdots + x_{mk_m} + c_m & (A_m) \end{array}$$

Let $T(\langle S_1 \rangle) = H_1 = (V, E, \chi)$ be the following vertex-colored graph:

- $V = \{g_0, g_1, \ldots, g_{|G|-1}\} \cup \{x \mid x \in X\} \cup \{a_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq k_i + 1\} \cup \{A_i \mid 1 \leq i \leq m\}$. Hence, the set of

vertices corresponds to the elements in G, the variables in X, the elements in the equations in $\langle S_1 \rangle$, and the equations in $\langle S_1 \rangle$.

- $E = \{\{x, a_{ij}\} \mid x = a_{ij}\} \cup \{\{c, a_{ij}\} \mid c = a_{ij}\} \cup \{\{a_{ij}, A_i\} \mid 1 \leq i \leq m, 1 \leq j \leq k_i + 1\}$. Thus, the edges indicate which variables and constants that occur in a given equation.

- The vertex coloring χ is used to distinguish the different categories of vertices. Of course, we must allow any permutation of the variables, so all vertices corresponding to variables in X are assigned the same color. We also need to allow the permutation of vertices corresponding to equations, and because the group is Abelian we allow the permutation of vertices corresponding to elements within the same side of each equation.

  - $\chi(g_i) = i$, i.e., if $c_t = g_i$ then $\chi(c_t) = i$,
  - $\chi(x) = |G|$ for all $x \in X$,
  - $\chi(A_r) = |G| + 1$,
  - $\chi(a_{ij}) = |G| + 2$ if $j \leq j_i$,
  - $\chi(a_{ij}) = |G| + 3$ if $k_i \geq j > j_i$,
  - $\chi(a_{ij}) = |G| + 4$ if $j = k_i + 1$.

Define $T(\langle S_2 \rangle) = H_2$ in the analogous way.

Given a permutation π of X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$, then it is easily realized that $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$. For the converse, given a permutation π on $T(\langle S_1 \rangle)$ witnessing the fact that $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$, then we have $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. Just note that any permutation of vertices corresponding to equations forces a permutation of the vertices corresponding to elements in the equations. Thus if $\pi(A_i) = A_j$ then we must have $\pi(a_{it}) = a_{js}$, for all $1 \leq t \leq k_i + 1$ and some $1 \leq s \leq k_j + 1$. Also note that because of the coloring, π is the identity mapping on

vertices corresponding to constants. Hence, the only thing left is the permutation of vertices corresponding to variables in X. Thus we have $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. □

Using the same idea as in the proof of the preceding theorem we can prove a tighter upper bound for ISO-B-EQN$^*_{G,k}$, compared to the trivial $\mathbf{\Sigma_2^P}$ upper bound for non-Abelian groups from Theorem 12. More specifically we prove that ISO-B-EQN$^*_{G,k}$ for non-Abelian groups is in $\mathbf{P}_{||}^{\mathbf{NP}}$, the class of problems solvable in polynomial time with parallel (i.e., truth-table) access to an **NP**-oracle. This result is analogous to Corollary 23 in [2] and the proof is the same.

Theorem 20 ISO-B-EQN$^*_{G,k}$ *is in* $\mathbf{P}_{||}^{\mathbf{NP}}$.

Proof: Let S_1 and S_2 be two systems of equations on variables X over a non-Abelian group G. We use the same normal form as in the proof of the preceding theorem, i.e., we compute $\langle S_1 \rangle$ and $\langle S_2 \rangle$. Following the proof of the preceding theorem it is easy to verify that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time with parallel access to an **NP**-oracle. Now the existence of a permutation π of the variables in X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$ can be determined with one query to an **NP**-oracle. Hence, it can be decided in polynomial time with two rounds of parallel queries to **NP** whether S_1 and S_2 are isomorphic, and thus by the results in [5] it follows that ISO-B-EQN$^*_{G,k}$ is in $\mathbf{P}_{||}^{\mathbf{NP}}$. □

Böhler et al. [4] have proved that the isomorphism problem for systems of equations having length at most 2 over the group $\mathbb{Z}_2$ can be solved in polynomial time. We prove that the isomorphism problem for systems of equations of length at most 2 is in **P** for all groups.

Theorem 21 ISO-EQN$^*_{G,k}$ *is in* **P** *for all groups* G *when* $k \leq 2$.

Proof: We know from the proof of Theorem 11 that we can check in polynomial time whether systems of equations of length at most 2 are solvable. If neither S_1 nor S_2 are solvable, then they are of course isomorphic, and if just one of S_1 and S_2 are solvable, then they are not isomorphic. Hence we can assume that both S_1 and S_2 are solvable.

Remember from the proof of Theorem 11 that we can represent the system of equations S_i, $i \in \{1, 2\}$, as a graph G_i with one vertex for each variable in S_i and an edge between any pair of vertices whose corresponding variables occur in the same equation in S_i. It should be clear that the connected components in G_i correspond to independent subsystems of S_i. Note that since S_i has a solution, so do each of these independent subsystems of S_i. In fact we know, from the proof of Theorem 11, how to compute in polynomial time all the solutions of each of these subsystems and that they have at most $|G|$ solutions each. Moreover, for any two different solutions α and β of such a subsystem, $\alpha(x) \neq \beta(x)$ hold for all variables x in the subsystem. As we will see, this is the key of the proof.

We begin by treating subsystems having a unique solution. Let X_1 and X_2 be the set of all variables occurring in subsystems, of S_1 and S_2 respectively, having a unique solution. Let α_1 and α_2 be the partial solutions, to S_1 and S_2 respectively, that are induced by the subsystems having a unique solution. It is easy to check whether there exists a bijective mapping π_{FIXED} from X_1 to X_2 such that for all variables $x \in X_1$, if $\pi_{FIXED}(x) = x'$ then $\alpha_1(x) = \alpha_2(x')$. If no such mapping π_{FIXED} exists, then clearly $S_1 \ncong S_2$. On the other hand, if $S_1 \cong S_2$, then there exists an isomorphism π from S_1 to S_2 such that π and π_{FIXED} agree on the variables in X_1. Hence, we can assume that each independent subsystem of S_1 and S_2 have at least 2 solutions.

Next we show how to check in polynomial time whether two independent subsystems, IS_1 and IS_2 corresponding to connected components of G_1 and G_2 respectively, are isomorphic. Let Y_1

and Y_2 be the set of variables that occur in IS_1 and IS_2 respectively. If IS_1 and IS_2 have a different number of solutions or $|Y_1| \neq |Y_2|$, then $IS_1 \ncong IS_2$. So we assume that IS_1 and IS_2 have the same number of variables and solutions. Represent the m ($m \leq |G|$) solutions of IS_1 as a $m \times |Y_1|$ matrix M_1 where the rows of M_1 are just the solutions of IS_1. Define the matrix M_2, corresponding to the solutions of IS_2, in the analogous way. Now, if there exists a permutation of the rows and columns of M_1 (yielding a matrix M_1') such that $M_1' = M_2$, then clearly $IS_1 \cong IS_2$. If no such permutation exists, then it is easy to see that $IS_1 \ncong IS_2$. The existence of such a permutation can be determined in polynomial time since the number of permutations of the rows in M_1 is bounded by the constant $|G|!$. After each such permutation it can be checked whether the resulting matrix M_1'' admits a permutation of the columns, yielding a matrix M_1' such that $M_1' = M_2$ (e.g., by sorting the columns in M_1'' and M_2 in lexicographical order and checking for identity).

Now, we prove that if there exists an independent subsystem IS_1 of S_1 (corresponding to a connected component of G_1) such that IS_1 is not isomorphic to any of the independent subsystems of S_2 (corresponding to connected components of G_2), then $S_1 \ncong S_2$. Assume the contrary, i.e., that IS_1 is not isomorphic to any of the independent subsystems of S_2 but there exists a permutation π such that $\pi(S_1) \equiv S_2$. Two cases emerge, either the variables in IS_1 are mapped, by π, to the variables occurring in a single independent subsystem of S_2, or they are mapped to variables from more than one independent subsystem in S_2.

If the variables in IS_1 are mapped, by π, to the variables in a single subsystem, IS_2, of S_2, then there must be a variable x in IS_2 that does not occur in $\pi(IS_1)$ (otherwise we get a contradiction with the assumption that $IS_1 \ncong IS_2$ and $\pi(S_1) \equiv S_2$). Now, if we assign a value a to a variable y in $\pi(IS_1)$, then in IS_2, x will be forced to take a value b. By the assumption that $\pi(S_1) \equiv S_2$, we have that x must occur in a subsystem of S_1

(having at least two solutions) that is independent of IS_1. So $\pi(S_1)$ has a solution where a is assigned to y and c is assigned to x, where $c \neq b$, again contradicting $\pi(S_1) \equiv S_2$.

In the second case $\pi(IS_1)$ contains variables from two different independent subsystems of S_1. Any solution of $\pi(IS_1)$ forces a solution to these independent subsystems of S_2. But again since these subsystems of S_2 have more than one solution each, there exists a solution to S_2 not satisfying $\pi(S_1)$ (contradicting $\pi(S_1) \equiv S_2$).

The outline of the polynomial-time algorithm for solving Iso-Eqn$^*_{G,2}$ should now be clear. For each independent subsystem, IS_1 of S_1 corresponding to a connected component in G_1, check whether there exists an independent subsystem, IS_2 of S_2 corresponding to a connected component in G_2, such that $IS_1 \cong IS_2$. If no such subsystem IS_2 exists, then we conclude that $S_1 \ncong S_2$. Otherwise remove IS_1 and IS_2 from S_1 and S_2 respectively, and continue with the next independent subsystem of S_1. When all independent subsystems of S_1 corresponding to connected components of G_1 have been checked, without reaching the conclusion that $S_1 \ncong S_2$, then we know that $S_1 \cong S_2$. □

4 Counting Isomorphisms

Mathon [10] showed that the counting version of Graph Isomorphism is polynomial time Turing reducible to the decision version. Thus, Graph Isomorphism behaves differently than the known **NP**-complete problems, because their counting versions seems much harder than their decision versions. This was historically the first hint that Graph Isomorphism might not be **NP**-complete. We prove analogous results for #Iso-Eqn$^*_G$.

Given a system of equations S on variables X over a finite group G, we are interested in the set of permutations π of the variables in X such that $\pi(S) \equiv S$. It is easy to see that this set of permutations forms a group, denoted $aut(S)$, the automorphism

group of S. #Aut-Eqn$^*_G$ is the problem of counting the number of automorphisms of a system of equations S over G. That is, computing $|aut(S)|$.

Lemma 22 *#Aut-Eqn$^*_G$ is Turing reducible to* Iso-Eqn$^*_G$.

Proof: First note that we can determine whether a system of equations S is solvable or not by making a single query to Iso-Eqn$^*_G$. If S is insoluble, then of course $|aut(S)| = |X|!$, where X is the set of variables occurring in S.

We call variables x_i and x_j equivalent with respect to S if, for any assignment α that satisfies S, we have $\alpha(x_i) = \alpha(x_j)$. By $E_S(x_i)$ we denote the set of variables in S that are equivalent to x_i. Consider any automorphism $\rho \in aut(S)$, if ρ maps x_i to x_k, then clearly $\rho(E_S(x_i)) = E_S(x_k)$; moreover, each bijection from $E_S(x_i)$ to $E_S(x_k)$ can be extended to an automorphism of S. Note that when S is a system of equations over an Abelian group, then the sets $E_S(x_i)$ for all x_i in X, can be computed in polynomial time: just check whether or not $S \equiv S \cup \{x_i = x_j\}$ for all x_j in X (remember that by Corollary 7 we know that equivalence of systems of equations over Abelian groups is in **P**). In the case where S is a system of equations over a non-Abelian group we can use an **NP**-oracle to check whether or not $S \equiv S \cup \{x_i = x_j\}$ for all x_j in X. Since Iso-Eqn$^*_G$ for non-Abelian groups is **coNP**-hard, we can in particular use this method when we have Iso-Eqn$^*_G$ as an oracle.

The notion of equivalence of variables is an equivalence relation on X. For each equivalence class choose a variable in that class, say x_i, and replace all occurrences of the variables from $E_S(x_i)$ by x_i in S. The resulting system of equations S' is defined on the set of variables I corresponding to the equivalence classes of X. The number of permutations ρ of X such that $\rho(S) \equiv S$ is equal to

$$|aut(S')| \prod_{x_i \in I} |E_S(x_i)|!.$$

Hence what remains to be done is to compute $|aut(S')|$, i.e., the number of permutations π of I such that $\pi(S') \equiv S'$.

We need a construction that forces a variable to be mapped to itself under any permutation ρ of I, such that $\rho \in aut(S')$. We use the same construction as the one used in the proof of [1, Lemma 5.1] to achieve this goal.

Note that ρ always maps x_i to a variable x_k such that $|E_{S'}(x_i)| = |E_{S'}(x_k)|$. The idea is to make $|E_{S'}(x_i)|$ unique by introducing new equations of the form $x_i = z_{ij}$, where z_{ij} is a new variable called a labelling variable. Recall that S' was constructed from S by eliminating all equivalent variables. Hence, for each x_i in $I = \{x_1, \ldots, x_n\}$ we have $|E_{S'}(x_i)| = 1$. Let $L(x_i)$ denote $\bigcup_{j=1}^{i}\{x_i = z_{ij}\}$, and $S'_{[I]}$ the system of equations that is constructed as follows,

$$S'_{[I]} = S' \cup \bigcup_{x_i \in I} L(x_i).$$

Now with the technicalities of the labelling out of the way, we can proceed to compute $|aut(S')|$. Let $S'_{[x_1,\ldots,x_n]}$ be the system of equations (corresponding to S') where all variables have been labelled in the way described above. Let $aut_{id}(S'_{[I]})$ denote the set (group) of automorphisms of $S'_{[I]}$ that acts as the identity mapping on all labelling variables. Hence, $|aut_{id}(S'_{[x_1,\ldots,x_n]})| = 1$ and $aut_{id}(S') = aut(S')$. The key for computing $|aut(S')|$ lies in the fact that $|aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})| = d_i|aut_{id}(S'_{[x_1,\ldots,x_{i-1},x_i]})|$, where d_i is the size of the orbit of x_i under the action of $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$.

Recall that the orbit of x_i under the action of $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$ is the set $\{\rho(x_i) \mid \rho \in aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})\}$. Let d_i be the size of the orbit o_i of x_i under the action of $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$, and let ϕ_k ($1 \leq k \leq d_i$) be an automorphism in $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$ which maps x_i to the k-th variable in o_i. Every $\rho \in aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})\}$ can be decomposed as a product of a unique $\phi \in \{\phi_1, \ldots, \phi_{d_i}\}$ and a unique $\tau \in aut_{id}(S'_{[x_1,\ldots,x_{i-1},x_i]})$. Hence, $|aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})| = d_i|aut_{id}(S'_{[x_1,\ldots,x_{i-1},x_i]})|$. It should be clear that $|aut(S')|$ equals

$d_1 d_2 \ldots d_n$. The fact that the order of a permutation group can be computed from the size of the orbits in the way described above is a standard result from group theory, see for example [7] for details.

The orbit of x_i under $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$ can be found by making $n-i$ queries to ISO-EQN$^*_G$. Ask the query $S'_{[x_1,\ldots,x_{i-1}]}[x_i] \cong S'_{[x_1,\ldots,x_{i-1}]}[x_j]$ for each variable x_j, $j \geq i$, where $S'_{[x_1,\ldots,x_{i-1}]}[x_i]$ and $S'_{[x_1,\ldots,x_{i-1}]}[x_j]$ denotes systems of equations where x_i and x_j have been assigned the same (unique) label (using the method described above). If the answer is yes, we know that x_j is in the orbit of x_i under $aut_{id}(S'_{[x_1,\ldots,x_{i-1}]})$. Hence $|aut(S')| = d_1 d_2 \ldots d_n$ can be computed by making $O(n^2)$ queries to ISO-EQN$^*_G$. This completes the proof of the fact that #AUT-EQN$^*_G$ is Turing reducible to ISO-EQN$^*_G$. □

The following theorem states that it is no harder to count the number of isomorphisms between two systems of equations over a fixed finite group than to decide whether an isomorphism exists at all. This indicates that ISO-EQN$^*_G$ is not **NP**-complete for Abelian groups, and that ISO-EQN$^*_G$ is not $\mathbf{\Sigma_2^P}$-complete for non-Abelian groups.

Theorem 23 *#ISO-EQN$^*_G$ is Turing equivalent to ISO-EQN$^*_G$.*

Proof: The proof is based on the same principle as the analogous proof for GRAPH ISOMORPHISM due to Mathon [10]. For Mathon's arguments to work, we need to make sure that all variables in X occur in both S_1 and S_2. This can be easily achieved by padding S_1 (S_2) with (dummy) equations of the form $x = x$, for all variables x that are in X but do not occur in S_1 (S_2). Denote the padded systems by S'_1 and S'_2 respectively. It should be clear that the number of permutations π of X such that $\pi(S_1) \equiv S_2$ equals the number of permutations π of X such that $\pi(S'_1) \equiv S'_2$. Hence we will assume from now on that all variables in X occur both in S_1 and S_2.

Of course ISO-EQN$^*_G$ is trivially polynomial-time reducible to #ISO-EQN$^*_G$. The rest of the proof will be spent on establishing a polynomial-time Turing reduction from #ISO-EQN$^*_G$ to ISO-EQN$^*_G$. Let N_1 be the number of permutations π of X such that $\pi(S_1) \equiv S_2$. If S_1 is not isomorphic to S_2, i.e., there exists no permutation π of X such that $\pi(S_1) \equiv S_2$, then $N_1 = 0$. Otherwise, $N_1 = |aut(S_1)| = |aut(S_2)|$. This can be realized by noting that if π is a permutation such that $\pi(S_1) \equiv S_2$, and $\rho \in aut(S_1)$, then $\rho \circ \pi$ is also a permutation such that $\rho \circ \pi(S_1) \equiv S_2$ (note that $\rho \circ \pi$ denotes the composition where ρ is applied prior to π, hence $\rho \circ \pi(S_1) = \pi(\rho(S_1))$). In addition, any permutation π' such that $\pi'(S_1) \equiv S_2$ can be uniquely expressed as $\pi' = \rho' \circ \pi$, where $\rho' \in aut(S_1)$. Thus, the set of isomorphisms from S_1 to S_2 is the right coset, $aut(S_1) \circ \pi$, where π is an isomorphism from S_1 to S_2. Hence, it follows from Lemma 22 that #ISO-EQN$^*_G$ is Turing reducible to ISO-EQN$^*_G$. □

Mathon's algorithm for computing the number of isomorphisms of two graphs, has the interesting property that at each intermediate stage the number of isomorphisms of the labelled graphs are known. This property has been exploited to prove that GRAPH ISOMORPHISM (GI) is low for **PP**, i.e., GRAPH ISOMORPHISM is powerless as an oracle to **PP** ($\mathbf{PP}^{\mathrm{GI}} = \mathbf{PP}$) [7]. This is generally interpreted as further evidence for the hypothesis that GRAPH ISOMORPHISM is not **NP**-complete. For the details and significance of lowness results, again consult [7]. The properties of our algorithm (that is based on Mathon's algorithm) for computing #ISO-EQN$^*_G$ together with the same argument as that preceding Theorem 5.3 in [1] implies the following lowness results.

