New Algebraic Tools for Constraint Satisfaction

Henning Schnoor, Ilka Schnoor

Institut für Theoretische Informatik, Leibniz Universität Hannover, Appelstr. 4, 30167 Hannover, Germany. {henning, ilka}.schnoor@thi.uni-hannover.de

Abstract. The Galois connection involving polymorphisms and co-clones has received a lot of attention in regard to constraint satisfaction problems. However, it fails if we are interested in a reduction giving equivalence instead of only satisfiability-equivalence. We show how a similar Galois connection involving weaker closure operators can be applied for these problems. As an example of the usefulness of our construction, we show how to obtain very short proofs of complexity classifications in this context.

Keywords. computational complexity, constraints, partial polymorphisms

1 Introduction

Constraint satisfaction problems (CSPs) play an important role in computational complexity theory. In particular, the non-uniform version of the problem, $\mathsf{CSP}(\Gamma)$ has been studied. In this context, a set of relations Γ, called a constraint language, over a finite domain is fixed, and so-called Γ-formulas, i.e., conjunctions of applications of relations from Γ to variables, are studied.

For the Boolean domain, these problems generalize many well-known restrictions of the propositional satisfiability problem, for example 2SAT, 3SAT, $\mathsf{Horn}-\mathsf{SAT}$, etc. CSPs can also be used to express various problems related to graphs (like search, colorability problems, and others) as well as database queries and various scheduling problems. Many combinatorial problems can be expressed in this context, and therefore constraint satisfaction problems can be seen as embodying the quintessence of the combinatorial aspects of complexity theory.

The study of constraint related problems in computational complexity started with Thomas Schaefer's seminal paper [Sch78]. He showed that for Boolean constraint languages Γ, the complexity of the satisfiability problem $\mathsf{CSP}(\Gamma)$ (the problem to determine if a given Γ-formula is satisfiable) is dichotomic: for a given Γ, this problem can be solved in P, or is already complete for NP, hence avoiding the infinitely many complexity degrees between these two classes which are known to exist due to a classic result by Ladner [Lad75]. Since Schaefer's work, many results have been obtained about the complexity of the problem $\mathsf{CSP}(\Gamma)$ for non-Boolean domains, leading to a proof of a similar dichotomy theorem for the three-element case by Andrei Bulatov [Bul06].

One of the most successful techniques to obtain results on the complexity of problems related to constraint formulas has been the application of tools from Universal Algebra. In particular, a Galois connection between constraint languages and functions on finite domains has been studied, and plays a crucial role in the above-mentioned dichotomy theorems.

This Galois connection can be used to show that if two constraint languages Γ_1 and Γ_2 have the same algebraic closure properties, then the complexity of the problems $\mathsf{CSP}(\Gamma_1)$ and $\mathsf{CSP}(\Gamma_2)$ is the same (up to $\leq_m^{\log}$-reductions [ABI+05]). The Galois connection relates the *expressive power* of a constraint language to its set of *polymorphisms*, i.e., algebraic closure properties. By a work of Emil Post [Pos41], the structure of the polymorphism sets occurring here, so-called *clones*, is well known for the Boolean case. Hence the Galois connection can be used to transfer results from these well-known classes to the complexity of constraint satisfaction problems.

Technically, the Galois connection gives a procedure transforming Γ_1-formulas into equivalent Γ_2-formulas, where in the newly constructed Γ_2-formulas, additional existentially quantified variables, and equality clauses may occur.

It is evident that this transformation preserves satisfiability, as in the satisfiability problem, new free variable in a formula can be regarded as existentially quantified, and the occurring equality clauses can be dealt with using variable identification. Similarly, in many cases where the newly introduced variables can be "hidden" in some way (for example, in many cases of problems involving quantified formulas), the Galois connection can easily be seen to preserve all interesting properties of the involved formulas. In these cases, we say that the Galois connection holds "a priori."

However, for problems different from satisfiability, this transformation is not sufficient. In the case of problems like counting and enumeration of solutions for constraint formulas, it is evident that the newly introduced variables are problematic, as they can change the set and the number of solutions to a given formula. In [SS06], it was shown that this feature of the Galois connection makes it inapplicable for the enumeration problem over non-Boolean domains.

Similarly, if the goal is to determine whether two formulas are equivalent, the introduction of equality clauses is a problem.

The identification of equality-constrained variables is also problematic when considering low complexity classes. In [ABI+05], it was shown that it is exactly the introduction of equality clauses which makes it impossible to refine the logspace reduction given by the Galois connection to a reduction computable in AC^0.

From these considerations, it is evident that weaker closure operators are of interest when looking at questions in the constraint context which are different from determining the $\leq_m^{\log}$-degree of complexity of the satisfiability problem. The obvious approach is to consider restricted closures, where the problematic features from the Galois connection mentioned above, i.e., introduction of new variables and equality clauses, are absent. This closure operator has been studied

in mathematics already in the 1960s, and a similar relation to closure properties of the involved relations was proven. This refined Galois connection tells us that instead of looking at the set of polymorphisms, we need to consider the set of partial polymorphisms. These form a structure which is a refinement of the clone structure exhibited by Post for the Boolean case. In particular, we know that instead of the countably many classes arising in Post's classification, there is an uncountable number of *partial clones*. In other words, the step from the usual Galois connection to the refinement comes with the price of additional complexity of the mathematical structures involved.

In light of these considerations, it is surprising to see that for many of the problems where the mentioned problems occur, a complexity classification, when achieved, often follows the lines of the well-known Galois connection. Among other examples, this is the case for the already mentioned problems of enumeration and counting of solutions for constraint formulas in the Boolean case. Hence, for these problems, the usual Galois connection holds "a posteriori."

The question for which problems the Galois connection holds has been of considerable interest in the constraint context. While proofs for an "a priori"-application of the Galois connection are very often close to trivial, there are many cases where the fact that the Galois connection holds only follows from a complete complexity classification of the involved problem (e.g. enumeation [CH97] and equivalence [BHRV02] problems in the Boolean case).

The contribution of this paper is twofold: first, we introduce the terminology of the refined Galois connection involving partial polymorphisms. Second, we give an answer to the above question: we give a method which can be used to determine if for a given problem, the Galois connection holds "a posteriori." The paper is structured as follows: In Section 2, we give the necessary definitions and initial results in our context. In Section 3, we state the Galois connection and give an elementary proof. In Section 4, we exhibit a set of relations which are fundamental to the answer to the above question about the Galois connection holding "a posteriori."

Finally, we demonstrate our technique with giving very short re-proofs of the above mentioned results about enumeration and equivalence for Boolean constraint languages, where almost the entire technical difficulty of the original proofs can be avoided using our algebraic results.

2 Preliminaries

We first briefly repeat the basic definitions in the constraint satisfaction context. For a relation R, let $\mathrm{ar}(R)$ denote its arity. In this paper, we only consider relations over a finite domain. For a tuple $\mathbf{v}$, the value $\mathbf{v}[i]$ denotes its i-th component. For any set D, let T_D denote the total finitary functions on D. A *constraint language* Γ is a finite set of finitary relations over a finite domain D. For a (not necessarily finite) set of relations Γ, a Γ-formula is a conjunction of the form

$$\varphi = \bigwedge_{i=1}^{n} R_i(x_1^i, \ldots, x_{k_i}^i),$$

where R_i are k_i-ary relations from Γ, and the x_j^i are (not necessarily distinct) variables. For a single-element constraint language $\{R\}$, we often simply speak about R-formulas, etc. We denote the set of variables of φ with $\mathrm{VAR}(\varphi)$. An assignment $I\colon \mathrm{VAR}(\varphi) \to D$ *satisfies* φ or is a *solution* of φ, if for all $i \in \{1, \ldots, n\}$, it holds that $(I(x_1^i), \ldots, I(x_{k_i}^i))$ is a tuple from R_i. We say that a constraint language is *Boolean*, if the domain D has cardinality 2.

There is a close relationship between formulas and relations, as each formula defines the relation of its satisfying assignments. We say that a formula *represents* or *expresses* the relation of its solutions.

We now define three closure operators for relations and constraint languages. The first one, $\langle . \rangle$, is the one usually considered in the constraint context, which has successfully been applied to classify the complexity of satisfiability problems. The other two are the refinements where we disallow the introduction of equality clauses and/or new variables.

Definition 2.1. *Let Γ be a set of relations.*

- $\langle \Gamma \rangle$ *is the set of relations which can be expressed as a $\Gamma \cup \{=\}$-formulas with additional existentially quantified variables.*
- $\langle \Gamma \rangle_{\nexists}$ *is the set of relations which can be expressed as a $\Gamma \cup \{=\}$-formulas.*
- $\langle \Gamma \rangle_{\nexists,\neq}$ *is the set of relations which can be expressed as a Γ-formulas.*

In the notation above, $\neq$ does not mean that the closure operator is allowed to use the inequality predicate, but that the operator is not allowed to use the equality predicate, similarly $\nexists$ means that the introduction of existentially quantified variables is not allowed. It is evident that for any set of relations Γ, the inclusion $\langle \Gamma \rangle_{\nexists,\neq} \subseteq \langle \Gamma \rangle_{\nexists} \subseteq \langle \Gamma \rangle$ holds.

The operators defined above are closure operators in the usual mathematical sense, i.e., for an operator $C \in \{\langle . \rangle, \langle . \rangle_{\nexists}, \langle . \rangle_{\nexists,\neq}\}$ and sets of relations Γ_1 and Γ_2, it holds that $\Gamma_1 \subseteq C(\Gamma_1)$, if $\Gamma_1 \subseteq \Gamma_2$, then $C(\Gamma_1) \subseteq C(\Gamma_2)$ also holds, and finally $C(C(\Gamma_1)) = \Gamma_1$. Sets Γ satisfying $C(\Gamma) = \Gamma$ are also called C-closed. The $\langle . \rangle$-closed sets are called *co-clones*.

It is obvious that each closure operator $C \in \{\langle . \rangle, \langle . \rangle_{\nexists}, \langle . \rangle_{\nexists,\neq}\}$ gives rise to a formula transformation. Let $\Gamma_1 \subseteq C(\Gamma_2)$. Then a Γ_1-formula can be transformed into an equivalent Γ_2-formula by replacing a clause $R(x_1, \ldots, x_n)$ for some relation $R \in \Gamma_1$ with its Γ_2-implementation, possibly adding new existentially quantified variables and/or equality clauses, depending on the closure operator. To remove these features and produce an actual Γ_2-formula, we proceed as follows: in the case of equality clauses $(x = y)$, we simply replace every occurrence of the variable y with x, and remove the equality clause. For existentially quantified variables, we simply drop the quantifiers, leaving the variables free. The new Γ_2-formula constructed in this way is not necessarily equivalent to the old one,

but in many cases, the transformation preserves many properties of the formulas which we are interested in.

In the following, let PROBLEM be a computational problem where the input instances are propositional formulas, and let PROBLEM(Γ) be its restriction to Γ-formulas as inputs. As examples to explain our results and techniques, we introduce the following problems: CSP is the problem to determine whether a formula is satisfiable. The problem EQUIV is the problem to determine whether two input formulas are equivalent, i.e., have the same set of solutions. The problem ENUM is the problem to enumerate, for a given propositional formula, its set of satisfying solutions. It is obvious that if P $\neq$ NP, then all these problems are computationally intractable: the problems CSP and EQUIV are NP-complete resp. coNP-complete. The problem ENUM is not a decision problem and hence cannot easily be related to these standard complexity classes, but it is obvious that any efficient enumeration algorithm can be used to decide the satisfiability problem in polynomial time for any reasonable notion of "efficiency."

For the problem PROBLEM, we say that the operator C *can be applied a to* PROBLEM, if for any two constraint languages Γ_1 and Γ_2 such that $C(\Gamma_1) = C(\Gamma_2)$, the complexity of the problem PROBLEM (Γ_1) is the same as the complexity of PROBLEM (Γ_2).

Such results can very often be obtained "for free," since in many cases transforming Γ_1-formulas into Γ_2-formulas in the way described above gives a reduction from the problem PROBLEM (Γ_1) to PROBLEM (Γ_2). It should be noted that depending on the type of problem that PROBLEM is, the notion of reduction here differs. If PROBLEM is a decision problem, then, as shown in [ABI+05], the closure operator $\langle . \rangle$ leads to a $\leq_m^{\log}$-reduction, and the closure operators $\langle . \rangle_{\not\equiv,\neq}$ and $\langle . \rangle_{\not\equiv}$ lead to an $\leq_m^{\mathrm{AC}^0}$-reduction (these results are not explicitly stated in this notation in [ABI+05], but follow immediately from the proofs). The reason for this is that obviously, the formula transformation obtained from the $\langle . \rangle$ operator leads to a satisfiability-equivalent formula.

If PROBLEM is a counting problem, then it is obvious that the closure operators $\langle . \rangle_{\not\equiv}$ and $\langle . \rangle_{\not\equiv,\neq}$ lead to a parsimonious reduction (i.e., a polynomial-time computable many-one transformation preserving the number of solutions of the involved formula), and for the enumeration problem, it is easy to see that $\langle . \rangle_{\not\equiv}$ and $\langle . \rangle_{\not\equiv,\neq}$ give a reduction transforming a Γ_1-formula φ_1 into a Γ_2-formula φ_2 such that there is a canonical and easily computable bijection between the solution sets of φ_1 and φ_2.

The operator $\langle . \rangle_{\not\equiv,\neq}$ obviously transforms formulas into equivalent ones. Therefore, this operator can be applied to almost every conceivable problem in the formula context. In particular, this holds for the equivalence problem as defined above.