Corollary 24 $\mathbf{PP}^{\mathrm{IAb}} = \mathbf{PP}$, *and* $\mathbf{PP}^{\mathrm{IG}} = \mathbf{PP}^{\mathbf{NP}}$, *where* IAb *denotes* ISO-EQN$^*_G$ *for Abelian groups and* IG *denotes* ISO-EQN$^*_G$ *for non-Abelian groups.*

Hence ISO-EQN$^*_G$ for Abelian groups is powerless as an oracle to

PP, and ISO-EQN$^*_G$ for non-Abelian groups is no more powerful than an **NP**-oracle for **PP**.

5 Conclusions

We give dichotomies (under the assumption that **coNP**≠**P**) for the complexity of EQUIV-EQN$^*_G$ and EQUIV-EQN$^*_{G,k}$ for all finite groups G and constants k. A natural direction for future research would be to prove similar dichotomies for EQUIV-EQN$^*_G$ and EQUIV-EQN$^*_{G,k}$ for all finite semigroups G and constants k. But, in light of the recent results in [8], we must say that these problems seem very challenging.

As for the complexity of ISO-EQN$^*_G$, the situation is not as clear. We prove that the problem is in $\mathbf{\Sigma_2^P}$ and **coNP**-hard in the non-Abelian case, and that it is in **NP** and GRAPH ISOMORPHISM-hard in the Abelian case. But the results in Theorem 23 and Corollary 24 give strong indications that these upper bounds ($\mathbf{\Sigma_2^P}$ and **NP** respectively) are not tight. For the isomorphism problem for systems of equations of bounded length, we prove that ISO-B-EQN$^*_{G,k}$ is in **P** when $k \leq 2$ (for all groups G). If $k \geq 3$, then ISO-B-EQN$^*_{G,k}$ is GRAPH ISOMORPHISM-complete when G is Abelian (and different from the trivial group), and **coNP**-hard and in $\mathbf{P}_{||}^{\mathbf{NP}}$ for all non-Abelian groups G.

Our results for the complexity of the equivalence and isomorphism problems for systems of equations over finite groups complement the complexity results for deciding solvability and counting solutions to systems of equations over finite groups presented in [6, 12]. The equivalence and isomorphism problems for systems of equations over finite groups are natural special cases of the equivalence and isomorphism problems for constraints over finite domains. Hence, our results can also be seen as a first attempt of generalizing the results due to Böhler et al. [2, 4] on the complexity of the equivalence and isomorphism problems for Boolean constraints.

It is interesting to observe that so many of the constructions and proof techniques previously used in the literature in relation to other isomorphism problems (e.g., [1, 2, 4, 10]) can be reused. This seems to suggest that a more unified treatment of these and similar isomorphism problems is possible.

Another question left open by the present paper is that of the relative complexity of the isomorphism problems, e.g., it is far from clear whether or not there exists a polynomial-time reduction from ISO-EQN$^*_G$ to ISO-B-EQN$^*_{G,k}$ for some constant k.

Moreover, it has been proved by the use of interactive proofs, that GRAPH ISOMORPHISM and formula isomorphism are not **NP**-complete and $\mathbf{\Sigma_2^P}$-complete respectively, unless the polynomial hierarchy collapses [1, 7]. It would be interesting to investigate whether similar techniques can be used to prove analogous results for ISO-EQN$^*_G$.

Acknowledgments

The author thanks the anonymous referees for many valuable comments.

References

[1] M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM J. Comput.*, 30(3):990–1009, 2000.

[2] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *Proc. Ann. Conf. of the European Association for Computer Science Logic (CSL)*, pages 412–426, 2002.

[3] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. Technical report, arXiv:cs.CC/0202036, 2002.

[4] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. The complexity of boolean constraint isomorphism. In *Proc. 21st Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 164–175, 2004.

[5] S. Buss and L. Hay. On truth-table reducibility to SAT. *Inf. Comput.*, 90(2):86–102, 1991.

[6] M. Goldmann and A. Russell. The complexity of solving equations over finite groups. *Inf. Comput.*, 178(1):253–262, 2002.

[7] H. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, Boston, 1993.

[8] O. Klíma, P. Tesson, and D. Thérien. Dichotomies in the complexity of solving systems of equations over finite semigroups. *Electr. Colloq. on Computational Complexity (ECCC)*, 11(091), 2004.

[9] S. Lang. *Algebra*. Addison-Wesley, Reading MA, 1993.

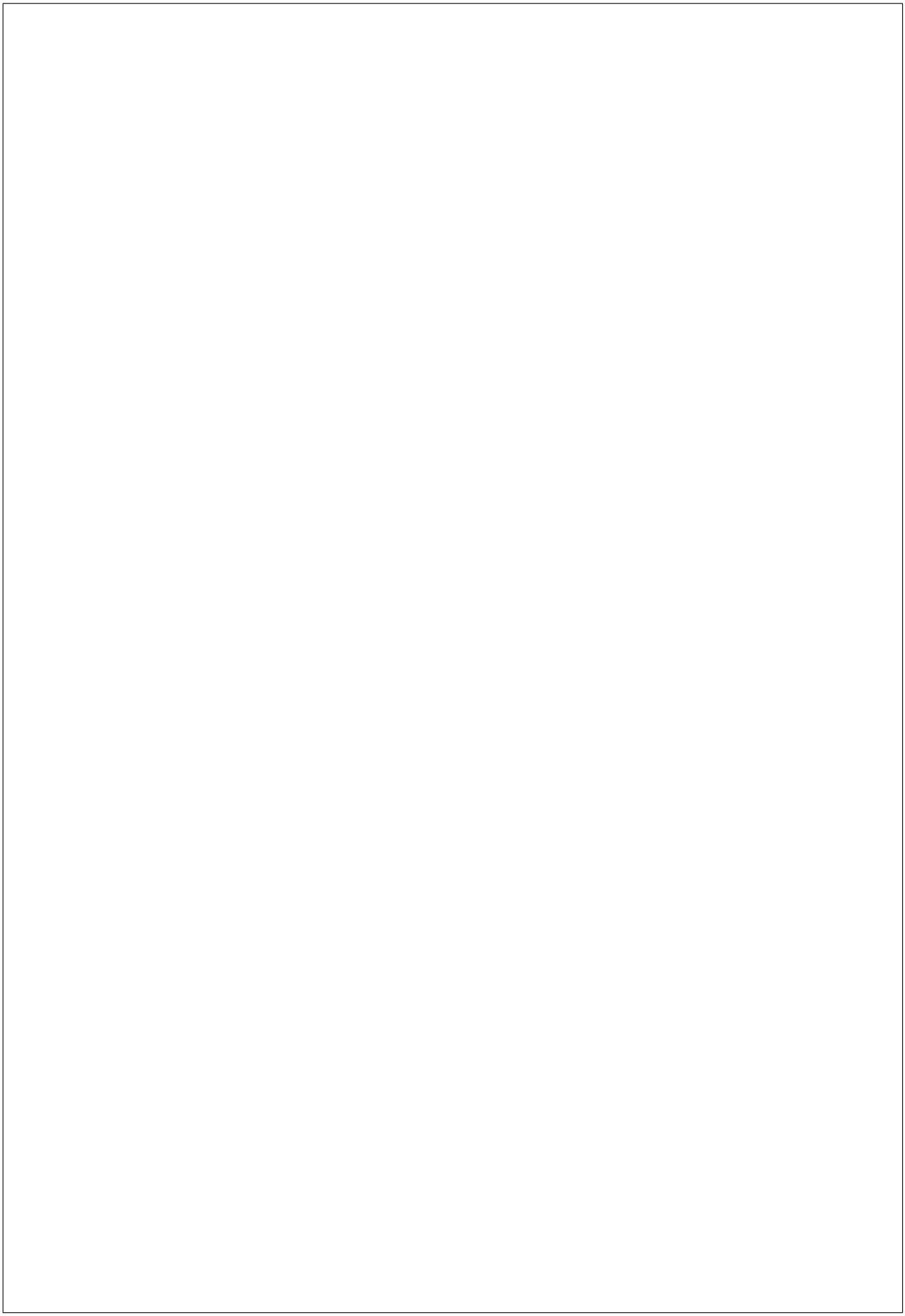
[10] R. Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979.

[11] C. Moore, P. Tesson, and D. Thérien. Satisfiability of systems of equations over finite monoids. In *Proc. 26th Int. Symp. on Mathematical Foundations of Computer Science (MFCS)*, pages 537–547, 2001.

[12] G. Nordh and P. Jonsson. The complexity of counting solutions to systems of equations over finite semigroups. In *Proc. 10th Int. Computing and Combinatorics Conference (COCOON)*, pages 370–379, 2004.

[13] T. Thierauf. The isomorphism problem for read-once branching programs and arithmetic circuits. *Chicago J. Theor. Comput. Sci.*, 1998.

Paper III



An Algebraic Approach to the Complexity of Propositional Circumscription

Gustav Nordh and Peter Jonsson

Abstract

Every logical formalism gives rise to two fundamental problems: model checking and inference. Circumscription is one of the most important and well-studied formalisms in the realm of nonmonotonic reasoning. The model checking and inference problem for propositional circumscription has been extensively studied from the viewpoint of computational complexity. We use a new approach based on algebraic techniques to study the complexity of the model checking and inference problems for propositional variable circumscription in a unified way. We prove that there exists a dichotomy theorem for the complexity of the inference problem in propositional variable circumscription. We also study the model checking and inference problem for propositional variable circumscription in many-valued logics using the same algebraic techniques. In particular we prove dichotomy theorems for the complexity of model checking and inference for propositional variable circumscription in the important case of 3-valued logic.

1 Introduction

Circumscription, introduced by McCarthy [17], is perhaps the most well developed and extensively studied formalism in non-monotonic reasoning. The key intuition behind circumscription is that by focusing on minimal models of formulas we achieve some degree of common sense, because minimal models have as few "exceptions" as possible.

Propositional circumscription in Boolean logic is the basic case of circumscription in which satisfying truth assignments of propositional formulas are partially ordered according to the coordinatewise partial order $\leq$ on Boolean vectors, which extends the order $0 \leq 1$ on $\{0, 1\}$. In propositional variable circumscription only a certain subset of the variables in formulas are subject to minimization, others must maintain a fixed value or are subject to no restrictions at all. Given a propositional Boolean formula T and a partition of the variables in T into three (possibly empty) disjoint subsets $(P; Z; Q)$ where P is the set of variables we want to minimize, Z is the set of variables allowed to vary and Q is the set of variables that must maintain a fixed value, we define the partial order on satisfying models as follows. Let α, β be two models of T, then $\alpha \leq_{(P;Z)} \beta$ if α and β assign the same value to the variables in Q and for every variable p in P, $\alpha(p) \leq \beta(p)$ (moreover if there exist a variable p in P such that $\alpha(p) \neq \beta(p)$, we write $\alpha <_{(P;Z)} \beta$). A minimal model of a formula T is a satisfying model α such that there exist no satisfying model β where $\beta <_{(P;Z)} \alpha$.

We will from now on call the restricted form of propositional circumscription where all variables are subject to minimization (that is $Q = Z = \emptyset$) for basic circumscription, and the more general propositional variable circumscription for propositional circumscription.

Every logical formalism gives rise to two fundamental decision problems: model checking and inference. In the case of propositional circumscription in Boolean logic the model checking and

inference problems can be formulated as follows.

- **Model checking:** Given a propositional Boolean formula T, a partition of the variables in T into three disjoint subsets $(P; Z; Q)$ and a truth assignment α, is α a minimal model of T.
- **Inference:** Given two propositional Boolean formulas T and T' and a partition of the variables in T into three disjoint (possibly empty) subsets $(P; Z; Q)$, is T' true in every minimal model of T. If T' is true in every minimal model of T, we denote this by $T \models_{CIRC} T'$.

The formulas T and T' are assumed to be given in conjunctive normal form. It is easy to realize that the inference problem is polynomial-time equivalent to the case where T' is a single clause, since T' can be inferred from T under propositional circumscription if and only if each clause of T' can be so inferred.

The inference problem for propositional circumscription is known to have strong connections to the inference problem under closed world assumptions. More specifically the inference problem for propositional Boolean logic under extended closed world assumption (ECWA) is equivalent to the inference problem for propositional circumscription in Boolean logic [12].

The inference and model checking problems for circumscription in propositional Boolean logic are very well-studied from the computational complexity perspective [5, 6, 7, 10, 11, 14, 15]. The model checking problem for propositional circumscription in Boolean logic has been proved to be **coNP**-complete [5]. The inference problem for propositional circumscription in Boolean logic has been proved to be $\mathbf{\Pi_2^P}$-complete [11]. These results display a dramatic increase in the computational complexity compared to the case of ordinary propositional Boolean logic, where the model checking problem is solvable in linear time and the inference problem is **coNP**-complete [8]. These negative results raise the problem of identifying restricted cases in which the

model checking and inference problem for propositional circumscription have computational complexity lower than the general case.

The most natural way to study such restrictions is to study restrictions on the formulas representing knowledge bases, denoted T in above. This is also the approach followed in most of the previous research in the area. Hence in the case of restrictions of the inference problem, we only restrict the formula T while T' are subject to no restrictions.

The ultimate goal of this line of research is to determine the complexity of every restricted special case of the problem. The first result of this type was proved by Schaefer [19], who succeeded in obtaining a complete classification of the satisfiability problem in propositional Boolean logic. He proved that every special case of the satisfiability problem in propositional Boolean logic either is tractable or **NP**-complete (note that this implies that the inference problem in propositional Boolean logic is either tractable or **coNP**-complete). Recall the result due to Ladner [16] that if $\mathbf{P} \neq \mathbf{NP}$, then there exist decision problems in **NP** that are neither tractable nor **NP**-complete. Hence the existence of dichotomy theorems like Schaefer's cannot be taken for granted. Schaefer raised the question of the existence of dichotomy theorems for the complexity of the satisfiability problem in propositional many-valued logics. Despite intensive research on this problem, progress has been annoyingly slow. Not until recently a dichotomy theorem for the complexity of the satisfiability problem in 3-valued logics was proved [1].

In the case of the complexity of circumscription in propositional logic, the situation is even more unsatisfying. Recently a dichotomy theorem was proved for the complexity of the model checking problem in the Boolean case [15]. Moreover for the case of basic circumscription (where $Q = Z = \emptyset$) in Boolean logic, it has been proved that every special case of the inference problem is either $\mathbf{\Pi_2^P}$-complete or lies in **coNP** [14]. In the case of the

complexity of circumscription in propositional many-valued logics, to the best of our knowledge, nothing is known neither on the model checking problem nor the inference problem.

In this paper we try to remedy this situation by investigating in a unified way the complexity of circumscription in propositional many-valued logics in general and the complexity of circumscription in propositional 3-valued logics in particular. This investigation is motivated by the fact that in many real world AI applications it is necessary to reason with incomplete or contradicting information. Consider for example the operation of an autonomous agent that has to cope with incomplete information of the world and contradicting information from its sensors. We could represent incomplete or contradicting information by introducing a new truth value, $\frac{1}{2}$. Now if we want to use circumscription in the process of reasoning we just have to give a suitable partial ordering of the truth values. In the case of an autonomous agent a suitable ordering might be $\frac{1}{2} \leq 0$ and $\frac{1}{2} \leq 1$, that is we would like to minimize incomplete and contradicting information. For an interesting example of how circumscription is used in a 3-valued setting, see [9].

We prove dichotomy theorems for the complexity of both the model checking and inference problems for circumscription in propositional 3-valued logics. Moreover we present for the first time a dichotomy theorem for the complexity of the inference problem for circumscription in propositional Boolean logic in the general case where Q and Z need not be empty. In addition all our dichotomy theorems comes with efficiently checkable criteria to decide the complexity of a given subclass of the problem. These results are obtained by the use of novel techniques from universal algebra.

Although basic circumscription (where $Q = Z = \emptyset$) is a restricted case of the problem we study, our dichotomies do not imply dichotomies for basic circumscription. This is because our hardness results does not in general carry over to the restricted

case where $Q = Z = \emptyset$. Hence the existence of dichotomies for the inference and model checking problems for basic circumscription in 3-valued logic is still open.

Also note that we do not fix a particular ordering of the truth values, i.e., we only demand that the ordering is a partial order. In particular we say that a special case of the problem is tractable if it is tractable for all possible partial orders of the truth values and that a special case of the problem is hard if there exist a partial order of the truth values such that it is hard. Since previous research has focused on circumscription in Boolean logic where there are only one reasonable ordering of the truth values (namely $0 \leq 1$), this decision has not been necessary before. But even in the case where the ordering of the truth values is fixed, all our results holds if the ordering is a total order. Moreover we follow the approach in [14, 15] where the clauses of T and T' are allowed to be arbitrary logical relations (sometimes called generalized clauses).

The paper is organized as follows. In Section 2 we give the necessary background on Constraint Satisfaction Problems (CSPs) and the algebraic techniques we will use throughout this paper. In Section 3 we prove our results on the complexity of the model checking problem for circumscription in propositional logic. In Section 4 we prove analogous results on the inference problem, and finally we give some ideas for future research in Section 5.

2 Preliminaries

In this section we introduce the notation and basic results on CSPs and the algebraic techniques that we will use in the rest of this paper.

Let D be a finite set. The set of all n-tuples of elements from D is denoted by D^n. Any subset of D^n is called an n-ary relation on D. The set of all finitary relations over D is denoted by R_D.

In this paper we will mainly deal with finitary relations over two or three-element sets, denoted R_2 and R_3 respectively.

Definition 1 A constraint language over a finite set, D, is an arbitrary set $\Gamma \subseteq R_D$.

Given a constraint language Γ over a domain $D = \{d_1, d_2, \ldots, d_k\}$, we denote the constraint language $\Gamma \cup \{\{(d_1)\}, \{(d_2)\}, \ldots, \{(d_k)\}\}$ by Γ^{id}. Constraint languages are the way in which we specify restrictions on our problems. For example in the case of the inference problem for propositional circumscription over the constraint language Γ, we demand that all the relations in the knowledge base are present in Γ.

Definition 2 The constraint satisfaction problem over the constraint language Γ, denoted $\textsc{Csp}(\Gamma)$, is defined to be the decision problem with instance (V, D, C), where

- V is a set of variables,
- $D = \{d_1, d_2, \ldots, d_k\}$ is a finite set of values (sometimes called a domain), and
- C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation over the set D, belonging to Γ, called the constraint relation.

The question is whether there exists s solution to (V, D, C), that is, a function from V to D such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

Example 3 Let NAE be the following ternary relation on $\{0, 1\}$: $NAE = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}$. It is easy to see that the well-known **NP**-complete problem Not-All-Equal Sat can be expressed as $\textsc{Csp}(\{NAE\})$.

Next we consider operations on the domain D. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows.