We therefore conclude:

Corollary 2.2. – $\langle . \rangle$ *can be applied to* CSP,
– $\langle . \rangle_{\not\equiv}$ *can be applied to* ENUM,
– $\langle . \rangle_{\not\equiv,\neq}$ *can be applied to* EQUIV.

It is obvious that if $\langle . \rangle$ can be applied to a problem, then the weaker closure operators can be applied as well, and similarly, if $\langle . \rangle_{\not\equiv}$ can be applied, then this also is true for $\langle . \rangle_{\not\equiv,\neq}$

In many cases, it cannot easily be seen that one of the above closure operators can be applied, but this result follows from a full complexity classification of the problem. This is true for the following problems:

Theorem 2.3. – $\langle . \rangle$ *can be applied to* ENUM *over Boolean domains [CH97],*
– $\langle . \rangle$ *cannot be applied to* ENUM *over non-Boolean domains [SS06],*
– $\langle . \rangle$ *can be applied to the* EQUIV *over Boolean domains [BHRV02].*

For both the enumeration and the equivalence problem, it is not clear that the complexity of the problem only depends on $\langle \Gamma \rangle$, since the introduction of new and yet in the Boolean case, this was shown to be true. Therefore, the question when the closure operator $\langle . \rangle$ can be applied is a very interesting research question in the constraint context. We give one possible answer at the end of Section 4.

The study of the closure operators we defined above is closely related to algebraic properties of the involved relations. In particular, the notion of a polymorphism has proven to be very useful. We will now define a generalization, which we then use to state the refined Galois connection.

Definition 2.4. *Let D be a finite domain, and let $R \subseteq D^n$ be a relation. Let $A \subseteq D^m$, and let $f\colon A \to D$ be a function. We say that f is a* partial polymorphism *of R, $f \in \mathrm{pPol}(R)$, if the following holds: For any $\mathbf{u_1} = (u_1^1, \ldots, u_n^1), \ldots, \mathbf{u_m} = (u_1^m, \ldots, u_n^m) \in R$, if $(u_i^1, \ldots, u_i^m) \in A$ for all $i \in \{1, \ldots, n\}$, then the coordinate-wise application*

$$f(\mathbf{u_1}, \ldots, \mathbf{u_m}) := (f(u_1^1, \ldots, u_1^m), \ldots, f(u_n^1, \ldots, u_n^m))$$

is in R. If $A = D^m$, we say that f is a polymorphism *of R, $f \in \mathrm{Pol}(R)$.*

We say that f in the definition above is a partial functoin on D if $A \neq D$, and f is total if $A = D$. The condition demanding that the vectors $(u_i^1, \ldots, u_i^m)$ must be elements of A ensures that the coordinate-wise application of the (possibly partial) function f gives a well-defined vector. It is obvious that for any relation R, it holds that $\mathrm{Pol}(R) = \mathrm{pPol}(R) \cap T_D$. For a set of functions B, the set $\mathrm{Inv}(B)$ denotes the set of relations R such that $B \subseteq \mathrm{pPol}(R)$. It is worth noting that although the operators $\mathrm{pPol}(.)$ and $\mathrm{Pol}(.)$ obviously differ, there is no need for two "versions" of the $\mathrm{Inv}(.)$-operator.

For a Boolean constraint language Γ, we say that Γ is *Schaefer* if Γ has a polymorphism which depends on at least two variables. Using this terminology, we can state a version of Schaefer's theorem. Note that the problem $\mathsf{CSP}(\Gamma \cup \{\{(0)\}, \{(1)\}\})$ is the problem to determine whether a given Γ-formula which also may contain clauses like $x = 0$ or $y = 1$ is satisfiable. This problem also is known as the "satisfiability problem with constants."

Theorem 2.5 ([Sch78]). *Let Γ be a Boolean constraint language. If Γ is Schaefer, then $\mathsf{CSP}(\Gamma \cup \{\{(0)\}, \{(1)\}\})$ can be solved in polynomial time. Otherwise, this problem is* NP*-complete.*

This version of Schaefer's theorem covers almost all of the tractable cases of $\mathsf{CSP}(\Gamma)$, the only additional cases are the trivial ones: if the constant 0 or 1-function is a polymorphism of Γ, then every Γ-formula is trivially satisfiable by the all-0-vector or the all-1-vector.

Polymorphisms have interesting algebraic properties: we say that a set B of (partial) functions on D is a *(partial) clone*, if B contains all projections and is closed under arbitrary composition. B is a *strong partial clone*, if in addition for every (partial) function $f \in B$, every further restriction of f again is a member of B. For a set B of (partial) functions, we denote with $[B]$ ($[B]_{\mathrm{p}}$) the clone generated by B (partial strong clone generated by B), i.e., the smallest (strong partial) clone which is a superset of B.

It is easy to see that the set $\mathrm{Pol}(\Gamma)$ is always a clone for a constraint language Γ, and the set $\mathrm{pPol}(\Gamma)$ always is a strong partial clone. For the Boolean case, Emil Post obtained a complete classification of all clones in [Pos41]. This classification is a powerful tool for the complexity classifications of Boolean constraint-related problems: whenever it can be shown that $\langle . \rangle$ can be applied to a problem, then his list of clones gives, via the Galois connection, a complete list of cases to look at.

We now consider a technical property of relations which is related to the introduction of equality clauses in the closure operators defined above.

Definition 2.6. *Let R be an n-ary relation over the domain D. We say that R is* irredundant*, if there are no natural numbers $i \neq j \in \{1, \ldots, n\}$ such that for all $\mathbf{v} \in R$, it holds that $\mathbf{v}[i] = \mathbf{v}[j]$.*

It is often useful to consider relations as matrices, where the rows are the tuples of the relation. It is obvious that such a representation is only unique up to the order of the rows. For example, the relation 1-in-3 containing all 3-tuples of Boolean values where exactly one 1 occurs, can be represented by the matrix

$$\text{1-in-3} = \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 1\,0\,0 \end{pmatrix}.$$

It is easy to see from the definition that a relation is irredundant if and only if its matrix does not contain two identical columns. Note that this condition does not depend on the order of the rows chosen in the matrix representation of the relation. Irredundant relations play a role in connection with the $=$-operator which is allowed in some of our closure operators: It is obvious that any non-empty relation which is defined with a formula containing an equality clause is redundant. In particular, the equality relation itself is obviously redundant.

For a relation R, let the *irredundant core of R* be defined as the relation represented by the matrix which is obtained by taking the matrix representing R, and for each column which appears more than once, keeping only the left-most copy of it. Note that this construction again is independent of the order of the rows chosen in the matrix. It is obvious that the irredundant core is indeed an irredundant relation. The following proposition is obvious:

Proposition 2.7. *Let R be a relation, and let R' be its irredundant core. Then*

$$\langle R \rangle_{\not\equiv} = \langle R' \rangle_{\not\equiv}.$$

Proof. Assume that only one column appears twice in R's matrix, the general case then follows by induction. Without loss of generality, assume that the last two columns in R's matrix are identical.