Definition 4 Let f be a k-ary operation on D and let R be an n-ary relation over D. For any collection of k tuples, $t_1, t_2, \ldots, t_k \in R$, the n-tuple $f(t_1, t_2, \ldots, t_k)$ is defined as follows:

$$f(t_1, t_2, \ldots, t_k) =$$

$$(f(t_1[1], t_2[1], \ldots, t_k[1]), \ldots, f(t_1[n], t_2[n], \ldots, t_k[n]))$$

where $t_j[i]$ is the i:th component in tuple t_j.

A technique that has shown to be useful in determining the computational complexity of $\textsc{Csp}(\Gamma)$ is that of investigating whether Γ is closed under certain families of operations [13].

Definition 5 Let $\varrho_i \in \Gamma$. If f is an operation such that for all $t_1, t_2, \ldots, t_k \in \varrho_i$ $f(t_1, t_2, \ldots, t_k) \in \varrho_i$, then ϱ_i is closed under f. If all constraint relations in Γ are closed under f then Γ is closed under f. An operation f such that Γ is closed under f is called a polymorphism of Γ. The set of all polymorphisms of Γ is denoted $Pol(\Gamma)$. Given a set of operations F, the set of all relations that is closed under all the operations in F is denoted $Inv(F)$.

Definition 6 For any set $\Gamma \subseteq R_D$ the set $\langle\Gamma\rangle$ consists of all relations that can be expressed using relations from $\Gamma \cup \{=_D\}$ ($=_D$ is the equality relation on D), conjunction, and existential quantification.

A relation belongs to $\langle\Gamma\rangle$ if and only if it can be represented as the projection of the set of all solutions to some instance of $\textsc{Csp}(\Gamma)$ onto some subset of variables. Intuitively, constraints using relations from $\langle\Gamma\rangle$ are exactly those which can be simulated by constraints using relations from Γ. The sets of relations of the

form $\langle\Gamma\rangle$ are often referred to as relational clones. An alternative characterization of relational clones is given in the following theorem.

Theorem 7 ([18]) *For every set* $\Gamma \subseteq R_D$, $\langle\Gamma\rangle = Inv(Pol(\Gamma))$.

The following set of operations will be essential in the classification of the complexity of subproblems specified by constraint languages.

Definition 8 Let D be a finite set. An operation f on D is called

- a constant operation if there is a $c \in D$ such that $f(d_1, \ldots, d_n) = c$ for any $d_1, \ldots, d_n \in D$;
- a near-unanimity operation if it is n-ary and satisfies the identities $f(d_2, d_1, \ldots, d_1) = f(d_1, d_2, d_1, \ldots, d_1) = \cdots = f(d_1, \ldots, d_1, d_2) = d_1$ for any $d_1, d_2 \in D$;
- a Mal'tsev operation if it is ternary and $f(d_1, d_1, d_2) = f(d_2, d_1, d_1) = d_2$, for any $d_1, d_2 \in D$;
- a semilattice operation, if it is binary and satisfies the following three conditions: $f(d_1, f(d_2, d_3)) = f(f(d_1, d_2), d_3)$, $f(d_1, d_2) = f(d_2, d_1)$, and $f(d_1, d_1) = d_1$, for any $d_1, d_2, d_3 \in D$;
- an essentially unary operation if $f(d_1, \ldots, d_n) = g(d_i)$ for some non constant unary operation g, and any $d_1, \ldots, d_n \in D$ (if g is the identity operation, then f is often called a projection).

The first dichotomy theorem for a broad class of decision problems was Schaefer's dichotomy theorem for the complexity of the satisfiability problem in propositional Boolean logic.

Theorem 9 ([19]) *For any constraint language* $\Gamma \subseteq R_2$, $\textsc{Csp}(\Gamma)$ *is tractable when one of the following conditions holds:*

1. *Every* ϱ *in* Γ *contains the tuple* $(0, 0, \ldots, 0)$.
2. *Every* ϱ *in* Γ *contains the tuple* $(1, 1, \ldots, 1)$.
3. *Every* ϱ *in* Γ *is definable by a CNF formula in which each conjunct has at most one negated variable.*
4. *Every* ϱ *in* Γ *is definable by a CNF formula in which each conjunct has at most one unnegated variable.*
5. *Every* ϱ *in* Γ *is definable by a CNF formula in which each conjunct has at most two literals.*
6. *Every* ϱ *in* Γ *is definable by a system of linear equations over the field with two elements.*

In all other cases $\textsc{Csp}(\Gamma)$ *is* **NP***-complete.*

Schaefer's result has later been given a much shorter and simplified proof using the algebraic techniques that we will later apply to the model checking and inference problems for circumscription in propositional logics. Schaefer's result can be reformulated in algebraic terms as follows.

Theorem 10 ([13]) *Let* $\Gamma \subseteq R_2$ *be a constraint language. Then,* $\textsc{Csp}(\Gamma)$ *is* **NP***-complete if* $Pol(\Gamma)$ *only contains essentially unary operations, and tractable otherwise.*

The following property of constraint languages is important when determining the complexity of $\textsc{Csp}(\Gamma)$. We only state the property in the restricted case where the constraint languages contain all unary singleton relations. The reason for this is that these restricted constraint languages will later be of great importance in our study of the model checking and inference problems in propositional circumscription. Note that this property is more

naturally stated in terms of finite algebras [3]. Since we only use this property indirectly and the actual definition of this property is not essential to the understanding of the paper we choose to give a less beautiful and intelligible definition in terms of relational clones (in order to avoid introducing the terminology of finite algebras).

Definition 11 Let $\Gamma \subseteq R_D$ be a constraint language. Γ^{id} is said to satisfy the G-Set property if there exist a subset A of D and an equivalence relation θ on A such that $\theta \in Inv(Pol(\Gamma^{id}))$, and every polymorphism of Γ^{id}, when restricted to A and then factorized by θ, is a projection on the factor set A/θ.

Bulatov have recently used algebraic techniques to prove a dichotomy theorem for the complexity of the satisfiability problem in propositional 3-valued logic. We only state the dichotomy theorem in the restricted case where the constraint languages contain all unary singleton relations.

Theorem 12 ([1]) *Let $\Gamma \subseteq R_3$ be a constraint language. Then,* Csp(Γ^{id}) *is* **NP***-complete if Γ^{id} satisfies the* G-Set *property, and tractable otherwise.*

Moreover a dichotomy conjecture has been proposed for the complexity of the satisfiability problem in propositional many-valued logic. Again, we only state the conjecture in the restricted case where the constraint languages contain all unary singleton relations.

Conjecture 13 ([1]) *Let $\Gamma \subseteq R_D$ be a constraint language.* Csp(Γ^{id}) *is* **NP***-complete if Γ^{id} satisfies the* G-Set *property, and tractable otherwise.*

Before we present the definitions of the constraint satisfaction analogous of the model checking and inference problems in

propositional circumscription we need to define formally the partial order on functions/vectors that we use to define minimal models.

Definition 14 Let $\leq$ denote a partial order on a domain D. Let k be a positive integer and let $\alpha = (a_1, \ldots, a_k)$, $\beta = (b_1, \ldots, b_k)$ be two k-tuples in D^k, let P, Q, Z be a partition of $\{1, 2, \ldots, k\}$ into disjoint (possibly empty) subsets. We write $\beta \leq_{(P;Z)} \alpha$ to denote that $b_i \leq a_i$ for all $i \in P$ and $b_j = a_j$ for all $j \in Q$. Let $\beta <_{(P;Z)} \alpha$ denote that $\beta \leq_{(P;Z)} \alpha$ and that there exist an $i \in P$ such that $b_i \neq a_i$.

Next we introduce the minimal constraint satisfaction problem, it should be clear that this problem is equivalent to the model checking problem for circumscription in many-valued logics.

Definition 15 The minimal constraint satisfaction problem over the constraint language Γ, denoted Min-Csp(Γ), is defined to be the decision problem with instance $(V, P, Z, Q, D, \leq, C, \alpha)$, where P, Z, Q is a partition of V into disjoint (possibly empty) subsets and

- V is a set of variables,
- P represents the variables to minimize,
- Z represents the variables that vary,
- Q represents the variables that are fixed,
- D is a finite set of values (sometimes called a domain),
- $\leq$ is a partial order on D,
- C is a set of constraints $\{C_1, \ldots, C_q\}$ in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation

over the set D, belonging to Γ, called the constraint relation, and

- α is a function from V to D such that, for each constraint in C the image of the constraint scope is a member of the constraint relation.

The question is whether α is a minimal solution. That is, whether there exists no function β from V to D such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation and $\beta <_{(P;Z)} \alpha$.

Next we introduce the minimal constraint inference problem, it should be clear that this problem is equivalent to the inference problem for circumscription in many-valued logics.

Definition 16 The minimal constraint inference problem over the constraint language Γ, denoted Min-Inf-Csp(Γ), is defined to be the decision problem with instance $(V, P, Z, Q, D, \leq, C, \psi)$, where $V, P, Z, Q, D, \leq$, and C are defined as in Definition 15 and ψ is a constraint such that the set of variables in ψ's constraint scope is a subset of V.

The question is whether each minimal solution α to $(V, P, Z, Q, D, \leq, C)$ is a solution to ψ.

The size of a problem instance of Min-Inf-Csp(Γ) (Min-Csp(Γ)) is the length of the encoding of all tuples in all the constraints in C.

We conclude this section by defining formally what we mean when we say that a certain special case of a problem is in, or complete for certain complexity class.

Definition 17 The problem Min-Csp(Γ) (Min-Inf-Csp(Γ)) is in C (for a complexity class C) if for any finite $\Gamma' \subseteq \Gamma$ the problem Min-Csp(Γ') (Min-Inf-Csp(Γ')) is in C. The problem Min-Csp(Γ) (Min-Inf-Csp(Γ)) is C-complete (for a complexity class

C) if Min-Csp(Γ') (Min-Inf-Csp(Γ')) is C-hard for a certain finite $\Gamma' \subseteq \Gamma$, and Min-Csp(Γ) (Min-Inf-Csp(Γ)) is in C.

3 Model Checking

The model checking problem for circumscription in propositional Boolean logic was proved by Cadoli to be **coNP**-complete [5]. Some special cases of the problem was also shown to be tractable in the same paper. The question of whether a dichotomy theorem could be proved for the computational complexity of the problem was raised at that time, but was left open. This open question was finally resolved by Kirousis and Kolaitis and given an affirmative answer in [15].

We give a simplified proof of the dichotomy theorem for the complexity of the model checking problem for circumscription in propositional Boolean logic. We also prove interesting results on the complexity of the corresponding problem in many-valued logics. In particular we prove a dichotomy theorem for the complexity of the model checking problem for circumscription in propositional 3-valued logics. In addition these dichotomy theorems provides us with efficiently checkable criteria to decide whether a given subclass of the problem is tractable or **coNP**-complete. We also make explicit the strong connection between this problem and CSP. Note that it is easy to realize that the model checking problem is in **coNP**.

We begin by stating the dichotomy theorem, due to Kirousis and Kolaitis [15], for the complexity of the model checking problem for circumscription in propositional Boolean logic.

Theorem 18 ([15]) *Let $\Gamma \subseteq R_2$ be a constraint language.* Min-Csp(Γ) *is tractable if at least one of the conditions 3,4,5,6 in Theorem 9 are satisfied; otherwise it is* **coNP***-complete.*

We will later (in Theorem 25) give an alternative proof of the preceding theorem using the algebraic approach.

Example 19 The ternary NOT-ALL-EQUAL relation on {0,1}, NAE, satisfies none of the conditions 3,4,5,6 in Theorem 9, hence MIN-CSP($\{NAE\}$) is **coNP**-complete.

The following theorem forms the basis of the algebraic approach to determine the complexity of the model checking problem for circumscription in propositional logic. It states that when studying the complexity of MIN-CSP(Γ) it is sufficient to consider constraint languages that are relational clones. Hence the class of constraint languages to be studied can be considerably reduced.

Theorem 20 MIN-CSP(Γ) *is tractable (***coNP***-complete) if and only if* MIN-CSP($\langle\Gamma\rangle$) *is tractable (***coNP***-complete).*

Proof: Since $\Gamma \subseteq \langle\Gamma\rangle$, if MIN-CSP($\Gamma$) is **coNP**-complete then so is MIN-CSP($\langle\Gamma\rangle$), and if MIN-CSP($\langle\Gamma\rangle$) is tractable then so is MIN-CSP(Γ).

To prove the converse implications take a finite set $\Gamma_0 \subseteq \langle\Gamma\rangle$ and an instance $S = (V, P, Z, Q, D, \leq, C, \alpha)$ of MIN-CSP(Γ_0). We transform S into an equivalent instance $S' = (V', P', Z', Q', D, \leq, C', \alpha')$ of MIN-CSP(Γ_1), where Γ_1 is a finite subset of Γ.

For every constraint $C = ((v_1, \ldots, v_m), \varrho)$ in S, ϱ can be represented on the form $\varrho(v_1, \ldots, v_m) =$

$$\exists_{v_{m+1}}, \ldots, \exists_{v_n} \varrho_1(v_{11}, \ldots, v_{1n_1}) \wedge \cdots \wedge \varrho_k(v_{k1}, \ldots, v_{kn_k})$$

where $\varrho_1, \ldots, \varrho_k \in \Gamma \cup \{=_D\}$, $v_{m+1}, \ldots, v_n$ are new variables not previously present in S, and $v_{11}, \ldots, v_{1n_1}, v_{21}, \ldots, v_{kn_k} \in \{v_1, \ldots, v_n\}$. Replace the constraint C with the constraints

$$((v_{11}, \ldots, v_{1n_1}), \varrho_1), \ldots, ((v_{k1}, \ldots, v_{kn_k}), \varrho_k).$$

Add $v_{m+1}, \ldots, v_n$ to V and Z. Extend α so that α is a solution to all of the new constraints. We know that this is always possible since α was a solution to $\varrho(v_1, \ldots, v_m)$, and since $\varrho_1 \wedge \cdots \wedge \varrho_k \in \langle\Gamma \cup \{=_D\}\rangle$ such an extension can be found in polynomial time

by checking each tuple in $\varrho_1 \wedge \cdots \wedge \varrho_k$. If we repeat the same reduction for every constraint in C it results in an equivalent instance $S'' = (V'', P, Z'', Q, D, \leq, C'', \alpha'')$ of MIN-CSP($\Gamma_1 \cup \{=_D\}$).

For each equality constraint $((v_i, v_j), =)$ in S'' we do the following:

- If both v_i and v_j are in P (Z'', Q) we remove v_i from P (Z'', Q) and V'', replace all occurrences of v_i in C'' by v_j and modify α'' so that it is not defined on v_i.
- If v_j is in Q and v_i is in P we remove v_i from P and V'', replace all occurrences of v_i in C'' by v_j and modify α'' so that it is not defined on v_i. The case where v_j is in P and v_i is in Q is handled in the same way.
- The case that remains is when one of v_i and v_j is in Z'', assume without loss of generality that v_i is in Z''. We remove v_i from Z'' and V'', replace all occurrences of v_i in C'' by v_j and modify α'' so that it is not defined on v_i.

Finally remove $((v_i, v_j), =)$ from C''. The resulting instance $S' = (V', P', Z',\ Q', D, \leq, C', \alpha')$ of MIN-CSP(Γ_1) is equivalent to S and has been obtained in polynomial time. Hence if MIN-CSP(Γ) is tractable then so is MIN-CSP($\langle\Gamma\rangle$), and if MIN-CSP($\langle\Gamma\rangle$) is **coNP**-complete then so is MIN-CSP(Γ). □

The following theorem states that all the information about the complexity of MIN-CSP(Γ) can be extracted from the set of polymorphisms of Γ.

Theorem 21 *Let Γ_1 and Γ_2 be sets of relations over a finite set, such that Γ_1 is finite. If $Pol(\Gamma_2) \subseteq Pol(\Gamma_1)$ then* MIN-CSP(Γ_1) *is polynomial-time reducible to* MIN-CSP(Γ_2).

Proof: $Pol(\Gamma_2) \subseteq Pol(\Gamma_1)$ implies that

$$Inv(Pol(\Gamma_1)) \subseteq Inv(Pol(\Gamma_2))$$

or equivalently $\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle$. Then by Theorem 20 it follows that MIN-CSP(Γ_1) is polynomial-time reducible to MIN-CSP(Γ_2). □

Hence what remains to be done is to determine the complexity of MIN-CSP(Γ) for all possible sets of polymorphisms of Γ. The following theorem will be very useful in this effort.

Theorem 22 MIN-CSP(Γ) *is tractable (***coNP***-complete) if and only if* MIN-CSP(Γ^{id}) *is tractable (***coNP***-complete).*

Proof: Since $\Gamma \subseteq \Gamma^{id}$, it follows that MIN-CSP(Γ) is tractable if MIN-CSP(Γ^{id}) is tractable, and MIN-CSP(Γ^{id}) is **coNP**-complete if MIN-CSP(Γ) is **coNP**-complete.

To prove the converse implications take a finite set $\Gamma_0 \subseteq \Gamma^{id}$ and an instance $S = (V, P, Z, Q, D, \leq, C, \alpha)$ of MIN-CSP(Γ_0). We transform S into an equivalent instance $S' = (V', P', Z', Q', D, \leq, C', \alpha')$ of MIN-CSP(Γ_1), where Γ_1 is a finite subset of Γ.

For all variables x occurring in a constraint in S of the type $((x), (d_i))$ two cases emerge.

- In the case where x occurs in no other constraints, remove x from V, P, Z and Q, let α be undefined on x, and remove the constraint.
- In the case where x do occurs in another constraint, remove x from P and Z, add x to Q and remove the constraint.

It is clear that α' is a minimal solution in the resulting instance $S' = (V', P', Z', Q', D, \leq, C', \alpha')$ of MIN-CSP(Γ_1) if and only if α is a minimal solution in the original instance $S = (V, P, Z, Q, D, \leq, C, \alpha)$ of MIN-CSP(Γ_0). □

There is a strong relation between the complexity of CSP(Γ) and MIN-CSP(Γ). This relation is in part made explicit in the following theorem that will be useful in proving tractability results for MIN-CSP(Γ).

Theorem 23 *If* CSP(Γ^{id}) *is tractable, then* MIN-CSP(Γ) *is tractable.*

Proof: We give a polynomial time Turing reduction from MIN-CSP(Γ) to CSP(Γ^{id}). Let $min(D)$ denote the set of minimal elements in D (as specified by $\leq$). Given an instance $S = (V, P, Z, Q, D, \leq, C, \alpha)$ of MIN-CSP(Γ), let P^+ denote the set $\{p \in P | \alpha(p)$ is not in $min(D)\}$, and $S' = (V, D, C')$ denote the instance of CSP(Γ^{id}) where

$$C' = C \cup \bigcup_{p \in P \setminus P^+} \{(p, \alpha(p))\} \cup \bigcup_{q \in Q} \{(q, \alpha(q))\}.$$

Then α is a minimal solution for C if and only if for each $s \in P^+$ and for each $d \in D$ such that $d < \alpha(s)$ the instance $S'' = (V, D, C'')$ of CSP(Γ^{id}) where $C'' = C' \cup \{(s, (d))\}$ is unsatisfiable.