Let n be the arity of R. It is obvious that for a set of variables $x_1, \ldots, x_n$, the following equivalences hold:

$$\begin{aligned} R'(x_1, \ldots, x_{n-1}) &\iff R(x_1, \ldots, x_{n-1}, x_{n-1}) \\ R(x_1, \ldots, x_n) &\iff R'(x_1, \ldots, x_{n-1}) \wedge (x_{n-1} = x_n). \end{aligned}$$

It therefore follows that $R' \in \langle R \rangle_{\not\equiv}$, and $R \in \langle R' \rangle_{\not\equiv}$. Since $\langle . \rangle_{\not\equiv}$ is a closure operator, this implies the proposition. □

The consequence of the above Proposition 2.7 is that as long as we allow equality clauses in our closure operators, i.e., for the operators $\langle . \rangle$ and $\langle . \rangle_{\not\equiv}$, we can assume that we are working with irredundant relations only.

3 The Galois Connection

The following theorem is the standard Galois connection which has been used to prove complexity results in the constraint satisfaction context:

Theorem 3.1 ([Gei68,JCG97]). *Let Γ be a finite constraint language over a domain D, and let R be a relation over D. Then $R \in \langle \Gamma \rangle$ if and only if* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(R)$.

In particular, this theorem can be used to show the following complexity result. It was first stated for polynomial-time many one reductions in [JCG97], and refined to a logspace reduction in [ABI+05].

Theorem 3.2 ([JCG97,ABI+05]). *Let Γ_1 and Γ_2 be constraint languages over a finite domain D, such that* $\mathrm{Pol}(\Gamma_2) \subseteq \mathrm{Pol}(\Gamma_1)$. *Then* $\mathsf{CSP}(\Gamma_1) \leq_m^{\log} \mathsf{CSP}(\Gamma_2)$.

This theorem is useful because as mentioned before, the sets $\mathrm{Pol}(\Gamma)$ are always clones, and their structure is well-studied. The following theorem is the refinement of the Galois connection exhibited in Theorem 3.1 to the $\langle . \rangle_{\not\equiv}$ operator. This Theorem was proven by Romov in 1981 [Rom81], but is already implicit in [Gei68].

Theorem 3.3 ([Rom81]). *Let Γ be a finite constraint language over a domain D, and let R be a relation over D. Then $R \in \langle \Gamma \rangle_{\not\equiv}$ if and only if* $\mathrm{pPol}(\Gamma) \subseteq \mathrm{pPol}(R)$.

The above Theorem 3.3 does not consider the closure operator $\langle . \rangle_{\not\ni,\neq}$. There also is a further refinement of the Galois connection to this case, where instead of functions, we consider hyperfunctions. However, we will show that for many practical applications, this is not necessary.

It should be noted that the seemingly subtle step from polymorphisms to partial polymorphisms has unfortunate consequences. While, as mentioned, the lattice of clones for the Boolean case is well understood and completely classified due to Post's work [Pos41], the picture for partial clones is much less clear. We know that already over the Boolean domain, there is an uncountable number of partial clones, and the structure of the lattice of partial clones is not completely known. This is the key reason why results that the operator $\langle . \rangle$ can be applied are so useful in practice: for a given Boolean constraint language Γ, it is easy to determine its set of polymorphisms. In fact, this can be done automatically by an algorithm. Hence, if there is a classification theorem giving the complexity of $\mathsf{PROBLEM}(\Gamma)$ as an easy function of $\langle \Gamma \rangle$, then the complexity of $\mathsf{PROBLEM}(\Gamma)$ can be determined automatically.

4 Maximal Partial Clones and Finite Bases

The Galois connection stated in Theorem 3.3 shows that in order to classify the complexity of problems where the standard Galois connection cannot be proven to hold "a priori," we need to study the sets of partial polymorphisms of a given constraint language. It is obvious that a given clone contains many partial clones. In fact, since there are uncountably many partial clones and only countably many clones, there are clones with uncountably many partial clones "inside them." We therefore need a more detailed examination of the partial clones occurring inside a given clone. As mentioned, the sets $\mathrm{pPol}(\Gamma)$ always form strong partial clones, and therefore we can restrict ourselves to strong partial clones.

Definition 4.1. *Let B be a clone on the domain D. We define*

$$\mathcal{C}_{=B} := \{C \mid C \text{ is a strong partial clone on } D \text{ and } C \cap T_D = B\}.$$

It should be noted that the condition $C \cap T_D = B$ in the above definition cannot be replaced by $C \cap T_D \subseteq B$. As we will see soon, there is a close correspondence between the partial clones defined above and partial functions with a special property, which we define now.

Definition 4.2. *Let B be a clone on the domain D, and let f be a partial n-ary function on D. We say that f is* B-total*, if for all functions $f_1, \ldots, f_n$ from B, the function $f(f_1, \ldots, f_n)$ is either non-total, or a member of B.*

The first thing we need to note about these functions is that they form a strong partial clone:

Theorem 4.3. *Let B be a clone on the domain D. Then the set of B-total functions on D forms a strong partial clone.*

Proof. It is obvious that the projections are B-total, since they are contained in the clone B. It remains to show that compositions of B-total functions are B-total. Therefore, let $f, f_1, \ldots, f_n$ be B-total, and define $g := f(f_1, \ldots, f_n)$. If one of the functions $f_1, \ldots, f_n$ is not total, then g is not total either, and hence is B-total. Therefore, assume that $f_1, \ldots, f_n$ are total. In this case, since they are B-total, it follows that $f_1, \ldots, f_n$ are functions from B. Since f is B-total, it follows that $g = f(f_1, \ldots, f_n)$ is either non-total, or is a function from B. In both cases, g is B-total, as claimed. □

We now show that the B-total functions are exactly those which are contained in the clones from $\mathcal{C}_{=B}$. The most important consequence of this theorem is that, together with Theorem 4.3, it implies that the set $\bigcup \mathcal{C}_{=B} = \bigcup_{C \in \mathcal{C}_{=B}} C$ forms a strong partial clone.

Theorem 4.4. *Let B be a clone on the domain D. Then $\bigcup \mathcal{C}_{=B}$ is exactly the set of partial functions which are B-total.*

Proof. First, let $f \in \bigcup \mathcal{C}_{=B}$. Then there is a strong partial clone C, such that $C \cap T_D = B$, and $f \in C$. In particular, it follows that $B \subseteq C$. Hence, the composition of f and functions from B still gives a function from C, in particular a function which is either non-total, or a member of B. Hence, f is B-total.

For the other direction, assume that f is B-total, and assume that $f \notin \bigcup \mathcal{C}_{=B}$. In particular, this implies that $[\{f\} \cup B]_{\mathrm{p}} \cap T_D \neq B$. Since $B \subseteq [\{f\} \cup B]_{\mathrm{p}}$, this implies that $[\{f\} \cup B]_{\mathrm{p}} \cap T_D \supsetneq B$. Therefore, there exists a total function $g \in [\{f\} \cup B]_{\mathrm{p}}$, such that $g \notin B$. Then g can be written as $f(f_1, \ldots, f_n)$, where the f_i are functions from $[\{f\} \cup B]_{\mathrm{p}}$, and such that this representation is the minimal one for a function with these properties. (due to minimality, we can disregard the case where the outmost function is from B.)

Since g is total, $f_1, \ldots, f_n$ must be total. Therefore, due to the minimality of g, it follows that $f_1, \ldots, f_n$ are functions from B. Since f is B-total, this implies that g is either non-total, or a function from B. This is a contradiction. □

Due to the above, the set $\bigcup \mathcal{C}_{=B}$ forms a strong partial clone. It is the largest strong partial clone which is contained in the clone B. Therefore, a constraint languages Γ satisfying $\mathrm{pPol}(\Gamma) = \bigcup \mathcal{C}_{=B}$ is in a certain way among those which lead the "easiest" problems among those languages with polymorphism set B. The following Proposition is obvious:

Proposition 4.5. *Let Γ_1 and Γ_2 be constraint languages over a finite domain D, such that $\mathrm{Pol}(\Gamma_1) = B$ for some clone B over D, and $\mathrm{pPol}(\Gamma_2) = \bigcup \mathcal{C}_{=B}$. Then $\Gamma_2 \subseteq \langle \Gamma_1 \rangle_{\sharp}$.*

Proof. By the definition of $\mathcal{C}_{=B}$, it is obvious that $\mathrm{pPol}(\Gamma_1) \cap T_D = B$. In particular, this implies that $\mathrm{pPol}(\Gamma_1) \in \mathcal{C}_{=B}$, and therefore, we know that $\mathrm{pPol}(\Gamma_1) \subseteq \bigcup \mathcal{C}_{=B} = \mathrm{pPol}(\Gamma_2)$. Theorem 3.3 now implies the result. □

Due to this observation, we know that for all computational problems from the constraint context where the $\langle . \rangle_{\not\equiv}$-operator can be applied, the problem for the constraint language Γ_2 reduces to the problem for the constraint language Γ_1 (for languages with the prerequisites of Proposition 4.5). Therefore, the language Γ_2 can be regarded as being the "easiest" in this clone. These constraint languages are of interest, because obviously, any hardness result shown for Γ_2 transfers to any language satisfying the properties of Γ_1 (*not* necessarily to any constraint language whose polymorphisms are a subset of B).

We now show that for all "interesting" clones B, a constraint language Γ_2 satisfying the above conditions indeed exists. A clone B is "interesting" in this way if there is some finite constraint language Γ which has exactly the set B as polymorphisms. Note that this is the case if and only if there is a single relation which has this set of polymorphisms. In [BRSV05], for each except 8 of the Boolean co-clones, a finite constraint language Γ generating it is given (and for the remaining 8, it was shown that such a language cannot exist). Therefore, the results presented in [BRSV05] can be applied to construct relations with the properties required in the following definition.

Definition 4.6. *Let* $R = \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$ *be an* r*-ary relation and let* $\{\mathbf{v}_1, \ldots, \mathbf{v}_{2^n}\} = \{0,1\}^n$ *(assume lexicographical ordering for both sets of tuples). The* semi-extension of R, $R^{semi\text{-}ext} = \{\mathbf{t'}_1, \ldots, \mathbf{t'}_n\}$ *is defined by* $\mathbf{t'}_i = (\mathbf{t}_i[1], \ldots, \mathbf{t}_i[r], \mathbf{v}_1[i], \ldots, \mathbf{v}_{2^n}[i])$. *The* extension of R, $R^{[ext]}$, *is the relation* $\{f(\mathbf{w}_1, \ldots, \mathbf{w}_k) \mid f \in \mathrm{Pol}(R)$ *of arity* k *and* $\mathbf{w}_1, \ldots, \mathbf{w}_k \in R^{semi\text{-}ext}\}$. *For a constraint language* Γ, *let the extension of* Γ *be defined as* $\Gamma^{[ext]} := (\times_{R \in Gamma} R)^{[ext]}$.

Note that in the definition above, the lexicographical ordering is not important, an arbitrary ordering would suffice to ensure that $R^{[\mathrm{ext}]}$ is well-defined. As an example, recall the relation 1-in-3 from Section 2. It can be shown that the polymorphisms of 1-in-3 contain only projections. Hence, it is easy to see that the extension of 1-in-3 has the following form:

$$\text{1-in-3}^{\text{semi-ext}} = \text{1-in-3}^{[\text{ext}]} = \begin{pmatrix} 0&0&1&0&0&0&0&1&1&1&1 \\ 0&1&0&0&0&1&1&0&0&1&1 \\ 1&0&0&0&1&0&1&0&1&0&1 \end{pmatrix}.$$

It is obvious that $\mathrm{Pol}(R) \subseteq \mathrm{Pol}\left(R^{[\mathrm{ext}]}\right)$. On the other hand, we can express R using $R^{[\mathrm{ext}]}$, since $R(x_1, \ldots, x_k)$ is equivalent to $\exists y_1, \ldots, y_{2^n}\, R^{[\mathrm{ext}]}(x_1, \ldots, x_k, y_1, \ldots, y_{2^n})$, and hence Theorem 3.1 implies that $\mathrm{Pol}\left(R^{[\mathrm{ext}]}\right) \subseteq \mathrm{Pol}(R)$. Hence, $\mathrm{Pol}\left(R^{[\mathrm{ext}]}\right) = \mathrm{Pol}(R)$. We now show that the partial polymorphisms of $R^{[\mathrm{ext}]}$ contain every partial function they possibly could.

Theorem 4.7. *Let* R *be a relation and* $B = \mathrm{Pol}(R)$. *Then* $\mathrm{pPol}\left(R^{[ext]}\right) = \bigcup \mathcal{C}_{=B}$.

Proof. Let $f \in \bigcup \mathcal{C}_{=B}$ a k-ary partial function and $R^{\text{semi-ext}} = \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$ of arity r. We show $f \in \mathrm{pPol}\left(R^{[\mathrm{ext}]}\right)$ first. Assume $f \notin \mathrm{pPol}\left(R^{[\mathrm{ext}]}\right)$, then there

exist $\mathbf{v}_1, \ldots, \mathbf{v}_k \in R^{[\text{ext}]}$ such that $\mathbf{v} = f(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ is well defined but not an element from $R^{[\text{ext}]}$. For every $i \in \{1, \ldots, k\}$ there exist $h_i \in \text{Pol}(R)$ of arity l_i and $j_1^i, \ldots, j_{l_i}^i \in \{1, \ldots, n\}$ such that

$$\mathbf{v}_i = h_i(\mathbf{t}_{j_1^i}, \ldots, \mathbf{t}_{j_{l_i}^i}).$$

We define an n-ary partial function g:

$$g(x_1, \ldots, x_n) = f(h_1(x_{j_1^1}, \ldots, x_{j_{l_1}^1}), \ldots, h_k(x_{j_1^k}, \ldots, x_{j_{l_k}^1})).$$