We begin by proving the if part. Suppose that α is a minimal solution of C and that there exists an $s \in P^+$ and a $d \in D$ such that $d < \alpha(s)$ and C'' is satisfiable. Let β be a solution of C''. Of course β is also a solution of C, and $\beta <_{(P;Z)} \alpha$ contradicting that α is a minimal solution of C.

Now to the only if part. Suppose that for each $s \in P^+$ and each $d \in D$ such that $d < \alpha(s)$, C'' is unsatisfiable, and suppose that a solution β of C exists such that $\beta <_{(P;Z)} \alpha$. Since $\beta <_{(P;Z)} \alpha$, there exists at least one $t \in P$, such that $\beta(t) < \alpha(t)$. Since $t \in P^+$, it follows that there exists one $s \in P^+$ and a $d \in D$ such that $d < \alpha(s)$ and C'' is satisfiable, contradicting the assumption above. □

All of our **coNP**-completeness results for the model checking problem can be deduced from the following theorem.

Theorem 24 *Let $\Gamma \subseteq R_D$ be a constraint language. If Γ^{id} satisfies the* G-SET *property then* MIN-CSP(Γ) *is* **coNP***-complete.*

Proof: If Γ^{id} satisfies the G-SET property, then it follows from the proof of Proposition 7.9 in [4] that there is a relation $\varrho \in \langle \Gamma^{id} \rangle$ such that $\varrho = f^{-1}(NAE)$, where NAE is the ternary NOT-ALL-EQUAL relation on {0,1}, and f is a function from some subset

$A \subseteq D$ onto $\{0,1\}$. Note that by $\varrho = f^{-1}(NAE)$ we mean that ϱ is the following relation $\{(a_1,a_2,a_3)|(c_1,c_2,c_3) \in NAE$, and $a_i \in f^{-1}(c_i)\}$. The intuition behind ϱ is that it is similar to NAE, but we can view more than one element as 0 (1).

We shall prove that MIN-CSP($\{\varrho\}$) is **coNP**-complete by reduction from the **coNP**-complete problem MIN-CSP($\{NAE\}$). Let $\leq$ be a total order on D, $A_0 = f^{-1}(0)$, and $A_1 = f^{-1}(1)$. Furthermore let a_0 be the least element of A_0 (w.r.t. $\leq$), a_1 the least element of A_1, and assume without loss of generality that $a_0 \leq a_1$. Then map every instance S of MIN-CSP($\{NAE\}$) to an instance S' of MIN-CSP($\{\varrho\}$) which have the same structure, but NAE is replaced by ϱ throughout, and in the given solution α, every 0 is replaced by a_0, and every 1 is replaced by a_1. It should be clear that by the choice of a_0, a_1, and the structure of ϱ, S is equivalent to S'. Since $\varrho \in \langle \Gamma^{id} \rangle$ it follows from Theorem 20 and Theorem 22 that MIN-CSP(Γ) is **coNP**-complete. □

Next we present an alternative proof of the dichotomy theorem for the complexity of the model checking problem for circumscription in propositional Boolean logic (Theorem 18).

Theorem 25 *Let $\Gamma \subseteq R_2$ be a constraint language.* MIN-CSP(Γ) *is* **coNP***-complete if $Pol(\Gamma^{id})$ only contains essentially unary operations, and tractable otherwise.*

Proof: If $Pol(\Gamma^{id})$ contains an operation that is not essentially unary then we can conclude from Theorem 10 and Theorem 23 that MIN-CSP(Γ^{id}) is tractable. Hence by Theorem 22 MIN-CSP(Γ) is tractable when $Pol(\Gamma^{id})$ contains an operation that is not essentially unary. What remains to be proved is that MIN-CSP(Γ) is **coNP**-complete when $Pol(\Gamma^{id})$ only contains essentially unary operations. First note that if $Pol(\Gamma^{id})$ only contains essentially unary operations then $Pol(\Gamma^{id})$ only contains essentially unary operations f whose corresponding unary operation g is the identity operation. This is because if $g(d_i) \neq d_i$, then the relation $\{(d_i)\}$ would not be closed under f. Hence

$Inv(Pol(\Gamma^{id})) = R_2$, the set of all finitary relations over $\{d_1, d_2\}$. Thus we conclude by Cadoli's **coNP**-completeness result for MIN-CSP(R_2) [5] and Theorem 20 that MIN-CSP(Γ^{id}) is **coNP**-complete and by Theorem 22, MIN-CSP(Γ) is **coNP**-complete. □

Note that the techniques used in [15] to prove the dichotomy theorem for the complexity of the model checking problem for circumscription in propositional Boolean logic does not seem to be applicable in the case of many-valued propositional logics. On the contrary the algebraic techniques used in this paper are capable of proving interesting results on the complexity of the model checking problem for circumscription in propositional many-valued logics. In particular we are able to present a dichotomy theorem for the complexity of the model checking problem for circumscription in propositional 3-valued logics.

Theorem 26 *Let $\Gamma \subseteq R_3$ be a constraint language.* MIN-CSP(Γ) *is* **coNP***-complete if Γ^{id} satisfies the* G-SET *property, and tractable otherwise.*

Proof: If Γ^{id} satisfies the G-SET property then by Theorem 24 we know that MIN-CSP(Γ) is **coNP**-complete. Otherwise we know by Theorem 12 that CSP(Γ^{id}) is tractable and thus by Theorem 23 we get that MIN-CSP(Γ) is tractable. □

Note that the borderline between tractability and **coNP**-completeness in the preceding theorem coincides with the borderline between tractability and **NP**-completeness for CSP(Γ) in Theorem 12. Hence when $\Gamma \subseteq R_3$ we can use the same polynomial time algorithm as presented in [1] to check whether MIN-CSP(Γ) is **coNP**-complete or tractable.

By using Theorem 23 and known results on the complexity of CSP(Γ) we can identify several tractable cases of the model checking problem for circumscription in propositional many-valued logics.

Theorem 27 *If $Pol(\Gamma)$ contains a near-unanimity operation, a*

semilattice operation or a Mal'tsev operation then MIN-CSP(Γ) *is tractable.*

Proof: It is proved in [2, 13] that CSP(Γ) is tractable if $Pol(\Gamma)$ contains a near-unanimity operation, a semilattice operation or a Mal'tsev operation. Note that if $Pol(\Gamma)$ contains a near-unanimity operation, a semilattice operation or a Mal'tsev operation, then so does $Pol(\Gamma^{id})$. Hence the result follows from Theorem 23. □

We conclude this section by presenting a conjecture on the complexity of the model checking problem for circumscription in propositional many-valued logics.

Conjecture 28 *Let* $\Gamma \subseteq R_D$ *be a constraint language.* MIN-CSP(Γ) *is* **coNP***-complete if* Γ^{id} *satisfies the* G-SET *property, and tractable otherwise.*

Note that the **coNP**-completeness part of the conjecture follows from Theorem 24. Also note that the tractability part of the conjecture would follow if the conjecture about the structure of the dichotomy for CSP(Γ) (Conjecture 13) was proved to be true.

4 Inference

The inference problem for circumscription in propositional Boolean logic was proved by Eiter and Gottlob to be $\mathbf{\Pi_2^P}$-complete [11]. Some special cases of the problem was already at that time known to have complexity lower than the general case, i.e., special cases that are in **coNP** [7]. Kirousis and Kolaitis managed to prove a dichotomy theorem for the complexity of the inference problem in basic circumscription ($Q = Z = \emptyset$) for propositional Boolean logic [14]. More specifically they prove that every special case of the problem is either in **coNP** or $\mathbf{Pi_2^P}$-complete.

We prove a dichotomy theorem for the complexity of the inference problem for circumscription in propositional Boolean logic.

We also prove interesting results on the complexity of the corresponding problem in many-valued logics. In particular we prove a dichotomy theorem for the complexity of the inference problem for circumscription in propositional 3-valued logics. In addition these dichotomy theorems provides us with efficiently checkable criteria to decide whether a given subclass of the problem is in **coNP** or $\mathbf{\Pi_2^P}$-complete. Note that it is easy to realize that the inference problem is in $\mathbf{\Pi_2^P}$.

The following theorem forms the basis of the algebraic approach to determine the complexity of the inference problem for circumscription in propositional logic. The proof is very similar to the proof of Theorem 20.

Theorem 29 MIN-INF-CSP(Γ) *is in* **coNP** *(*$\mathbf{\Pi_2^P}$*-complete) if and only if* MIN-INF-CSP($\langle\Gamma\rangle$) *is in* **coNP** *(*$\mathbf{\Pi_2^P}$*-complete).*

Proof: Since $\Gamma \subseteq \langle\Gamma\rangle$, if MIN-INF-CSP($\Gamma$) is $\mathbf{\Pi_2^P}$-complete then so is MIN-INF-CSP($\langle\Gamma\rangle$), and if MIN-INF-CSP($\langle\Gamma\rangle$) is tractable then so is MIN-INF-CSP(Γ).

To prove the converse implications take a finite set $\Gamma_0 \subseteq \langle\Gamma\rangle$ and an instance $S = (V, P, Z, Q, D, \leq, C, \psi)$ of MIN-INF-CSP(Γ_0). We transform S into an equivalent instance $S' = (V', P', Z', Q', D, \leq, C', \psi')$ of MIN-INF-CSP(Γ_1), where Γ_1 is a finite subset of Γ.

For every constraint $C = ((v_1, \ldots, v_m), \varrho)$ in S, ϱ can be represented on the form $\varrho(v_1, \ldots, v_m) =$

$$\exists_{v_{m+1}}, \ldots, \exists_{v_n} \varrho_1(v_{11}, \ldots, v_{1n_1}) \wedge \cdots \wedge \varrho_k(v_{k1}, \ldots, v_{kn_k})$$

where $\varrho_1, \ldots, \varrho_k \in \Gamma \cup \{=_D\}$, $v_{m+1}, \ldots, v_n$ are new variables not previously present in S, and $v_{11}, \ldots, v_{1n_1}, v_{21}, \ldots, v_{kn_k} \in \{v_1, \ldots, v_n\}$. Replace the constraint C with the constraints

$$((v_{11}, \ldots, v_{1n_1}), \varrho_1), \ldots, ((v_{k1}, \ldots, v_{kn_k}), \varrho_k).$$

Add $v_{m+1}, \ldots, v_n$ to V and Z. If we repeat the same reduction for every constraint in C it results in an equivalent instance $S'' = (V'', P, Z'', Q, D, \leq, C'', \psi)$ of MIN-INF-CSP($\Gamma_1 \cup \{=_D\}$).

For each equality constraint $((v_i, v_j), =)$ in S'' we do the following:

- If both v_i and v_j are in P (Z'', Q) we remove v_i from P (Z'', Q) and V'', replace all occurrences of v_i in C'' and ψ by v_j.
- If v_j is in Q and v_i is in P we remove v_i from P and V'', replace all occurrences of v_i in C'' and ψ by v_j. The case where v_j is in P and v_i is in Q is handled in the same way.
- The case that remains is when one of v_i and v_j is in Z'', assume without loss of generality that v_i is in Z''. We remove v_i from Z'' and V'', replace all occurrences of v_i in C'' and ψ by v_j.

Finally remove $((v_i, v_j), =)$ from C''.

The resulting instance $S' = (V', P', Z', Q', D, \leq, C', \psi')$ of Min-Inf-Csp(Γ_1) is equivalent to S and has been obtained in polynomial time. Hence if Min-Inf-Csp(Γ) is tractable then so is Min-Inf-Csp($\langle\Gamma\rangle$), and if Min-Inf-Csp($\langle\Gamma\rangle$) is $\mathbf{\Pi_2^P}$-complete then so is Min-Inf-Csp(Γ). □

The following theorem states that all the information about the complexity of Min-Inf-Csp(Γ) can be extracted from the set of polymorphisms of Γ. The proof is essentially the same as the proof of Theorem 21.

Theorem 30 *Let Γ_1 and Γ_2 be sets of relations over a finite set, such that Γ_1 is finite. If $Pol(\Gamma_2) \subseteq Pol(\Gamma_1)$ then* Min-Inf-Csp(Γ_1) *is polynomial-time reducible to* Min-Inf-Csp(Γ_2).

Hence what remains to be done is to determine the complexity of Min-Inf-Csp(Γ) for all possible sets of polymorphisms of Γ. The following theorem will be very useful in this effort. The proof is substantially different from the proof of the corresponding result (Theorem 22) for Min-Csp(Γ). Here we are forced

to make changes to the constraint to be inferred when reducing MIN-INF-CSP(Γ^{id}) to MIN-INF-CSP(Γ).

Theorem 31 MIN-INF-CSP(Γ) *is in* **coNP** *(*$\mathbf{\Pi_2^P}$*-complete) if and only if* MIN-INF-CSP(Γ^{id}) *is in* **coNP** *(*$\mathbf{\Pi_2^P}$*-complete).*

Proof: Since $\Gamma \subseteq \Gamma^{id}$, it follows that MIN-INF-CSP(Γ) is in **coNP** if MIN-INF-CSP(Γ^{id}) is in **coNP**, and MIN-INF-CSP(Γ^{id}) is $\mathbf{\Pi_2^P}$-complete if MIN-INF-CSP(Γ) is $\mathbf{\Pi_2^P}$-complete.

To prove the converse implications take a finite set $\Gamma_0 \subseteq \Gamma^{id}$ and an instance $S = (V, P, Z, Q, D, \leq, C, \psi)$ of MIN-INF-CSP(Γ_0). We transform S into an equivalent instance $S' = (V', P', Z', Q', D, \leq, C', \psi')$ of MIN-CSP(Γ_1), where Γ_1 is a finite subset of Γ.

For all variables x, y occurring in a constraint in S of the type $((x), (d_i))$ and $((y), (d_i))$, replace the constraint $((y), (d_i))$ by $((x, y), =)$. By the proof of Theorem 29 we know that MIN-INF-CSP($\Gamma \cup \{=_D\}$) is polynomial time reducible to MIN-INF-CSP(Γ). Hence only $|D|$ constraints of the type $((x), (d_i))$ need to be considered. For all variables x occurring in a constraint in S of the type $((x), (d_i))$ three cases emerge.

- In the case where x occurs neither in ψ nor in any other constraints, remove x from V, P, Z and Q, and remove the constraint.

- In the case where x occurs in another constraint but not in ψ, remove x from P and Z, add x to Q and remove the constraint. Update ψ as follows. Append x to the constraint scope of ψ. Let the constraint relation consist of the following tuples $\{\{D \setminus d_i\} \times D^n\} \cup \{d_i \times \varphi_\psi\}$ where n is the length of ψ's constraint scope and φ_ψ is ψ's constraint relation.

- In the case where x occurs in another constraint and in ψ, remove x from P and Z, add x to Q and remove the

constraint. Update ψ as follows. Let the constraint relation consist of the following tuples $\{\{D^j \times \{D \setminus d_i\} \times D^k\} \cup \varphi_\psi\}$ where $j+k+1$ is the length of ψ's constraint scope, φ_ψ is ψ's constraint relation, and $j+1$ is the position of x in ψ's constraint scope.

More informally the idea behind the reduction is as follows. If $((x),(d_i))$ is a constraint in C, we remove it and modify ψ to make sure that every minimal model α of $C \setminus \{((x),(d_i))\}$ such that $\alpha(x) \neq d_i$, is a model of ψ, and in the case where $\alpha(x) = d_i$ we make sure that α is a model of the modified ψ if and only if α was a model of the original ψ. It should be clear that S and S' are equivalent. □

There is a strong connection between the model checking and inference problem for circumscription in propositional logics.

Theorem 32 ([7]) *If* Min-Csp(Γ) *is tractable then* Min-Inf-Csp(Γ) *is in* **coNP**.

Note that the argument in [7] only deals with circumscription in propositional Boolean logic, but it is obvious that the same argument holds also in the case of propositional many-valued logics.

All of our $\mathbf{\Pi_2^P}$-completeness results for the inference problem can be deduced from the following theorem. The proof is similar to the proof of Theorem 24.

Theorem 33 *Let $\Gamma \subseteq R_D$ be a constraint language. If Γ^{id} satisfies the* G-Set *property then* Min-Inf-Csp(Γ) *is $\mathbf{\Pi_2^P}$-complete.*

Proof: If Γ^{id} satisfies the G-Set property, then it follows from the proof of Proposition 7.9 in [4] that there is a relation $\varrho \in \langle \Gamma^{id} \rangle$ such that $\varrho = f^{-1}(NAE)$, where NAE is the ternary Not-All-Equal relation on {0,1} in Example 3, and f is a function from some subset $A \subseteq D$ onto $\{0,1\}$. Note that by $\varrho = f^{-1}(NAE)$

we mean that ϱ is the following relation $\{(a_1, a_2, a_3)|(c_1, c_2, c_3) \in NAE$, and $a_i \in f^{-1}(c_i)\}$. The intuition behind ϱ is that it is similar to NAE, but we can view more than one element as 0 (1).

We shall prove that MIN-INF-CSP($\{\varrho\}$) is $\mathbf{\Pi_2^P}$-complete by reduction from the $\mathbf{\Pi_2^P}$-complete problem MIN-INF-CSP($\{NAE\}$). Let $\leq$ be a total order on D, $A_0 = f^{-1}(0)$, and $A_1 = f^{-1}(1)$. Furthermore let a_0 be the least element of A_0 (w.r.t. $\leq$), a_1 the least element of A_1, and assume without loss of generality that $a_0 \leq a_1$. Then map every instance S of MIN-INF-CSP($\{NAE\}$) to an instance S' of MIN-INF-CSP($\{\varrho\}$) which have the same structure, but NAE is replaced by ϱ throughout, and in the constraint to be inferred ψ, every 0 is replaced by a_0, and every 1 is replaced by a_1. It should be clear that by the choice of a_0, a_1, and the structure of ϱ, S is equivalent to S'. Since $\varrho \in \langle \Gamma^{id} \rangle$ it follows from Theorem 29 and Theorem 31 that MIN-INF-CSP(Γ) is $\mathbf{\Pi_2^P}$-complete. □

Next we present a proof of the dichotomy theorem for the complexity of the inference problem for circumscription in propositional Boolean logic. Note that previously no dichotomy theorem was known for this problem. The proof is very similar to the proof of Theorem 25.

Theorem 34 *Let $\Gamma \subseteq R_2$ be a constraint language.* MIN-INF-CSP(Γ) *is $\mathbf{\Pi_2^P}$-complete if $Pol(\Gamma^{id})$ only contains essentially unary operations, and it is in* **coNP** *otherwise.*

Proof: If $Pol(\Gamma^{id})$ contains an operation that is not essentially unary, then we know from Theorem 25 that MIN-CSP(Γ) is tractable. Hence by Theorem 32 we conclude that MIN-INF-CSP(Γ) is in **coNP** when $Pol(\Gamma^{id})$ contains an operation that is not essentially unary.