It holds that $g(\mathbf{t}_1, \ldots, \mathbf{t}_n) = f(\mathbf{v}_1, \ldots, \mathbf{v}_k)$, therefore $g(\mathbf{t}_1[i], \ldots, \mathbf{t}_n[i])$ is defined for every $1 \leq i \leq \text{ar}(R^{\text{semi-ext}}) = r + 2^n$. Since due to the construction of $R^{\text{semi-ext}}$ for every $\mathbf{s} \in \{0,1\}^n$ there is an $i \in \{r+1, \ldots, r+2^n\}$ such that $\mathbf{s} = (\mathbf{t}_1[i], \ldots, \mathbf{t}_n[i])$, it follows that g is a total function. By construction g is in $\bigcup \mathcal{C}_{=B}$, hence $g \in B = \text{Pol}(R)$. Then, since $\text{Pol}(R) = R^{[\text{ext}]}$, $g(\mathbf{t}_1, \ldots, \mathbf{t}_n) \in R^{[\text{ext}]}$ which is a contradiction to $f(\mathbf{v}_1, \ldots, \mathbf{v}_k) \notin R^{[\text{ext}]}$.

The other inclusion holds because $\text{pPol}\left(R^{[\text{ext}]}\right) \cap T_D = \text{Pol}\left(R^{[\text{ext}]}\right) = \text{Pol}(R)$ and therefore $\text{pPol}\left(R^{[\text{ext}]}\right) \in \mathcal{C}_{=\text{Pol}(R)}$. □

Proposition 4.5 immediately gives the following corollary:

Corollary 4.8. *Let Γ be a constraint language. Then $\Gamma^{[ext]} \in \langle\Gamma\rangle_{\nexists}$.*

When considering the closure operator $\langle . \rangle_{\nexists,\neq}$, we are interested in irredundant relations, as previously explained. The following corollary immediately follows from the above, the observation that the set of partial polymorphisms of a relation and its irredundant core are the same, and that any formulas which represents a non-empty irredundant relation cannot contain any non-tautological equality clauses:

Corollary 4.9. *Let Γ be a constraint language, and let R be the irredundant core of $\Gamma^{[ext]}$. Then $R \in \langle\Gamma\rangle_{\nexists,\neq}$.*

The irredundant core of $R^{[\text{ext}]}$ needed in the above corollary can be obtained from R as follows: from R, construct the semi-extension $R^{\text{semi-ext}}$, and then delete redundant columns in that matrix, and then apply all the polsmorphisms of R. It is obvious that in the last step, no redundancies are added.

It should be noted that since the irredundant core of $R^{\text{semi-ext}}$ contains every possible "column" in that matrix exactly once, this relation is determined (up to order of the columns) by the number of elements in R. Therefore, up to order of the columns and redundancies, the relation $R^{[\text{ext}]}$ is uniquely determined by the number of elements in R, and the set of polymorphisms of R.

We now return to the question commented on on the introduction: when does the Galois connection hold "a posteriori?" In the above, we exhibited, for a clone B, the "easiest" constraint language with this set of polymorphisms. In [CKZ05], Creignou, Kolaitis, and Zanuttini constructed "hardest" constraint languages for the Boolean case, i.e., for each Boolean clone B, they give a set

of relations Γ_B such that every constraint language Γ' with $\mathrm{Pol}(\Gamma') = B$ is contained in $\langle \Gamma_B \rangle_{\not\exists,\neq}$. These sets are in most cases not finite, but they are highly uniform, for example consisting of all relations OR^m for every possible arity m.

We will now show how the bases from [CKZ05] and our $\Gamma^{[\mathrm{ext}]}$-construction can be used in a complexity context:

Lemma 4.10. *Let* PROBLEM *be a computational problem such that* $\langle . \rangle_{\not\exists,\neq}$ *can be applied to* PROBLEM. *Then for every Boolean constraint language* Γ, *there exists a finite subset* Γ' *of* $\Gamma_{\mathrm{Pol}(\Gamma)}$ *such that* PROBLEM(Γ) *reduces to* PROBLEM(Γ').

Proof. By the properties of $\Gamma_{\mathrm{Pol}(\Gamma)}$, it follows that every relation from Γ can be expressed by a $\Gamma_{\mathrm{Pol}(\Gamma)}$-formula. Since Γ is finite, the set Γ' of all relations from $\Gamma_{\mathrm{Pol}(\Gamma)}$ required in these expressing formulas is finite. It is obvious that every Γ-formula can be efficiently transformed into an equivalent Γ'-formula, and since $\langle . \rangle_{\not\exists,\neq}$ can be applied to PROBLEM, this gives the reduction. □

Hence, the finite subsets of $\Gamma_{\mathrm{Pol}(\Gamma)}$ give an upper complexity bound. We now show that our relations $R^{[\mathrm{ext}]}$ give a similar lower bound.

Corollary 4.11. *Let* PROBLEM *be a computational problem such that* $\langle . \rangle_{\not\exists}$ *can be applied to* PROBLEM. *Then for any constraint language* Γ, PROBLEM$\left(\Gamma^{[ext]}\right)$ *reduces to* PROBLEM(Γ).

Proof. This follows directly from Corollary 4.8. □

We have a similar result for the closure operator $\langle . \rangle_{\not\exists,\neq}$:

Corollary 4.12. *Let* PROBLEM *be a computational problem such that* $\langle . \rangle_{\not\exists,\neq}$ *can be applied to* PROBLEM, *let* Γ *be a constraint language, and let* R *be the irredundant core of* $\Gamma^{[ext]}$. *Then* PROBLEM(R) *reduces to* PROBLEM(Γ).

Proof. This follows directly from Corollary 4.9. □

The above lemmas immediately give the following corollary:

Corollary 4.13. *Let* PROBLEM *be a computational problem for formulas over the Boolean domain, such that* $\langle . \rangle_{\not\exists}$ *can be applied to* PROBLEM. *Then* $\langle . \rangle$ *can be applied to* PROBLEM *if and only if for any Boolean constraint language* Γ, *and any finite subset* Γ' *of* $\Gamma_{\mathrm{Pol}(\Gamma)}$, PROBLEM$(\Gamma')$ *reduces to* PROBLEM$\left(\Gamma^{[ext]}\right)$.

The analogous result of course holds for the operator $\langle . \rangle_{\not\exists,\neq}$:

Corollary 4.14. *Let* PROBLEM *be a computational problem for formulas over the Boolean domain, such that* $\langle . \rangle_{\not\exists,\neq}$ *can be applied to* PROBLEM. *Then* $\langle . \rangle$ *can be applied to* PROBLEM *if and only if for any Boolean constraint language* Γ, *and any finite subset* Γ' *of* $\Gamma_{\mathrm{Pol}(\Gamma)}$, PROBLEM$(\Gamma')$ *reduces to* PROBLEM(R), *where* R *is the irredundant core of* $\Gamma^{[ext]}$.

The finite subsets of $\Gamma_{\mathrm{Pol}(\Gamma)}$ appearing in the above constructions are of a very regular form: since the sets $\Gamma_{\mathrm{Pol}(\Gamma)}$ are uniform sets of relations containing, for example, the relation OR of arbitrary arities, for the arbitrary finite subsets it suffices to consider subsets containing, for example, all OR-relations up to some finite arity.

5 Applications

For problems where the complexity of the problem does not only depend on the set of total polymorphisms, like enumeration for the non-Boolean case [SS06], it is obvious that for a classification it is necessary to look into the partial clones, which constitute a refinement of the clone lattice. However, as we mentioned in Theorem 2.3, there are many cases where the complexity does only depend on the polymorphisms of the involved constraint languages, but where the Galois connection does not hold a priori.

For the already-mentioned problems ENUM and EQUIV, we show how our methods can be used to obtain a full complexity classification. We show that our techniques can be used to obtain much simpler proofs for some of these problems. We will only cover the hardness proofs of the considered problems in detail, and just give a rough idea of how the polynomial time algorithms work, since the purpose of this paper is to explain how our techniques can be applied to these problems.

Since we consider these problems over the Boolean domain only, we introduce some important clones arising here. The clone N contains all Boolean functions which are either constant or depend only on one variable, i.e., basically constants, projections, and negation. The clone N_2 contains all non-constant functions from N. The clone I contains both constants and the projections, I_1 (I_0) contains the projection and the constant 1 (the constant 0), and finally I_2 contains only the projections. It is easy to see that a constraint language Γ is not Schaefer if and only if its polymorphism set is one of the above-defined clones, and all of these clones are subsets of N. The Schaefer property is essential in the case of Boolean constraint languages, because for many problems, the tractable instances are exactly those which are Schaefer.

5.1 Non-constant solutions

The problem to decide whether a propositional formula has a solution which is not constant is denoted with CSP^*. In [CH97], the following result was shown using a number of technical implementation results. We now show how this theorem can be proven in a much simpler way, using the tools introduced in Section 4.

Theorem 5.1. *Let Γ be a finite constraint language such that Γ is not Schaefer. Then $\mathsf{CSP}^*(\Gamma)$ is* NP*-complete.*

Proof. If Γ is not Schaefer, then $\mathrm{Pol}(\Gamma) \subseteq \mathrm{N}$. It is obvious that the problem is in NP. For $\Gamma = \{R_1, \ldots, R_n\}$ let $R = R_1 \times \cdots \times R_n$. Since $\mathsf{CSP}^*(R) \leq_m^p \mathsf{CSP}^*(\Gamma)$, we can assume $\Gamma = \{R\}$ without loss of generality. We show $\mathsf{CSP}(R^+) \leq_m^p \mathsf{CSP}^*\left(R^{[\mathrm{ext}]}\right) \leq_m^p \mathsf{CSP}^*(R)$. For the first reduction let

$$\varphi = \bigwedge_{i=1}^{k} R(x_1^i, \ldots, x_r^i) \bigwedge_{i=1}^{l} y^i \bigwedge_{i=1}^{m} z^i$$

for not necessarily distinct variables x^i_j, y^i, z^i. We construct an $R^{[\text{ext}]}$-formula ψ. If $\{y^1,\dots,y^l\}\cap\{z^1,\dots,z^m\}\neq\emptyset$ we set $\psi = R^{[\text{ext}]}(x,\dots,x)$, otherwise we define

$$\psi = \bigwedge_{i=1}^{k} R^{[\text{ext}]}(x'^i_1,\dots,x'^i_r,f,w^i_1,\dots,w^i_{2^{|R|}-2},t),$$

with $x'^i_j = \begin{cases} t & \text{if } x^i_j \in \{y^1,\dots,y^l\},\\ f & \text{if } x^i_j \in \{z^1,\dots,z^m\},\\ x^i_j & \text{otherwise.}\end{cases}$ for new distinctive variables t, f, and w^i_j for $1\le i\le k$ and $1\le j\le 2^{|R|}-2$.

Due to the definition of $R^{[\text{ext}]}$ for every $(v_1,\dots,v_r)\in R$ there are $t_1,\dots,t_{2^{|R|}-2}\in\{0,1\}$ such that $(v_1,\dots,v_r,0,t_1,\dots,t_{2^{|R|}-2},1)$ is an element of $R^{[\text{ext}]}$. Therefore every truth assignment for φ can be extended to a non-constant solution for ψ.

Now let *psi* have a non constant solution I. Then there must be a clause in ψ such that I restricted to the variables appearing in the clause is non constant (otherwise I would be constant because t appears in every clause).

Case: $\mathrm{Pol}\,(R)\in\{\mathrm{I},\mathrm{I}_0,\mathrm{I}_1,\mathrm{I}_2\}$**.** If $R^{[\text{ext}]}$ has elements where the last $2^{|R|}$ values are all 0 or all 1, then that are the tuples $(0,\dots,0)$ and $(1,\dots,1)$. In all other tuples the $r+1$st value, which in all clauses of ψ is addressed by f, is 0 and the last value, addressed by t, is 1. Since I is non constant for some clause in ψ, that means that $I(f)=0$ and $I(t)=1$.

Case: $\mathrm{Pol}\,(R)\in\{\mathrm{N},\mathrm{N}_2\}$**.** Again the last $2^{|R|}$ values of an tuple of $R^{[\text{ext}]}$ can be all equal only if all values of the tuple are equal. In all other elements the $r+1$st and the last value are not equal. Therefore it holds that $I(f)\neq I(t)$. Since $\mathrm{Pol}\left(R^{[\text{ext}]}\right)=\mathrm{Pol}\,(R)\in\{\mathrm{N},\mathrm{N}_2\}$, it holds that $R^{[\text{ext}]}$ is complementive, so we can assume $I(f)=0$ and $I(t)=1$ without loss of generality (otherwise consider the solution I' defined by $I'(x)=1$ iff $I(x)=0$).

So, the assignment J defined by $J(x)=\begin{cases} I(t) & \text{if } x\in\{y^1,\dots,y^l\}\\ I(f) & \text{if } x\in\{z^1,\dots,z^m\}\\ I(x) & \text{otherwise}\end{cases}$ is a solution for φ. Hence, $\varphi\in\mathsf{CSP}\,(R^+)$ if and only if $\psi\in\mathsf{CSP}^*\left(R^{[\text{ext}]}\right)$.

The second reduction $\mathsf{CSP}^*\left(R^{[\text{ext}]}\right)\le^p_m\mathsf{CSP}^*\,(R)$ is trivial due to the fact that $R^{[\text{ext}]}\in\langle R\rangle_{\not\equiv}$. □

5.2 Enumeration

In our context, an *efficient enumeration algorithm* enumerates, given a formula φ, the set of solutions of φ such that the time between the printing of the first solution, the time between the printing of each two consecutive soloutions, and between the printing of the final solution and the termination of the algorithm is bunded by a polynomial in the input. The following classification theorem was originally shown by Creignou and Hébrard in [CH97].