What remains to be proved is that MIN-INF-CSP(Γ) is $\mathbf{\Pi_2^P}$-complete when $Pol(\Gamma^{id})$ only contains essentially unary operations. First note that if $Pol(\Gamma^{id})$ only contains essentially unary

operations then $Pol(\Gamma^{id})$ only contains essentially unary operations f whose corresponding unary operation g is the identity operation. This is because if $g(d_i) \neq d_i$, then the relation $\{(d_i)\}$ would not be closed under f. Hence $Inv(Pol(\Gamma^{id})) = R_2$, the set of all finitary relations over $\{d_1, d_2\}$. Thus we conclude by Eiter and Gottlob's $\mathbf{\Pi_2^P}$-completeness result for MIN-INF-CSP(R_2) [11] and Theorem 29 that MIN-INF-CSP(Γ^{id}) is $\mathbf{\Pi_2^P}$-complete and hence by Theorem 31, MIN-INF-CSP(Γ) is $\mathbf{\Pi_2^P}$-complete. □

Now we are ready to present the dichotomy theorem for the complexity of the inference problem for circumscription in propositional 3-valued logics.

Theorem 35 *Let $\Gamma \subseteq R_3$ be a constraint language.* MIN-INF-CSP(Γ) *is $\mathbf{\Pi_2^P}$-complete if Γ^{id} satisfies the* G-SET *property, and in* **coNP** *otherwise.*

Proof: If Γ^{id} satisfies the G-SET property then the $\mathbf{\Pi_2^P}$-completeness part follows from Theorem 33. Otherwise we know from Theorem 26 that MIN-CSP(Γ) is tractable. Hence by Theorem 32 we conclude that MIN-INF-CSP(Γ) is in **coNP**. □

Note that the borderline between subproblems in **coNP** and $\mathbf{\Pi_2^P}$-complete subproblems in the preceding theorem coincides with the borderline between tractability and **NP**-completeness for CSP(Γ) in Theorem 12. Hence when $\Gamma \subseteq R_3$ we can use the same polynomial time algorithm as presented in [1] to check whether MIN-INF-CSP(Γ) is in **coNP** or $\mathbf{\Pi_2^P}$-complete.

We can identify several cases of the inference problem for circumscription in propositional many-valued logics having lower complexity than the general case.

Theorem 36 *If $Pol(\Gamma)$ contains a near-unanimity operation, a semilattice operation or a Mal'tsev operation then* MIN-INF-CSP(Γ) *is in* **coNP**.

Proof: It is proved in Theorem 27 that MIN-CSP(Γ) is tractable if $Pol(\Gamma)$ contains a near-unanimity operation, a semilattice operation or a Mal'tsev operation. Thus it follows from Theorem 32 that MIN-INF-CSP(Γ) is in **coNP**. □

We conclude this section by presenting a conjecture on the complexity of the inference problem for circumscription in propositional many-valued logics.

Conjecture 37 *Let $\Gamma \subseteq R_D$ be a constraint language.* MIN-INF-CSP(Γ) *is $\mathbf{\Pi_2^P}$-complete if Γ^{id} satisfies the* G-SET *property, and in* **coNP** *otherwise.*

Note that the $\mathbf{\Pi_2^P}$-completeness part of the conjecture follows from Theorem 33. Also note that the tractability part of the conjecture would follow if the conjecture about the structure of the dichotomy for CSP(Γ) (Conjecture 13) was proved to be true.

5 Future Research

The obvious continuation of this work would be to extend the results from 3-valued logics to n-valued logics, but in light of the close relationship here uncovered between these problems and CSP, this will probably have to await the resolution of the dichotomy conjecture for CSP. Another interesting open problem is the existence of dichotomies for basic circumscription (where $Q = Z = \emptyset$) in 3-valued logics. Moreover it is certainly the case that there exist constraint languages (knowledge bases) $\Gamma \subseteq R_2$ such that MIN-INF-CSP(Γ) is tractable. Does there in fact exist a trichotomy (**P**, **coNP**-complete, $\mathbf{\Pi_2^P}$-complete) for the complexity of MIN-INF-CSP(Γ)?

Acknowledgements

The authors want to thank Andrei Krokhin for suggesting the proof of Theorem 24 and providing many valuable remarks on an earlier version of this paper, and Andrzej Szalas for interesting discussions on the topics in this paper. Gustav Nordh is supported by the *National Graduate School in Computer Science* (CUGS), Sweden. Peter Jonsson is partially supported by the *Center for Industrial Information Technology* (CENIIT) under grant 04.01, and by the *Swedish Research Council* (VR) under grants 221-2000-361 and 621-2003-3421.

References

[1] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 649–658, 2002.

[2] A. Bulatov. Mal'tsev constraints are tractable. Technical report, PRG-RR-02-05, Computing Laboratory, University of Oxford, 2002.

[3] A. Bulatov, P. Jeavons, and A. Krokhin. Constraint satisfaction problems and finite algebras. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 272–282, 2000.

[4] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras (revised version). *Submitted for publication, available at* `http://www.dcs.warwick.ac.uk/people/academic/Andrei.Krokhin/papers.html`, 2003.

[5] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 42:113–118, 1992.

[6] M. Cadoli. *Tractable Reasoning in Artificial Intelligence*. Number 941 in Lecture Notes in Artificial Intelligence. Springer Verlag Berlin Heidelberg, 1995.

[7] M. Cadoli and M. Lenzerini. The complexity of closed world reasoning and circumscription. *Journal of Computer and System Sciences*, pages 255–301, 1994.

[8] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[9] P. Doherty, J. Kachniarz, and A. Szałas. Using contextually closed queries for local closed-world reasoning in rough knowledge databases. In S. Pal, L. Polkowski, and A. Skowron, editors, *Rough-Neuro Computing: Techniques for Computing with Words*, Cognitive Technologies, pages 219–250. Springer–Verlag, 2003.

[10] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is **coNP**-complete. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, pages 451–462, 2003.

[11] T. Eiter and G. Gottlob. Propositional circumscription and extended closed-world reasoning are $\mathbf{\Pi_2^P}$-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[12] M. Gelfond, H. Przymusinska, and T. Przymusinsky. On the relationship between circumscription and negation as failure. *Artificial Intelligence*, 38:49–73, 1989.

[13] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[14] L. Kirousis and P. Kolaitis. A dichotomy in the complexity of propositional circumscription. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 71–80, 2001.

[15] L. Kirousis and P. Kolaitis. The complexity of minimal satisfiability problems. *Information and Computation*, 187(1):20–39, 2003.

[16] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[17] J. McCarthy. Circumscription - a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[18] R. Pöschel and L. Kaluznin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.

[19] T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.

Paper IV

A Trichotomy in the Complexity of Propositional Circumscription

Gustav Nordh

Abstract

Circumscription is one of the most important and well-studied formalisms in the realm of nonmonotonic reasoning. The inference problem for propositional circumscription has been extensively studied from the viewpoint of computational complexity. We prove that there exists a trichotomy for the complexity of the inference problem in propositional variable circumscription. More specifically we prove that every restricted case of the problem is either $\mathbf{\Pi_2^P}$-complete, **coNP**-complete, or in **P**.

1 Introduction

Circumscription, introduced by McCarthy [13], is perhaps the most well developed and extensively studied formalism in non-monotonic reasoning. The key intuition behind circumscription is that by focusing on minimal models of formulas we achieve some degree of common sense, because minimal models have as few "exceptions" as possible.

Propositional circumscription is the basic case of circumscription in which satisfying truth assignments of propositional formulas are partially ordered according to the coordinatewise partial order $\leq$ on Boolean vectors, which extends the order $0 \leq 1$ on $\{0, 1\}$. In propositional variable circumscription only a certain subset of the variables in formulas are subject to minimization, others must maintain a fixed value or are subject to no restrictions at all. Given a propositional formula T and a partition of the variables in T into three (possibly empty) disjoint subsets $(P; Z; Q)$ where P is the set of variables we want to minimize, Z is the set of variables allowed to vary and Q is the set of variables that must maintain a fixed value, we define the partial order on satisfying models as follows. Let α, β be two models of T, then $\alpha \leq_{(P;Z)} \beta$ if α and β assign the same value to the variables in Q and for every variable p in P, $\alpha(p) \leq \beta(p)$ (moreover if there exists a variable p in P such that $\alpha(p) \neq \beta(p)$, we write $\alpha <_{(P;Z)} \beta$). A minimal model of a formula T is a satisfying model α such that there exists no satisfying model β where $\beta <_{(P;Z)} \alpha$.

We will from now on call the restricted form of propositional circumscription where all variables are subject to minimization (that is $Q = Z = \emptyset$) for basic circumscription, and the more general propositional variable circumscription for propositional circumscription.

Every logical formalism gives rise to the fundamental problem of inference. In the case of propositional circumscription the inference problem can be formulated as follows.

- **Inference:** Given two propositional Boolean formulas T and T' and a partition of the variables in T into three disjoint (possibly empty) subsets $(P; Z; Q)$, is T' true in every minimal model of T.

The formulas T and T' are assumed to be given in conjunctive normal form. It is easy to realize that the inference problem is equivalent (under polynomial-time conjunctive reductions) to the case where T' is a single clause, since T' can be inferred from T under propositional circumscription if and only if each clause of T' can be so inferred. Moreover we follow the approach in [10, 11, 14] where the clauses of T are allowed to be arbitrary logical relations (sometimes called generalized clauses). This approach was first used by Schaefer to classify the complexity of the satisfiability problem in propositional logic and is sometimes referred to as Schaefer's framework [18].

Circumscription in propositional logic is very well-studied from the computational complexity perspective [2, 4, 7, 8, 10, 11, 14]. The inference problem for propositional circumscription has been proved to be $\mathbf{\Pi_2^P}$-complete [8]. This result displays a dramatic increase in the computational complexity compared to the case of ordinary propositional logic, where the inference problem is **coNP**-complete [3]. This negative result raise the problem of identifying restricted cases in which the inference problem for propositional circumscription have computational complexity lower than the general case.

The most natural way to study such restrictions is to study restrictions on the formulas representing knowledge bases, denoted T in above. This is also the approach followed in most of the previous research in the area. Hence in the case of restrictions of the inference problem, we only restrict the formula T while T' are subject to no restrictions. The ultimate goal of this line of research is to determine the complexity of every restricted special case of the problem. The first result of this type was proved by Schaefer [18], who succeeded in obtaining a complete

classification of the satisfiability problem in propositional logic. He proved that every special case of the satisfiability problem in propositional logic either is tractable or **NP**-complete (note that this implies that the inference problem in propositional logic is either tractable or **coNP**-complete). Recall the result due to Ladner [12] that if $\mathbf{P} \neq \mathbf{NP}$, then there exist decision problems in **NP** that are neither tractable nor **NP**-complete. Hence the existence of dichotomy theorems like Schaefer's cannot be taken for granted.

Some partial results are known for the complexity of the inference problem. More specifically, both in the general case and in the case where $Q = Z = \emptyset$, it has been proved that every special case of the problem is either $\mathbf{\Pi_2^P}$-complete or lies in **coNP** [10, 14].

Until now we have lacked a clear picture of the complexity of the inference problem in propositional circumscription, i.e., no complete classification of special cases of the problem with a complexity in **coNP** as **coNP**-complete or in **P** is known. Some cases are known to be **coNP**-complete, but to the best of our knowledge only one case of the inference problem (where Q and Z need not be empty) is known to be in **P** [2].

We prove that there exists a trichotomy theorem for the complexity of the inference problem, i.e., for every special case of the problem it is either in **P**, **coNP**-complete, or $\mathbf{\Pi_2^P}$-complete. Moreover we discover two new tractable cases. These results are obtained by the use of techniques from universal algebra. These techniques were first applied to the propositional circumscription problem in [14] where dichotomies for the model checking and inference problem for propositional circumscription in 3-valued logic were proved.

Although basic circumscription (where $Q = Z = \emptyset$) is a restricted case of the problem we study, our trichotomy does not imply a trichotomy for basic circumscription. This is because our hardness results do not in general carry over to the restricted case

where $Q = Z = \emptyset$. Hence the existence of a trichotomy for the inference problem in basic propositional circumscription is still open.

The paper is organized as follows. In Section 2 we give the necessary background on Constraint Satisfaction Problems (CSPs) and the algebraic techniques that we will use throughout this paper. In Section 3 we prove our trichotomy theorem for the complexity of circumscription in propositional logic and finally in Section 4 we give some conclusions.

2 Preliminaries

In this section we introduce the notation and basic results on CSPs and the algebraic techniques that we will use in the rest of this paper.

2.1 Constraint Satisfaction Problems

The set of all n-tuples of elements from $\{0,1\}$ is denoted by $\{0,1\}^n$. Any subset of $\{0,1\}^n$ is called an n-ary relation on $\{0,1\}$. The set of all finitary relations over $\{0,1\}$ is denoted by BR.

Definition 1 A constraint language over $\{0,1\}$ is an arbitrary set $\Gamma \subseteq BR$.

Constraint languages are the way in which we specify restrictions on our problems. For example in the case of the inference problem for propositional circumscription over the constraint language Γ, we demand that all the relations in the knowledge base are present in Γ.

Definition 2 The Boolean constraint satisfaction problem (or the generalized satisfiability problem as Schaefer called it) over the constraint language $\Gamma \subseteq BR$, denoted $\textsc{Csp}(\Gamma)$, is defined to be the decision problem with instance (V, C), where

- V is a set of variables, and
- C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation over the set $\{0, 1\}$, belonging to Γ, called the constraint relation.

The question is whether there exists a solution to (V, C), that is, a function from V to $\{0, 1\}$ such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

Example 3 Let NAE be the following ternary relation on $\{0, 1\}$:

$$NAE = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

It is easy to see that the well-known **NP**-complete problem Not-All-Equal 3-Sat can be expressed as Csp($\{NAE\}$).

Next we consider operations on $\{0, 1\}$. Any operation on $\{0, 1\}$ can be extended in a standard way to an operation on tuples over $\{0, 1\}$, as follows.

Definition 4 Let f be a k-ary operation on $\{0, 1\}$ and let R be an n-ary relation over $\{0, 1\}$. For any collection of k tuples, $t_1, t_2, \ldots, t_k \in R$, the n-tuple $f(t_1, t_2, \ldots, t_k)$ is defined as follows:

$$f(t_1, t_2, \ldots, t_k) =$$

$$(f(t_1[1], t_2[1], \ldots, t_k[1]), \ldots, f(t_1[n], t_2[n], \ldots, t_k[n])),$$

where $t_j[i]$ is the i-th component in tuple t_j.

A technique that has shown to be useful in determining the computational complexity of Csp(Γ) is that of investigating whether Γ is closed under certain families of operations [9].

Definition 5 Let $\varrho_i \in \Gamma$. If f is an operation such that for all $t_1, t_2, \ldots, t_k \in \varrho_i$ $f(t_1, t_2, \ldots, t_k) \in \varrho_i$, then ϱ_i is closed under f. If all constraint relations in Γ are closed under f then Γ is closed under f. An operation f such that Γ is closed under f is called a polymorphism of Γ. The set of all polymorphisms of Γ is denoted $Pol(\Gamma)$. Given a set of operations F, the set of all relations that is closed under all the operations in F is denoted $Inv(F)$.

Definition 6 For any set $\Gamma \subseteq BR$ the set $\langle\Gamma\rangle$ consists of all relations that can be expressed using relations from $\Gamma \cup \{=\}$ ($=$ is the equality relation on $\{0, 1\}$), conjunction, and existential quantification.

Intuitively, constraints using relations from $\langle\Gamma\rangle$ are exactly those which can be simulated by constraints using relations from Γ. The sets of relations of the form $\langle\Gamma\rangle$ are referred to as relational clones, or co-clones. An alternative characterization of relational clones is given in the following theorem.

Theorem 7 ([17]) *For every set $\Gamma \subseteq BR$, $\langle\Gamma\rangle = Inv(Pol(\Gamma))$.*

The first dichotomy theorem for a broad class of decision problems was Schaefer's dichotomy theorem for the complexity of the satisfiability problem in propositional logic [18]. Schaefer's result has later been given a much shorter and simplified proof using the algebraic techniques that we will later apply to the inference problem for propositional circumscription. Schaefer's result can be formulated in algebraic terms as follows.

Theorem 8 ([9]) *Let $\Gamma \subseteq BR$ be a constraint language.* Csp(Γ) *is* **NP***-complete if $Pol(\Gamma)$ only contains essentially unary operations, and tractable otherwise. Note that an operation f is essentially unary if and only if $f(d_1, \ldots, d_n) = g(d_i)$ for some non constant unary operation g, and any $d_1, \ldots, d_n \in \{0, 1\}$.*

As we will see later, constraint languages containing the relations $\{(0)\}$ and $\{(1)\}$ will be of particular importance to us.

Definition 9 Given a constraint language Γ, the idempotent constraint language corresponding to Γ is $\Gamma \cup \{\{(0)\}, \{(1)\}\}$ which is denoted by Γ^{id}.

2.2 Propositional Circumscription

In this section we make some formal definitions and recall some of the results from [14]. Note that the focus of [14] is on propositional circumscription in many-valued logics, and as a consequence the clause to be inferred is allowed to be a general constraint. Since we only consider circumscription in Boolean logic in this paper, this generalization is no longer necessary and in order to comply with the definitions of the problem in [2, 10] we require that the clause to be inferred is an ordinary clause. The results from [14] still holds.

First we introduce the minimal constraint inference problem. It should be clear that this problem is equivalent to the inference problem for propositional circumscription.

Definition 10 The minimal constraint inference problem over the constraint language $\Gamma \subseteq BR$, denoted MIN-INF-CSP(Γ), is defined to be the decision problem with instance (V, P, Z, Q, C, ψ), where $(P; Z; Q)$ is a partition of V into disjoint (possibly empty) subsets and

- V is a set of variables,
- P represents the variables to minimize,
- Z represents the variables that vary,
- Q represents the variables that are fixed,

- C is a set of constraints $\{C_1, \ldots, C_q\}$ in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation over the set $\{0, 1\}$, belonging to Γ, called the constraint relation, and

- ψ is a clause such that the set of variables in ψ is a subset of V.

The question is whether each minimal model α of (V, P, Z, Q, C) is also a model of ψ.

The size of a problem instance of MIN-INF-CSP(Γ) is the length of the encoding of all tuples in all the constraints in C.

We define formally what we mean when we say that a certain special case of a problem is tractable or complete for certain complexity class.

Definition 11 The problem MIN-INF-CSP(Γ) is called tractable if for any finite $\Gamma' \subseteq \Gamma$ the problem MIN-INF-CSP(Γ') is solvable in polynomial time. The problem MIN-INF-CSP(Γ) is called C-complete (for a complexity class C) if MIN-INF-CSP(Γ') is C-hard for a certain finite $\Gamma' \subseteq \Gamma$, and MIN-INF-CSP(Γ) $\in$ C.

The following theorem forms the basis of the algebraic approach to determine the complexity of the inference problem for circumscription in propositional logic. It states that when investigating the complexity of MIN-INF-CSP(Γ) it is sufficient to consider constraint languages that are relational clones.

Theorem 12 ([14]) MIN-INF-CSP(Γ) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete) if and only if* MIN-INF-CSP($\langle\Gamma\rangle$) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete).*

The following theorem reduces the set of constraint languages that need to be considered even further.