Theorem 5.2. *Let Γ be a Boolean constraint language. If Γ is Schaefer, then there exists an efficient algorithm computing for each Γ-formula its set of solutions. If Γ is not Schaefer, then such an algorithm does not exist, unless* P = NP.

Proof. First assume that Γ is Schaefer. Due to Theorem 2.5, it follows that for any Γ-formula φ with variables $x_1, \ldots, x_n$, the questions if $\varphi[x_1/0]$ or $\varphi[x_1/1]$ are satisfiable can be solved in polynomial time. This easily gives an efficient algorithm to enumerate all satisfying assignments for φ, by recursively enumerating the solutions for these two formulas, if they are satisfiable.

Now assume that Γ is not Schaefer. Then, due to Theorem 5.1, we know that the problem to determine for a given Γ-formula if it has a non-constant solution is NP-complete. But it is obvious that any efficient enumeration algorithm can be used to answer this question. Hence, such an algorithm cannot exist, unless P = NP. □

5.3 Equivalence

The equivalence problem, $\mathsf{EQUIV}(\Gamma)$, is the problem to decide whether two Γ-formulas have the same set of solutions. This problem is obviously in coNP, and can be shown to be truth-table reducible to the constraint satisfaction problem for Γ-formulas with constants ([BHRV02]). Hence, the equivalence problem is solvable in polynomial time if Γ is Schaefer due to Theorem 2.5. For the dichotomy, it remains to prove that the problem is coNP-hard in all other cases. The result was originally proven by Böhler, Hemaspaandra, Reith, and Vollmer in [BHRV02]. Again, we show how the proof can be simplified using our techniques (note that the proof from [BHRV02] also relies on the complexity classification of CSP^* from [CH97]):

Theorem 5.3. *Let Γ be a constraint language such that Γ is not Schaefer. Then* $\mathsf{EQUIV}(\Gamma)$ *is* coNP*-complete.*

Proof. The upper bound is obvious. Let R be the cartesian product over all relations in Γ, then $\mathrm{Pol}(R) = \mathrm{Pol}(\Gamma)$, and since $\mathsf{EQUIV}(R) \leq_m^p \mathsf{EQUIV}(\Gamma)$, it suffices to show the hardness result for R. By Theorem 5.1, it suffices to show $\mathsf{CSP}^*(R) \leq_m^p \overline{\mathsf{EQUIV}(R)}$.

Let R' be the irredundant core of $R^{\text{semi-ext}}$, and let R'' denote the relation obtained from R' by applying all polymorphisms from R to it. Then R'' is the irredundant core of $R^{[\text{ext}]}$. Due to Corollary 4.9, $R'' \in \langle R \rangle_{\not\equiv,\neq}$. We know that in R', there appear two columns which are the constant-0 and the constant-1 column. Without loss of generality, assume that these appear as the last two columns in R'. Note that since Γ is not Schaefer, R is not Schaefer either and therefore not closed under conjunction. Hence, R contains at least two elements, and therefore R' is at least 4-ary.

Let φ be an R-formula, and let x be an arbitrary variable from φ. For each variable y appearing in φ, introduce a clause $R''(y, \ldots, y, x, x)$, and let ψ denote the conjunction of all these clauses. Due to Corollary 4.9, ψ can be written as

an R-formula. Obviously, the set of variables of ψ and the set of variables of φ is the same.

We show that φ has a non-constant solution if and only if it is not equivalent to ψ. Note that φ and ψ have the same constant solutions: the constant c-assignment is a solution for R if and only if $(c, \ldots, c) \in R$ if and only if c is a polymorphism of R if and only if c is a polymorphism of $R^{[\text{ext}]}$ (since $\text{Pol}\,(R) = \text{Pol}\left(R^{[\text{ext}]}\right)$) if and only if the constant c-assignment is a solution for ψ.

We prove that all solutions of ψ are constant. By the construction of $R^{[\text{ext}]}$, the only tuples in $R^{[\text{ext}]}$ where the last two components are identical are constant tuples: In the relation $R^{\text{semi-ext}}$, the last two components of each tuple are different. Since $\text{Pol}\,(R)$ only contains injective functions and constants, such tuples are only generated by the application of constant polymorphisms. Therefore, every variable is assigned the same value as x in every solution for ψ, i.e., ψ only has constant solutions.

Now assume that φ only has consant solutions. Since the sets of constant solutions of φ and of ψ are identical and ψ only has constant solutions, φ and ψ are equivalent. Otherwise, φ and ψ cannot be equivalent, since φ has a non-constant solution, and ψ does not. Therefore, the reduction is complete. □

6 Conclusion and Future Research

We have shown that the refinement of the usual Galois connection to partial polymorphisms can be applied to give easier proofs for complexity classifications over the Boolean domain. By constructing the relations $\Gamma^{[\text{ext}]}$, we have exhibited natural relations, such that it is sufficient to prove hardness results for these.

The next interesting question in this line of research is the application of our tools to problems where the $\langle . \rangle$ operator cannot be applied. We believe that using these techniques, further results on the enumeration problem for non-Boolean domains can be achieved.

It will also be interesting to study the structure of the partial clones appearing "inside" a clone. It is worth noting that our results allow us to identify those clones which do not split up into more than one strong partial clone: this is the case if and only if the bases exhibited by Creignou, Kolaitis, and Zanuttini in [CKZ05] and our relation $\Gamma^{[\text{ext}]}$ have the same $\langle . \rangle_{\not\equiv}$-closure (unpublished work by Nordh and Zanuttini suggests that there are many co-clones for which this is the case).

References

ABI+05. E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, pages 71–82, 2005.

BHRV02. E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Computer Science Logic*,

volume 2471 of *Lecture Notes in Computer Science*, pages 412–426, Berlin Heidelberg, 2002. Springer Verlag.

BRSV05. E. Böhler, S. Reith, H. Schnoor, and H. Vollmer. Bases for Boolean co-clones. *Information Processing Letters*, 96:59–66, 2005.

Bul06. Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

CH97. N. Creignou and J.-J. Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 31(6):499–511, 1997.

CKZ05. N. Creignou, P. Kolaitis, and B. Zanuttini. Preferred representations of Boolean relations. Technical Report TR05-119, Electronic Colloquium on Computational Complexity (ECCC), 2005.

Gei68. D. Geiger. Closed systems of functions and predicates. *Pac. J. Math*, 27(2):228–250, 1968.

JCG97. P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

Lad75. R. Ladner. On the structure of polynomial-time reducibility. *Journal of the ACM*, 22:155–171, 1975.

Pos41. E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

Rom81. B. A. Romov. The algebras of partial functions and their invariants. *Cybernetics and Systems Analysis*, 17(2):157–167, 1981.

Sch78. T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

SS06. H. Schnoor and I. Schnoor. Enumerating all solutions for constraint satisfaction problems, 2006. *These Proceedings*.