Theorem 13 ([14]) MIN-INF-CSP(Γ) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete) if and only if* MIN-INF-CSP(Γ^{id}) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete).*

We conclude this section by stating the dichotomy for the complexity of the inference problem in propositional circumscription that was proved in [14].

Theorem 14 ([14]) *Let* $\Gamma \subseteq BR$ *be a constraint language.* MIN-INF-CSP(Γ) *is* $\mathbf{\Pi_2^P}$*-complete if* $Pol(\Gamma^{id})$ *only contains essentially unary operations, and it is in* **coNP** *otherwise.*

3 Trichotomy Theorem for the Inference Problem

In this section we prove our trichotomy theorem for the complexity of the inference problem in propositional circumscription. In the light of Theorem 14, what remains to be proved is that every problem MIN-INF-CSP(Γ) in **coNP** is either **coNP**-hard or in **P**. Some important cases like Horn clauses [2] and affine clauses [7] are already known to be **coNP**-complete. But to the best of our knowledge the only case known to be in **P** is when the knowledge base only consists of clauses containing at most one positive and negative literal (i.e., clauses that are both Horn and dual-Horn) [2]. Our main results is the discovery of two new tractable classes of knowledge bases (width-2 affine and clauses only containing negative literals) and a proof that for all other classes of knowledge bases the problem is **coNP**-hard.

We prove this by further exploiting the results obtained in [14], e.g., Theorem 12 that states that to determine the complexity of MIN-INF-CSP(Γ) it is sufficient to consider constraint languages that are relational clones.

Emil Post [16] classified all Boolean clones/relational clones and proved that they form a lattice under set inclusion. Our

proofs rely heavily on Post's lattice of Boolean clones/relational clones. An excellent introduction to Post's classification of Boolean clones can be found in the recent survey article [1], for a more complete account, see [15, 17].

See Figure 4 for the lattice of Boolean relational clones. Note that the names for the relational clones in Figure 4 do not agree with Post's names. Post also considered other classes of Boolean functions/relations, so called iterative classes, and this leads to some confusion and inconsistencies if we would use Post's names. The terminology used in Figure 4 was developed by Klaus Wagner in an attempt to construct a consistent scheme of names for clones/relational clones, and was subsequently used in [1].

Now we introduce some relational clones that will be of particular importance to us.

- **Relational Clone $\mathbf{IR_2}$:** For $a \in \{0, 1\}$, a Boolean function f is called a-reproducing if $f(a, \ldots, a) = a$. The clones R_a contain all a-reproducing Boolean functions. The clone R_2 contains all functions that are both 0-reproducing and 1-reproducing. Hence $Inv(R_2) = IR_2$ is the relational clone consisting of all relations closed under all functions that are both 0-reproducing and 1-reproducing. Note that functions satisfying $f(a, \ldots, a) = a$ for all a in its domain are usually called idempotent.

- **Relational Clone $\mathbf{ID_1}$:** ID_1 is the relational clone consisting of all relations closed under the affine operation $f(x, y, z) = x \oplus y \oplus z$ and the ternary majority operation $g(x, y, z) = xy \vee yz \vee xz$. It is proved in [5] (Lemma 4.11) that any relation in ID_1 can be represented as a linear equation on at most two variables over the two element field $GF(2)$. Constraint languages $\Gamma \subseteq ID_1$ are usually called width-2 affine in the literature.

- **Relational Clone $\mathbf{IS_1}$:** S_1 is the clone consisting of all 1-separating functions (see [1] for the definition of 1-separat-

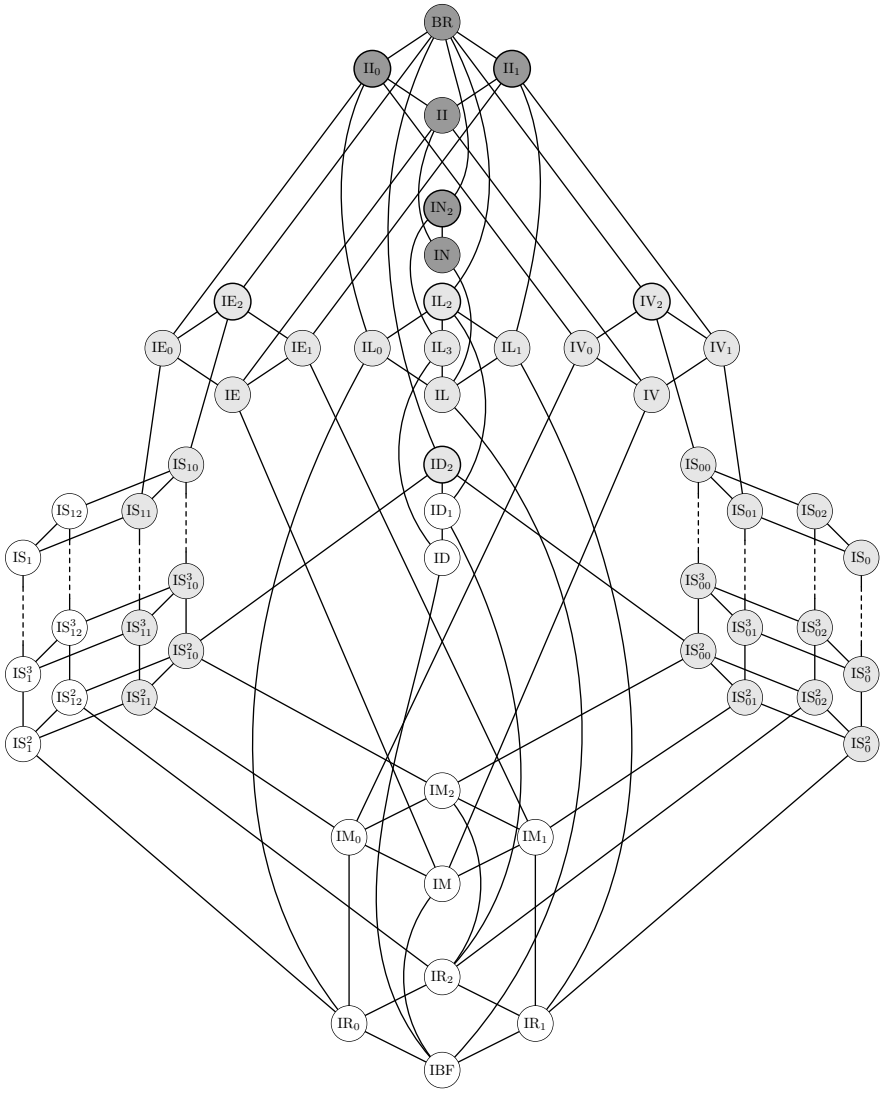


Figure 4: Lattice (under set inclusion) of all Boolean relational clones (co-clones) and their complexity for the inference problem in propositional circumscription. White means in **P**, light grey means **coNP**-complete, and dark grey means $\mathbf{\Pi_2^P}$-complete.

ing functions). It is proved in [6] (Lemma 39) that IS_1 (i.e., $Inv(S_1)$) is the relational clone consisting only of relations of the form $\{0,1\}^n \setminus (1,1,\ldots,1)$. That is, IS_1 consists of all relations corresponding to clauses where all literals are negative. Note that [6] uses Post's original names for the Boolean clones and that S_1 is F_8^∞ in Post's notation.

As we have seen in Theorem 13 relational clones of the form $\langle \Gamma^{id} \rangle$ are of particular importance to us (remember that MIN-INF-CSP(Γ) is of the same complexity as MIN-INF-CSP(Γ^{id})). We call relational clones of this form for idempotent relational clones. It can be deduced that a relational clone Γ is idempotent if and only if $IR_2 \subseteq \Gamma$. Hence we have the following lemma.

Lemma 15 *Let Γ_1 be a Boolean relational clone and Γ_2 the relational clone that is the least upper bound of IR_2 and Γ_1 in Post's lattice of relational clones. Then the following holds,* MIN-INF-CSP(Γ_1) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete) if and only if* MIN-INF-CSP(Γ_2) *is in* **P** *(***coNP***-complete,* $\mathbf{\Pi_2^P}$*-complete).*

Proof: Remember that IR_2 is the relational clone consisting of all relations that are closed under all functions that are both 0-reproducing and 1-reproducing. Thus, $\{\{(0)\},\{(1)\}\} \subseteq IR_2$. If F is a set of Boolean functions containing a non a-reproducing function f, then $\{\{(0)\},\{(1)\}\} \not\subseteq Inv(F)$. Hence, given a relational clone Γ, then $\{\{(0)\},\{(1)\}\} \subseteq \Gamma$ if and only if $IR_2 \subseteq \Gamma$.

Thus it follows that the least upper bound of IR_2 and Γ_1 in the lattice of relational clones is $\langle \Gamma_1^{id} \rangle = \Gamma_2$. Now by Theorem 13 we get that MIN-INF-CSP(Γ_1) is in **P** (**coNP**-complete, $\mathbf{\Pi_2^P}$-complete) if and only if MIN-INF-CSP(Γ_2) is in **P** (**coNP**-complete, $\mathbf{\Pi_2^P}$-complete). □

Next we prove our two new tractable cases of MIN-INF-CSP(Γ). First out is the width-2 affine case.

Lemma 16 MIN-INF-CSP(ID_1) *is in* **P**.

Proof: Remember that the set of constraints that can be expressed by ID_1 can be represented as a system of linear equations over $GF(2)$ where each equation contains at most two variables. Hence each constraint is equivalent to an equation of the following form, $x = c$, $x = y$ or $x = -y$, where $c = 0$ or 1.

Now consider an instance $S = (V, P, Z, Q, C, \psi)$ of Min-Inf-Csp(Γ), where $\Gamma \subseteq ID_1$. We begin by reducing $S = (V, P, Z, Q, C, \psi)$ into an equivalent instance $S' = (V', P', Z', Q', C', \psi')$ such that C' has some special properties. Note that by the symmetry of equations the cases where the roles of x and y are reversed are handled in the same way.

- For all equations of the form $x = c$ we do as follows. If $x = 1$ ($x = 0$) is an equation in C and x ($\neg x$) is a literal in ψ, then every minimal model of C is also a model of ψ and we are done. Otherwise, replace all occurrences of x in C by 1 (0). Remove x from V, P, Z, Q, and remove $\neg x$ (x) from ψ. Finally remove $x = c$ from C.

- For all equations of the form $x = y$ ($x = -y$) where $x \in Q$ and y is present in another equation we do as follows. Replace all occurrences of y in all other equations and ψ by x ($-x$) and remove y from V, P, Z, and Q. Finally remove $x = y$ ($x = -y$) from C.

- For all equations of the form $x = y$ ($x = -y$) where $y \in Z$ and y is present in another equation we do as follows. Replace all occurrences of y in all other equations and ψ by x ($-x$) and remove y from Z and V. Finally remove $x = y$ ($x = -y$) from C.

- For all equations of the form $x = y$ ($x = -y$) where $x \in P$, $y \in P$, and y is present in another equation we do as follows. Replace all occurrences of y in all other equations and ψ by x ($-x$) and remove y from P and V. Finally remove $x = y$ ($x = -y$) from C.

We repeat the above process until no more equations can be removed. It can be realized that in the resulting system of equations C', no variable in Z' occurs in more than one equation, no variable in Q' occurs in an equation together with a variable that occurs in another equation. Moreover, no variable in P' occurs in an equation together with a variable from $P' \cup Z'$ that occurs in another equation.

Now, if ψ' is a tautology, then of course ψ' is true in every minimal model of C', and we are done. So we assume that ψ' is not a tautology. Since ψ' is a clause it is easy to find the (single) assignment of the variables (in ψ') that does not satisfy ψ'. Note that since C' is affine it is easy to decide whether a partial solution can be extended to a total solution, and it is clear that ψ' can be inferred from C' under circumscription if and only if this assignment cannot be extended to a minimal solution to C'. So the question that remains is whether this partial solution (that can be extended to a total solution) can be extended to a minimal solution to C' or not.

Consider the equations of the form $x = y$ or $x = -y$ where neither x nor y is in Q and x is present in ψ'. Then an assignment to x can be extended to a minimal solution to C' if and only if

- x is assigned to 0 in all equations $x = y$ where $x \in P'$ and $y \in P' \cup Z'$ and all equations $x = -y$ where $x \in P'$ and $y \in Z'$, and
- x is assigned to 0 (1) in all equations $x = y$ ($x = -y$) where $x \in Z'$ and $y \in P'$.

□

Now on to the case of clauses where all literals are negative.

Lemma 17 Min-Inf-Csp(IS_1) *is in* **P**.

Proof: We recall that IS_1 consists of all relations corresponding to clauses where all literals are negative. That is, relations of the form $\{0,1\}^n \setminus (1,1,\ldots,1)$.

Now consider an instance $S = (V, P, Z, Q, C, \psi)$ of MIN-INF-CSP(Γ), where $\Gamma \subseteq IS_1$. The cases where ψ is a tautology is trivial, so we assume that ψ is not a tautology. Since ψ is a clause it is easy to find the (single) assignment of the variables that does not satisfy ψ. It is clear that ψ can be inferred from C under circumscription if and only if this assignment cannot be extended to a minimal solution to C. Note that since C consists of Horn clauses (with only negative literals) it is easy to decide whether a partial solution can be extended to a total solution, and it should be clear that such a partial solution can be extended to a minimal solution if and only if all variables in P that are assigned by this partial solution are assigned the value 0. Hence MIN-INF-CSP(IS_1) is in **P**. □

Next we give the complexity of MIN-INF-CSP(Γ) for 8 particular relational clones.

Theorem 18 MIN-INF-CSP(Γ) *is* **coNP***-complete when: 1.* $\Gamma = IS^2_{11}$*; 2.* $\Gamma = IS^2_0$*; 3.* $\Gamma = IL$*; 4.* $\Gamma = IV$*; or 5.* $\Gamma = IE$*.* MIN-INF-CSP(Γ) *is in* **P** *when: 6.* $\Gamma = IS_{12}$*; 7.* $\Gamma = ID_1$*; or 8.* $\Gamma = IM_2$*.*

Proof: We prove each statement separately.

1. The least upper bound of IS^2_{11} and IR_2 is IS^2_{10}. IS^2_{10} contains all Horn clauses with at most 2 variables and it is proved in [2] that MIN-INF-CSP(IS^2_{10}) is **coNP**-complete, hence by Lemma 15 it follows that MIN-INF-CSP(IS^2_{11}) is **coNP**-complete.

2. IS^2_0 contains clauses of the form $(x \vee y)$, that is clauses only consisting of two positive literals. It is proved in [2] that MIN-INF-CSP(IS^2_0) is **coNP**-complete.

3. The least upper bound of IL and IR_2 is IL_2, the set of affine clauses. It is proved in [7] that MIN-INF-CSP(IL_2) is **coNP**-complete, hence by Lemma 15 it follows that MIN-INF-CSP(IL) is **coNP**-complete.

4. The least upper bound of IV and IR_2 is IV_2, the set of dual-Horn clauses. By Case 2. and Lemma 15 it follows that MIN-INF-CSP(IV) is **coNP**-complete.

5. The least upper bound of IE and IR_2 is IE_2, the set of Horn clauses. By Case 1. and Lemma 15 it follows that MIN-INF-CSP(IE) is **coNP**-complete.

6. It is proved in Lemma 17 that MIN-INF-CSP(IS_1) is in **P**. The least upper bound of IS_1 and IR_2 is IS_{12}, hence by Lemma 15 it follows that MIN-INF-CSP(IS_{12}) is in **P**.

7. This is proved in Lemma 16.

8. IM_2 consists of all clauses that are both Horn and dual Horn. It is proved in [2] that MIN-INF-CSP(IM_2) is in **P**.

□

The previous theorem together with the structure of Post's lattice of relational clones and the results proved in [14] yields a trichotomy for the complexity of MIN-INF-CSP(Γ). The results are summarized in terms of the relational clones in Figure 4. A perhaps more intelligible summary is given in the conclusions below.

4 Conclusions

Only one tractable case of the inference problem for propositional circumscription (where Q and Z need not to be empty) is known, namely when the knowledge base only consists of clauses that are both Horn and dual-Horn [2]. We have found two new tractable classes of knowledge bases (width-2 affine clauses, and Horn clauses only containing negative literals) for the inference problem in propositional circumscription. We have proved that the inference problem is **coNP**-hard for all other classes of knowledge bases. This together with the results in [14] gives us the

following trichotomy for the complexity of the inference problem in propositional circumscription:

- **P:** If Γ is Horn and dual-Horn, width-2 affine, or negative Horn;
- **coNP-complete:** If Γ is Horn, dual-Horn, affine, or bijunctive, (and not Horn and dual-Horn, width-2 affine, or negative Horn);
- $\mathbf{\Pi_2^P}$**-complete:** If Γ is neither Horn, dual-Horn, affine, nor bijunctive.

In closing we note that the problem of establishing a trichotomy (as conjectured in [10]) for the complexity of the inference problem for propositional circumscription in the restricted case where all variables must be minimized ($Q = Z = \emptyset$) is still open.

Acknowledgments

The author thanks Steffen Reith for providing the visualization of Post's lattice in Figure 4.

References

[1] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter*, 34(4):38–52, 2003.

[2] M. Cadoli and M. Lenzerini. The complexity of closed world reasoning and circumscription. *Journal of Computer and System Sciences*, pages 255–301, 1994.

[3] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[4] S. Coste-Marquis and P. Marquis. Complexity results for propositional closed world reasoning and circumscription from tractable knowledge bases. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 24–29, 1999.

[5] N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM, Philadelphia, 2001.

[6] V. Dalmau. *Computational Complexity of Problems over Generalized Formulas*. PhD thesis, Department LSI, Universitat Politècnica de Catalanya, Barcelona, 2000.

[7] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is **coNP**-complete. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, pages 451–462, 2003.

[8] T. Eiter and G. Gottlob. Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[9] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[10] L. Kirousis and P. Kolaitis. A dichotomy in the complexity of propositional circumscription. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 71–80, 2001.

[11] L. Kirousis and P. Kolaitis. The complexity of minimal satisfiability problems. *Information and Computation*, 187(1):20–39, 2003.

[12] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[13] J. McCarthy. Circumscription - a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[14] G. Nordh and P. Jonsson. An algebraic approach to the complexity of propositional circumscription. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 367–376, 2004.

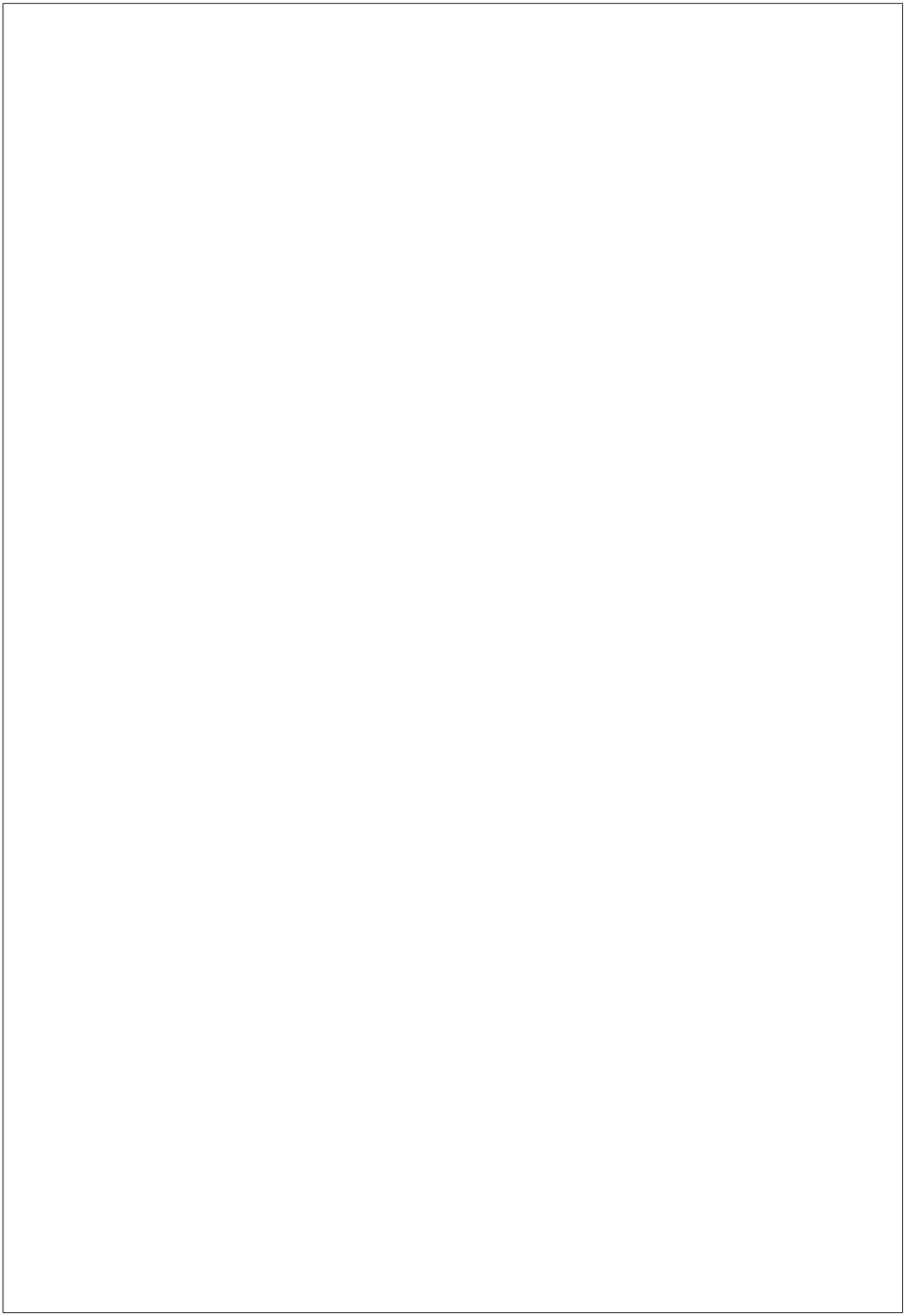
[15] N. Pippenger. *Theories of Computability*. Cambridge University Press, Cambridge, 1997.

[16] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[17] R. Pöschel and L. Kaluznin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.

[18] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.

Paper V



Propositional Abduction is Almost Always Hard

Gustav Nordh and Bruno Zanuttini

Abstract

Abduction is a fundamental form of nonmonotonic reasoning that aims at finding explanations for observed manifestations. Applications of this process range from car configuration to medical diagnosis. We study here its computational complexity in the case where the application domain is described by a propositional theory built upon a fixed constraint language and the hypotheses and manifestations are described by sets of literals. We show that depending on the language the problem is either polynomial-time solvable, **NP**-complete, or $\mathbf{\Sigma_2^P}$-complete. In particular, we show that under the assumption **P**≠**NP**, only languages that are affine of width 2 have a polynomial algorithm, and we exhibit very weak conditions for **NP**-hardness.

1 Introduction

In this paper we investigate the computational complexity of abduction, a method of reasoning extensively studied by Peirce [4]. Abductive reasoning is used to search for explanations of observed manifestations. The importance of this problem to Artificial Intelligence was first emphasized by Morgan [19] and by Pople [20].

The abductive process has demonstrated its practical importance in many domains. In particular, it has been used to formalize processes in medical diagnosis [5], text interpretation [16], system diagnosis [24] and configuration problems [1].

In this paper we are interested in propositional logic-based abduction, i.e., the background knowledge is represented by a propositional theory. Even in this framework several formalizations of the problem have been studied in the literature, depending on the syntactic restrictions imposed to the manifestations and on the hypotheses over which explanations must be formed. We use the following formalization: Given a propositional theory T formalizing a particular application domain, a set M of literals describing a set of manifestations, and a set H of literals containing possible hypotheses, find an explanation for M, that is, a set $E \subseteq H$ such that $T \cup E$ is consistent and logically entails M. This framework is more general than most frameworks studied in the literature; an exception is the definition by Marquis [18], which allows the manifestation to be encoded by any propositional formula.

Example 1 Consider the following example from the domain of motor vehicles, inspired by [6].

$$
\begin{array}{rcl}
T & = & (\neg\text{rich\_mixture} \vee \neg\text{lean\_mixture}) \wedge \\
 & & (\text{rich\_mixture} \rightarrow \neg\text{low\_fuel\_consumption}) \wedge \\
 & & (\text{lean\_mixture} \rightarrow \text{overheating}) \wedge \\
 & & (\text{low\_water} \rightarrow \text{overheating}), \\
H & = & \{\text{rich\_mixture, lean\_mixture, low\_water}\}, \\
M & = & \{\neg\text{low\_fuel\_consumption, overheating}\}.
\end{array}
$$

Then $E = \{\text{rich\_mixture, low\_water}\}$ is an explanation of M, while, e.g., {lean_mixture, low_water} is not because it does not explain low_fuel_consumption, and {rich_mixture, lean_mixture} is not because it is inconsistent with T.

Formalizations of abduction also differ in the literature in the notions of preferred explanations: In this setting, the aim is to compute an explanation that is minimal among all explanations according to some criteria (e.g., inclusion or cardinality). A good overview is given by Eiter and Gottlob [12]. However we are not concerned here with these different notions; we indeed focus on the decision problem asking whether there exists an explanation at all for a given instance. This problem is from now on denoted Abduction.

This problem is well-studied from the computational complexity perspective [14, 23, 12, 10, 9, 26], although with different formalizations of the problem, as mentioned above. It has been proved that it is $\mathbf{\Sigma_2^P}$-complete in the general case [12], while propositional deduction is known to be "only" **coNP**-complete [7]. This negative result raises the problem of identifying restricted cases in which Abduction has computational complexity lower than the general case.

The most natural way to study such restrictions is to study restrictions on the theories representing knowledge bases. This is also the approach followed in most of the previous research in the area. For example, it is known that when the knowledge base T is a conjunction of Horn clauses, then Abduction is **NP**-complete [23].

The ultimate goal of this line of research is to determine the

complexity of every restricted special case of the problem. The first result of this type was proved by Schaefer [22], who proved that the satisfiability problem of conjunctions of Boolean constraints over a fixed language is either in **P** or **NP**-complete, depending on the language. Recall the result due to Ladner [17] stating that if $\mathbf{P} \neq \mathbf{NP}$, then there exist decision problems in **NP** that are neither in **P** nor **NP**-complete. Hence the existence of dichotomy theorems like Schaefer's cannot be taken for granted. The book by Creignou et al. [8] surveys such results.

In this paper we completely classify the complexity of Abduction in Schaefer's framework and exhibit a trichotomy. More precisely, we prove that Abduction is:

- In **P** if the language is affine of width 2,
- Otherwise, **NP**-complete if the language is Horn, dual Horn, bijunctive or affine,
- Otherwise, $\mathbf{\Sigma_2^P}$-complete.

As far as we know, the (only) polynomial case and the minimal **NP**-hard languages that we exhibit are all new results. Note that the only property that guarantees a polynomial algorithm is that of being affine of width 2, i.e., the case where the theory representing the domain is only a conjunction of constants and (in)equalities between two variables. Thus the problem is very hard as soon as sets of literals are allowed as hypotheses and manifestations (instead of atoms or sets of atoms as in, e.g., [26]).

We do not consider directly the complexity of the search problem consisting of computing an explanation or asserting there is none. It is however easily seen that this problem is hard as soon as Abduction is; as for languages that are affine of width 2, our proof that Abduction is in **P** exhibits an efficient algorithm.

2 Preliminaries

The set of all n-tuples of elements from $\{0,1\}$ is denoted by $\{0,1\}^n$. Any subset of $\{0,1\}^n$ is called an *n-ary relation* on $\{0,1\}$. The set of all finitary relations over $\{0,1\}$ is denoted by BR. A *constraint language* over $\{0,1\}$ is an arbitrary finite set $\Gamma \subseteq BR$. Constraint languages are the way in which we impose restrictions on the knowledge bases for the ABDUCTION problem.

A *constraint* over the constraint language Γ is an application of a relation R in Γ to a tuple of variables, written $R(x_1, \ldots, x_n)$ (possibly with repeated variables). An assignment m to the variables *satisfies* the constraint $R(x_1, \ldots, x_n)$ if $(m(x_1), \ldots, m(x_n))$ is a tuple in R. Such a satisfying assignment m is called a *model* of $R(x_1, \ldots, x_n)$. A *theory* T over Γ is a conjunction of constraints over Γ; a *model* m of T is an assignment that satisfies all its constraints simultaneously, denoted $m \models T$. If there is such a model, T is said to be *satisfiable*. Finally, a theory T *entails* a theory T', written $T \models T'$, if every model of T is a model of T'.

The unary relations $F = \{(0)\}$ and $T = \{(1)\}$, which force the value of a variable to 0 and 1 respectively, have a special role for the ABDUCTION problem. We will often use the shorthand notation $\neg x$ and x to denote the constraints $F(x)$ and $T(x)$ respectively. Constraint languages containing F and T will be of particular importance to us. Given a constraint language Γ, the *idempotent constraint language* corresponding to Γ is $\Gamma \cup \{F, T\}$ which is denoted by Γ^{id}.

Given a theory T, $Vars(T)$ denotes the set of all variables that occurs in T. Given a set of variables V, $Lits(V)$ denotes the set of constraints $\bigcup_{x \in V} \{F(x) \cup T(x)\}$ or, equivalently, the set of all literals formed upon the variables in V. The opposite of a literal ℓ is written $\overline{\ell}$. Given a set of literals L, their conjunction is denoted by $\bigwedge L$.

The ABDUCTION problem restricted to the finite constraint language Γ is denoted by ABDUCTION(Γ) and is defined as follows; the problem is said to be *parameterized* by Γ.

Problem 2 (Abduction(Γ)) *An instance* $\mathscr{P}$ *of* Abduction(Γ) *consists of a tuple* (V, H, M, T)*, where:*

- V *is a set of variables,*
- $H \subseteq Lits(V)$ *is the set of hypotheses,*
- $M \subseteq Lits(V)$ *is the set of manifestations, and*
- T *is a theory over* Γ *with* $Vars(T) = V$*.*

The question is whether there exists an explanation for $\mathscr{P}$*, i.e., a set* $E \subseteq H$ *such that* $T \wedge \bigwedge E$ *is satisfiable and* $T \wedge \bigwedge E \models \bigwedge M$*.*

The size of $\mathscr{P} = (V, H, M, T)$ is the total number of occurrences of variables in it.

Recall the following standard restrictions on the constraint languages (see, e.g. [8]).

Definition 3 (restrictions on Γ)

- Γ is *Horn* if every relation R in Γ is the set of models of a CNF formula having at most one unnegated variable in each clause,
- Γ is *dual Horn* if every relation R in Γ is the set of models of a CNF formula having at most one negated variable in each clause,
- Γ is *bijunctive* if every relation R in Γ is the set of models of a CNF formula having at most two literals in each clause,
- Γ is *affine* if every relation R in Γ is the set of models of a system of linear equations over $GF(2)$, the field with two elements,
- Γ is *affine of width* 2 if every relation R in Γ is the set of models of a system of linear equations over $GF(2)$ in which each equation has at most 2 variables.

We emphasize that Γ is assumed to be a finite constraint language in the definition of ABDUCTION(Γ). Moreover, it parameterizes the problem, and is not part of its input. Thus we can assume any convenient presentation of the relations in Γ is stored in a catalog; thus we will assume, e.g., that if Γ is Horn then a theory T over Γ is given as a CNF formula with at most one unnegated variable per clause.

We now recall Schaefer's result, which will be of great importance throughout the paper. Call SAT(Γ) the problem of deciding whether a given theory over Γ is satisfiable. Schaefer completely classified the complexity of SAT(Γ); we only report the result for idempotent constraint languages.

Theorem 4 ([22]) SAT(Γ^{id}) *is in* **P** *if* Γ *is Horn, dual Horn, bijunctive or affine. Otherwise it is* **NP***-complete.*

Finally, we assume that the reader is familiar with the basic notions of complexity theory, but we briefly recall the following. **P** is the class of decision problems solvable in deterministic polynomial time. **NP** is the class of decision problems solvable in nondeterministic polynomial time. $\mathbf{\Sigma_2^P}=\mathbf{NP^{NP}}$ is the class solvable in nondeterministic polynomial time with access to an **NP**-oracle. A problem is **NP**-complete ($\mathbf{\Sigma_2^P}$-complete) if every problem in **NP** ($\mathbf{\Sigma_2^P}$) is polynomial-time reducible to it. Throughout the paper we assume $\mathbf{P}\neq\mathbf{NP}\neq\mathbf{\Sigma_2^P}$.

3 Polynomial Case

The following proposition gives the only (maximal) polynomial case of the ABDUCTION(Γ) problem. Note that, as mentioned in the introduction, its proof gives a polynomial algorithm that *computes* an explanation if there exists one.

Throughout the proof we write $(\ell = a)$ $(a \in \{0,1\})$ for the linear equation $(x = a)$ if $\ell = x$, and for the linear equation

$x = a \oplus 1$ if $\ell = \neg x$. The shorthand $(\ell = \ell')$ is used in the same manner and is equivalent to $(\ell \oplus \ell' = 0)$.

Proposition 5 *If Γ is affine of width 2, then* ABDUCTION(Γ) *is polynomial.*

Proof: Let $\mathscr{P} = (V, H, M, T)$ be an instance of ABDUCTION(Γ), where Γ is affine of width 2. If M' is the set of all manifestations m such that $T \not\models m$, then obviously the explanations of $\mathscr{P}$ are exactly those of (V, H, M', T). Since $T \models m$ can be decided efficiently with Gaussian elimination on $T \wedge (m = 0)$, we assume $M' = M$.

For every manifestation $m \in M$ write E_m for the set of literals $\{h \in H \mid T \models (h = m)\}$; once again every E_m can be computed efficiently with Gaussian elimination on $T \wedge (h \oplus m = 1)$ for every $h \in H$. We show that $\mathscr{P}$ has an explanation if and only if $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable and no E_m is empty. Since the satisfiability of $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ can be decided efficiently with again Gaussian elimination (on $T \wedge \bigwedge_{m \in M} \bigwedge_{h \in E_m} (h = 1)$), this will conclude the proof.

Assume first $\mathscr{P}$ has an explanation E. Then $T \wedge \bigwedge M$ is consistent; since for every $m \in M$ and $h \in E_m$ we have $T \models (h = m)$, we also have $T \wedge \bigwedge M \models T \wedge \bigwedge_{m \in M} \bigwedge E_m$, and thus $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable. Now we also have $\forall m \in M, T \models (\bigwedge E \rightarrow m)$. Since T is affine of width 2 it is bijunctive, thus every clause entailed by T can be minimized into a bijunctive one; since $T \wedge \bigwedge E$ is satisfiable and $T \not\models m$, the only possibility is a minimal clause of the form $h \rightarrow m$ with $h \in E$. But since T is affine this implies that $m \rightarrow h$ also is an implicate of it, and finally we have $T \models (h \oplus m = 0)$, which shows that E_m is nonempty. For more details we refer the reader to [25]. Conversely, assume $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable and no E_m is empty. Then since $(h \oplus m = 0) \models (h \rightarrow m)$ it is easily seen that $\bigwedge_{m \in M} \bigwedge E_m$ is an explanation for $\mathscr{P}$. □

4 NP-complete Cases

We now exhibit the **NP**-complete cases of ABDUCTION(Γ). Since $T \wedge \bigwedge E \models \bigwedge M$ holds if and only if $T \wedge \bigwedge E \wedge \overline{\ell}$ is unsatisfiable for every $\ell \in M$, the following result is obvious.

Lemma 6 *If* SAT(Γ^{id}) *is in* **P**, *then* ABDUCTION(Γ) *is in* **NP**.

We first establish **NP**-completeness for particular languages. In Section 5 we will establish more general results.

Let $R_{\neg x \vee y} = \{(1,1),(0,1),(0,0)\}$, i.e., the set of models of $\neg x \vee y$. Observe that $R_{\neg x \vee y}$ is both Horn and dual Horn.

Proposition 7 ABDUCTION($\{R_{\neg x \vee y}\}$) *is* **NP**-*complete.*

Proof: Membership in **NP** follows directly from Theorem 4 and Lemma 6. As regards hardness, we give a reduction from the **NP**-complete problem MONOTONE-SAT [15], i.e., the satisfiability problem for CNF formulas where each clause contains either only positive literals or only negative literals. Let $\psi = \bigwedge_{i=1}^{k} N_i \wedge \bigwedge_{i=1}^{\ell} P_i$ be such a formula, where each N_i is a negative clause, written $N_i = \bigvee_{j=1}^{\nu_i} \neg x_i^j$, and every P_i is a positive clause written $P_i = \bigvee_{j=1}^{\pi_i} y_i^j$. We build the instance $\mathscr{P} = (V, H, M, T)$ of ABDUCTION($R_{\neg x \vee y}$) where:

- $V = \{\gamma_i \mid i = 1, \ldots, k\} \cup \{\delta_i \mid i = 1, \ldots, \ell\} \cup Vars(\psi)$; $\neg\gamma_i$ will intuitively represent satisfaction of clause N_i and δ_i, that of clause P_i
- $T = \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{\nu_i} (x_i^j \vee \neg\gamma_i) \wedge \bigwedge_{i=1}^{\ell} \bigwedge_{j=1}^{\pi_i} (\neg y_i^j \vee \delta_i)$; this encodes the implications $\neg x_i^j \rightarrow \neg\gamma_i$ and $y_i^j \rightarrow \delta_i$, i.e., the fact that N_i (P_i) is satisfied if at least one of the $\neg x_i^j$ (y_i^j), $j \in \{1, \ldots, \nu_i\}$ ($j \in \{1, \ldots, \pi_i\}$) is
- $H = Lits(Vars(\varphi))$
- $M = \bigwedge_{i=1}^{k} \neg\gamma_i \wedge \bigwedge_{i=1}^{\ell} \delta_i$.

Obviously enough, the theory T is over the language $\{R_{\neg x \vee y}\}$. Now it is easily seen that if ψ has at least one model, say m, then $E = \{\ell \mid m \models \ell\}$ is an explanation for $\mathscr{P}$, and that if $\mathscr{P}$ has an explanation E, then any assignment m to $Vars(\psi)$ with $\forall \ell \in E, m \models \ell$ is a model of ψ. □

Similarly, we now prove that ABDUCTION(Γ) is **NP**-complete if Γ is the singleton language containing only the relation $R_{x \vee y} = \{(1,1),(1,0),(0,1)\}$.

Proposition 8 ABDUCTION($\{R_{x \vee y}\}$) *is* **NP***-complete.*

Proof: Since $R_{x \vee y}$ is dual Horn, membership in **NP** follows from Theorem 4 and Lemma 6. As for hardness, we give a reduction from MONOTONE-SAT (see the proof of Proposition 7), where positive clauses in an instance of this problem are restricted to contain at most two literals. Thus an instance of this problem is a formula of the form $\psi = \bigwedge_{i=1}^{k} N_i \wedge \bigwedge_{i=1}^{\ell}(y_i^1 \vee y_i^2)$, where the y_i^j's are variables and every N_i is a negative clause written $N_i = \bigvee_{j=1}^{\nu_i} \neg x_i^j$. The **NP**-completeness of this restricted problem follows directly from Schaefer's result [22].

Given an instance ψ of MONOTONE-SAT as above we build the instance $\mathscr{P} = (V, H, M, T)$ of ABDUCTION($\{R_{x \vee y}\}$) where:

- $V = \{\gamma_i \mid i = 1, \ldots, k\} \cup Vars(\psi)$; γ_i will intuitively represent satisfaction of clause N_i
- $T = \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{\nu_i}(x_i^j \vee \gamma_i) \wedge \bigwedge_{i=1}^{\ell}(y_i^1 \vee y_i^2)$; clauses $(x_i^j \vee \gamma_i)$ encode the implications $\neg x_i^j \rightarrow \gamma_i$
- $H = Lits(Vars(\varphi))$
- $M = \bigwedge_{i=1}^{k} \gamma_i$.

We show that ψ has a model if and only if $\mathscr{P}$ has an explanation. Assume first that ψ has a model m; then it is easily seen that $E = \{\ell \mid m \models \ell\}$ is an explanation for $\mathscr{P}$. Now assume $\mathscr{P}$ has an

explanation E. Then from $\forall i = 1, \ldots, k, T \wedge \bigwedge E \models \gamma_i$ it follows that for every $i = 1, \ldots, k$ E contains at least one $\neg x_i^j$, and thus any assignment satisfying E satisfies every negative clause of ψ; on the other hand, since $T \wedge \bigwedge E$ is satisfiable there is a model m of $T \wedge \bigwedge E$ that satisfies every positive clause of ψ, and this m thus satisfies ψ. □

The following proposition can be shown with the same proof as Proposition 8 with all variables renamed.

Proposition 9 *Let* $R_{\neg x \vee \neg y} = \{(1,0),(0,1),(0,0)\}$. *Then the* ABDUCTION$(\{R_{\neg x \vee \neg y}\})$ *problem is* **NP***-complete.*

We finally prove that ABDUCTION(Γ) is **NP**-complete for a particular affine language. This will be achieved by reducing to it another important problem in nonmonotonic reasoning, namely the inference problem for propositional circumscription. A model $m = (m_1, \ldots, m_n)$ of a formula φ is said to be a *minimal model* of φ if there is no model $m' = (m'_1, \ldots, m'_n)$ of φ such that $m \neq m'$ and $\forall i = 1, \ldots, n, m'_i \leq m_i$.

Durand and Hermann proved that the inference problem for propositional circumscription of affine formulas is **coNP**-complete. In the process, they proved the following theorem.

Theorem 10 ([11]) *The problem of deciding whether there is a minimal model of a given affine formula* φ *that does not satisfy a given negative clause* $(\neg q_1 \vee \cdots \vee \neg q_n)$ $(\forall i = 1, \ldots, n, q_i \in Vars(\varphi))$ *is* **NP***-complete.*

A careful reading of their proof shows that the theorem remains true even if the linear equations in the input affine formulas are all restricted to contain at most 6 variables. We thus define the language Γ_{6aff} to be the set of all k-ary affine relations with $k \leq 6$. Obviously, Γ_{6aff} is finite, which is necessary for problem ABDUCTION(Γ_{6aff}) to be well-defined.

Proposition 11 ABDUCTION(Γ_{6aff}) *is* **NP***-complete.*

Proof: Membership in **NP** follows directly from Theorem 4 and Lemma 6. As for hardness, let φ be a formula and $q_1, \ldots, q_n \in Vars(\varphi)$. We show that the clause $(\neg q_1 \vee \cdots \vee \neg q_n)$ is false in some minimal model of φ if and only if the abduction problem with $T = \varphi$, $M = \{q_1, \ldots, q_n\}$, and $H = \{\neg x \mid x \in Vars(\varphi) \setminus \{q_1, \ldots, q_n\}\}$ has an explanation, which will conclude by Theorem 10 and the above remark.

Assume first that $(\neg q_1 \vee \cdots \vee \neg q_n)$ is false in a minimal model m of φ. Define E to be $\{\neg x_i \mid m \models \neg x_i\}$. Since $m \models \varphi$ by assumption and $m \models \bigwedge E$ by construction, $\varphi \wedge \bigwedge E$ is satisfiable. Now assume for sake of contradiction that there is m' satisfying $\varphi \wedge \bigwedge E \wedge (\neg q_1 \vee \cdots \vee \neg q_n)$. Then since $m' \models \bigwedge E$ and E is negative we get $\forall x \in E, m'(x) \leq m(x)$; now for $x \in Vars(\varphi) \setminus Vars(E)$ we have by assumption $m(x) = 1$ and thus $m'(x) \leq m(x)$ again. Finally, we have $\exists q_i, m'(q_i) = 0 < 1 = m(q_i)$, which contradicts the minimality of m. Thus $\varphi \wedge \bigwedge E \models (q_1 \wedge \cdots \wedge q_n)$ and E is an explanation.

Conversely, assume that E is an explanation. Then $\varphi \wedge \bigwedge E$ is satisfiable; write m for one of its minimal models. By assumption the formula $\varphi \wedge \bigwedge E \wedge (\neg q_1 \vee \cdots \vee \neg q_n)$ is unsatisfiable, thus $m \not\models (\neg q_1 \vee \cdots \vee \neg q_k)$. We also have $m \models \varphi$ by assumption. Finally, assume for sake of contradiction that m is not a minimal model of φ, and let m' be such that $m' \models \varphi$, $m' \leq m$ and $m' \neq m$. Then since E is negative (because H is) and $m \models \bigwedge E$ we have $m' \models \bigwedge E$, thus $m' \models \varphi \wedge \bigwedge E$, which contradicts the minimality of m among the models of $\varphi \wedge \bigwedge E$. □

5 Classification

We finally put together the results in the previous sections for obtaining our complete classification. The concept of a *relational clone* is central to our approach.

Definition 12 (relational clone) Let $\Gamma \subseteq BR$. The relational

clone of Γ is written $\langle\Gamma\rangle$ and is the set of all relations that can be expressed using relations from $\Gamma \cup \{=\}$ ($=$ is the equality relation on $\{0, 1\}$), conjunction, and existential quantification.

Intuitively, the constraints over $\langle\Gamma\rangle$ are those which can be simulated by constraints over Γ.

The following result states that when studying the complexity of the ABDUCTION(Γ) problem it is enough to consider constraint languages that are relational clones.

Lemma 13 *Let Γ be a finite constraint language and $\Gamma' \subseteq \langle\Gamma\rangle$ finite. Then* ABDUCTION(Γ') *is polynomial-time reducible to* ABDUCTION(Γ).

Proof: Let (V', H', M', T') be an instance of ABDUCTION(Γ'). By the definition of a relational clone there is a set of variables W disjoint from V' and a theory $T_=$ over $\Gamma \cup \{=\}$ with $Vars(T_=) = V_= = V' \cup W$ and such that T' is logically equivalent to the formula $\exists W, T_=$. Since W is disjoint from V' there is no variable occurring in H' or M' and in W at the same time, and it is then easily seen that the abduction problem $(V_=, H', M', T_=)$ has an explanation if and only if (V', H', M', T') has one. Now for every constraint $(x_i = x_j)$ $(i < j)$ it is enough to replace x_j with x_i everywhere in $V_=, T_=, M'$ and H' and to remove the constraint from $T_=$ for obtaining a still equivalent instance (V, H, M, T) of ABDUCTION(Γ), which concludes. □

We can reduce even further the set of constraints languages to be considered, namely to idempotent ones.

Lemma 14 *Let Γ be a finite constraint language.* ABDUCTION(Γ^{id}) *is polynomial time reducible to* ABDUCTION(Γ).

Proof: Let $\mathscr{P} = (V, H, M, T)$ be an instance of ABDUCTION(Γ^{id}). We build an instance $\mathscr{P}' = (V, H', M', T')$ of ABDUCTION(Γ) by removing every constraint $F(x)$ or $T(x)$ from T and adding it to H and M. It is then easy to see that

(V, H', M', T') has an explanation if and only if (V, H, M, T) has one. □

Given these two lemmas, our classification of the complexity of the ABDUCTION(Γ) problem heavily relies on Post's remarkable classification of all Boolean relational clones [21]. Post proved in particular that the relational clones form a lattice under set inclusion. An excellent introduction to Post's lattice can be found in the recent survey articles [2, 3].

Lemmas 13 and 14 say that for any finite $\Gamma' \subseteq \langle \Gamma^{id} \rangle$, ABDUCTION($\Gamma'$) is polynomial-time reducible to ABDUCTION(Γ). In other words, when studying the complexity of ABDUCTION(Γ) it is enough to consider constraint languages that are idempotent relational clones. The lattice of all idempotent Boolean relational clones is given on Figure 5.

Those that are most relevant to our classification are the following:

- BR, the set of all Boolean relations,
- IE_2, the set of all Horn relations,
- IV_2, the set of all dual Horn relations,
- ID_2, the set of all bijunctive relations,
- IL_2, the set of all affine relations,
- ID_1, the set of all affine relations of width 2,
- IM_2, the set of all relations that are Horn and dual Horn.

Thus, according to Post's lattice there is only one idempotent relational clone that is not Horn, not dual Horn, not affine, and not bijunctive, namely the relational clone consisting of all Boolean relations BR. Hence the following result follows intuitively from the result due to Eiter and Gottlob [12] stating that ABDUCTION is $\mathbf{\Sigma_2^P}$-complete for the general case of theories given by CNF formulas.

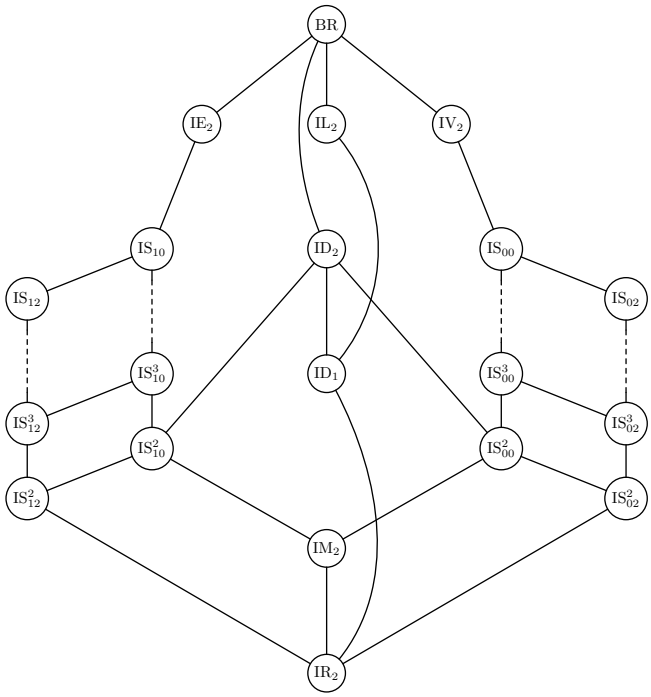


Figure 5: Lattice of all idempotent Boolean relational clones.

Proposition 15 *If* Γ *is not Horn, not dual Horn, not affine and not bijunctive then* ABDUCTION(Γ) *is* $\mathbf{\Sigma_2^P}$*-complete.*

Proof: It is well-known that for any CNF formula ψ there is a set of variables W disjoint from $Vars(\psi)$ and a CNF formula ψ' over $Vars(\psi) \cup W$ with at most 3 variables per clause such that the formulas ψ and $\exists W \psi'$ are logically equivalent. That fact together with a proof similar to that of Lemma 13 show that the abduction problem for general CNF theories reduces to ABDUCTION(Γ_3), where Γ_3 is the (finite) set of all ternary relations. Since Γ_3 is not Horn, not dual Horn, not affine and not bijunctive, we have $\langle \Gamma_3^{id} \rangle = BR$, and Lemma 13 and Lemma 14 concludes. □

We are finally able to classify the complexity of ABDUCTION(Γ).

Theorem 16 (classification) *Let* Γ *be a constraint language. Then, the* ABDUCTION(Γ) *problem is:*

- *In* **P** *if* Γ *is affine of width* 2*,*
- *Otherwise,* **NP***-complete if* Γ *is Horn, dual Horn, bijunctive or affine,*
- *Otherwise,* $\mathbf{\Sigma_2^P}$*-complete.*

Proof: Proposition 5 shows the result for languages that are affine of width 2. Now it can be seen that the relations $R_{\neg x \vee y}$, $R_{x \vee y}$ and $R_{\neg x \vee \neg y}$ of Propositions 7, 8 and 9 are in the relational clones IM_2, IS^2_{02} and IS^2_{12}, respectively; this can be verified by checking that they are invariant under the operations defining the corresponding clones (for more details see [2, 3]). Moreover, the language Γ_{6aff} of Proposition 11 is affine, thus it is in IL_2. Consequently, Figure 5 shows that the minimal idempotent relational clones that are not affine of width 2, namely IM_2, IS^2_{02}, IS^2_{12} and IL_2 are **NP**-complete. On the other hand, we know from Theorem 4 and Lemma 6 that the relational clones IL_2 (affine), ID_2 (bijunctive), IE_2 (Horn) and IV_2 (dual Horn) are in **NP**. Thus ABDUCTION(Γ) is **NP**-complete when $\langle \Gamma^{id} \rangle$ contains IM_2, IS^2_{02}, IS^2_{12} or IL_2 and is contained in IL_2, ID_2, IE_2, or IV_2. This covers exactly the languages that are Horn, dual Horn, bijunctive, or affine and that are not affine of width 2.

Finally, Proposition 15 concludes the proof. □

6 Discussion and Future Work

We have completely classified the complexity of propositional abduction in Schaefer's framework when manifestations and hypotheses are described by sets of literals. This result can prove

useful in helping the designers of knowledge based systems to deal with the expressivity/tractability tradeoff when choosing a language for their system. Our result indeed completes the picture of the complexity of reasoning for propositional constraint languages. In particular, we have shown that this problem is very hard, in the sense that only languages that are affine of width 2 allow for polynomial abduction. Also note that in many cases **NP**-hardness remains even when restricting further the problem; e.g., to $H = Lits(V \backslash Vars(M))$ (see the proofs of Propositions 7–9).

It is important to note that the complexity of abduction for a constraint language given in extension (i.e., by the set of all tuples in every relation) can be determined efficiently; the case of Theorem 16 in which a language falls can indeed be determined efficiently by using the closure properties of the concerned co-clones (see, e.g., [3]).

It would be interesting to try to extend this work into at least three directions. First of all, besides the problem of deciding the existence of an explanation for a given instance, of great importance are the problems of relevance and necessity, which ask whether a given hypothesis is part of at least one (resp. of all) preferred explanation(s). These problems involve a preference criterion which can have a great impact on their complexity; for more details we refer the reader to [12]. Hence, it would be interesting to investigate the complexity of these problems. In the same vein, Eiter and Makino recently studied the problem of enumerating all the explanations of a Horn abduction problem [13]; it would be interesting to try to extend their work to other classes of formulas.

Secondly, although Schaefer's framework is quite general, there are restrictions on propositional formulas that it cannot express. For instance, Eshghi [14] and del Val [9] study such restrictions that yield polynomial cases of the abduction problem. It would thus be of great interest to try to identify still more

such tractable classes.

Finally, it would be interesting to study the case where the domains of variables are more general, e.g., for conjunctions of constraints over finite domains.

References

[1] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSPs - application to configuration. *Artificial Intelligence*, 135(1-2):199–234, 2002.

[2] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter*, 34(4):38–52, 2003.

[3] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part II: Constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35(1):22–35, 2004.

[4] J. Buchler. *Philosophical Writings of Peirce*. Dover, New York, 1955.

[5] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49:25–60, 1991.

[6] L. Console, D. Theseider Dupre, and P. Torasso. On the relationship between abduction and deduction. *J. Logic and Computation*, 1(5):661–690, 1991.

[7] S. Cook. The complexity of theorem proving procedures. In *Proc. STOC'71*, pages 151–158, 1971.

[8] N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of Boolean constraint satisfaction problems*. SIAM Monographs on Discrete Mathematics and Applications, 2001.

[9] A. del Val. The complexity of restricted consequence finding and abduction. In *Proc. AAAI'00*, pages 337–342, 2000.

[10] A. del Val. On some tractable classes in deduction and abduction. *Artificial Intelligence*, 116(1-2):297–313, 2000.

[11] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is **coNP**-complete. In *Proc. STACS'03*, pages 451–462, 2003.

[12] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. of the ACM*, 42(1):3–42, 1995.

[13] T. Eiter and K. Makino. On computing all abductive explanations. In *Proc. AAAI'02*, pages 62–67, 2002.

[14] K. Eshghi. A tractable class of abduction problems. In *Proc. IJCAI'93*, pages 3–8, 1993.

[15] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of* **NP***-Completeness*. Freeman, San Francisco, 1979.

[16] J. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[17] R. Ladner. On the structure of polynomial time reducibility. *J. of the ACM*, 22:155–171, 1975.

[18] P. Marquis. *Handbook of Defeasible Reasoning and Uncertainty Management Systems (DRUMS)*, volume 5, chapter Consequence finding algorithms, pages 41–145. Kluwer Academic, 2000.

[19] C. Morgan. Hypothesis generation by machine. *Artificial Intelligence*, 2:179–187, 1971.

[20] H. Pople. On the mechanization of abductive logic. In *Proc. IJCAI'73*, pages 147–152, 1973.

[21] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[22] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC'78*, pages 216–226, 1978.

[23] B. Selman and H. Levesque. Abductive and default reasoning: A computational core. In *Proc. AAAI'90*, pages 343–348, 1990.

[24] M. Stumptner and F. Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127:1–29, 2001.

[25] B. Zanuttini and J.-J. Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6):335–339, 2002.

[26] B. Zanuttini. New polynomial classes for logic-based abduction. *J. Artificial Intelligence Research*, 19:1–10, 2003.