Enumerating all Solutions for Constraint Satisfaction Problems

Henning Schnoor, Ilka Schnoor

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

Abstract. We contribute to the study of efficient enumeration algorithms for all solutions of constraint satisfaction problems. The only algorithm known so far, presented by Creignou and Hébrard [CH97] and generalized by Cohen [Coh04], reduces the enumeration problem for a constraint language Γ to the decision problem for a slightly enlarged constraint language Γ^+, i.e., it yields an efficient enumeration algorithm for the case where $\mathsf{CSP}(\Gamma^+)$ is tractable. We develop a new class of algorithms, yielding efficient enumeration algorithms for a broad class of constraint languages. For the three-element domain, we achieve a first step towards a dichotomy theorem for the enumeration problem.

Keywords. computational complexity, constraints, enumeration

1 Introduction

Constraint satisfaction problems (CSPs) have attracted considerable attention in complexity theory. Especially the non-uniform version of the problem, $\mathsf{CSP}(\Gamma)$ has been studied. Here, we fix a constraint language Γ, which is a set of finitary relations over an arbitrary domain. The problem $\mathsf{CSP}(\Gamma)$ is the satisfiability problem for propositional formulas, where the form of the clauses appearing is restricted by Γ, so-called Γ-formulas. If the domain is Boolean, then these problems generalize many common restrictions of the satisfiability problem (2SAT, 3SAT, Horn-SAT, etc). In the non-Boolean case, $\mathsf{CSP}(\Gamma)$ can generalize problems as colorability in graphs, scheduling problems, database queries, and others. In fact, most combinatorial problems where the goal is to find some assignment to variables which needs to satisfy a collection of "local" conditions can be seen as a CSP. Due to this property, CSPs can be seen as the "combinatorial core of complexity theory" [CKS01], and are of interest for theoretical reasons as well.

The complexity of $\mathsf{CSP}(\Gamma)$ was first studied by Thomas Schaefer. In his seminal paper [Sch78], he showed that in the Boolean case, this problem is always solvable either in polynomial time, or is NP-complete, and he gave easy criteria for Γ which allow for a polynomial time decision procedure. The study of constraint satisfaction problems becomes much more challenging when considering non-Boolean domains. For example, Bulatov proved that an analog of Schaefer's dichotomy holds for the case where the domain is of cardinality three [Bul06], but the proof is much more involved. It is conjectured that dichotomy results hold for arbitrary finite domains. In fact, CSP is in a certain context the largest class of problems for which such results are possible [FV98].

Besides satisfiability of formulas, other computational goals have been studied, as for example optimal satisfiability [RV00], counting of the number of satisfying solutions for constraint formulas [CH96], and equivalence and isomorphism [BHRV02] for a fixed constraint language Γ. In many of these cases, dichotomy theorems have been proven for Boolean domains. A lot of research has been done for the case where the domain is a non-Boolean finite set, see e.g. [JCG97], [Dal05], [DK06], and [Bul06].

A problem which is very relevant in practice is the enumeration problem for constraint formulas, which we study in this paper. Here the task is to enumerate, for a given Γ-formula, the set of its solutions. Roughly speaking, an "efficient" algorithm requires only polynomial time for each solution it generates. Such an algorithm can only exist if the satisfiability problem for Γ-formulas can be solved in polynomial time. For the Boolean domain, the question in which cases efficient enumeration algorithms exist has been studied by Creignou and Hébrard in [CH97]. In [Coh04], it was shown that their algorithm can be applied to arbitrary finite domains. The algorithm reduces the enumeration problem to the decision problem as follows: for the constraint language Γ, let Γ^+ be the constraint language containing the relations from Γ and relations representing literals over the domain D, i.e., it lets us express clauses like $x = \alpha$ for variables x and values α from D. If the satisfiability problem for Γ^+ can be solved in polynomial time, then a search-reduces-to-decision algorithm can be used to generate all solutions to a Γ-formula. This is the only enumeration algorithm for constraint formulas known so far, and in [CH97], it was shown that it is indeed the only one for the Boolean domain. It has been conceivable that this is also true for arbitrary domains, i.e., that Γ-formulas can be efficiently enumerated if and only if the constraint satisfaction problem for Γ^+ can be solved in polynomial time. In exhibiting a new class of enumeration algorithms, we prove that this is not the case, unless P = NP. The contribution of this paper is as follows:

1. We consider refinements of the notion of efficient enumeration, demanding that the solutions can be generated not only efficiently, but also in highly customizable order. We show that this condition is in fact equivalent to the above-mentioned requirement that $\mathsf{CSP}(\Gamma^+)$ can be solved in polynomial time.
2. We develop efficient enumeration algorithms for broad classes of constraint languages Γ such that $\mathsf{CSP}(\Gamma^+)$ cannot be solved in polynomial time (unless P = NP). All of these cannot be enumerated by the known search-reduces-to-decision algorithm.
3. For the three-element case, we obtain a first step towards a full classification. We show that in the case where the constraint language satisfies an algebraic condition, our algorithms cover all cases that exist. Hence, we obtain a dichotomy theorem for enumeration in this case.
4. We show that the usual algebraic tools used in the constraint satisfaction context cannot be applied to enumeration problems for non-Boolean domains.

The structure of the paper is as follows: In Section 2, we state the necessary definitions and known results about constraints and enumeration which are relevant to our work. In Section 3, we present the above-mentioned refinement of enumeration algorithms to deal with orderings, and show that such an algorithm exists if and only if $\mathsf{CSP}(\Gamma^+)$ is tractable. Then we present our new enumeration algorithms, and show that there is a broad class of constraint languages for which these give an efficient enumeration procedure. Section 4 contains our hardness results, in particular the above-mentioned dichotomy theorem for a subclass of constraint languages over the three-element domain. In Section 5, we show that the usual algebraic tools cannot be applied for the enumeration problem. The technically most involved result of the paper are Lemma 3.15, which gives an easily verifiable criterion that a constraint language Γ needs to fullfill for our algorithms to be applicable, and Theorem 5.1, the above-mentioned negative result concerning algebraic techniques.

2 Preliminaries

2.1 Constraint Satisfaction Problems

There are many different ways to define constraint satisfaction problems (CSP). For this paper, we find the definition as logical formulas the most convenient, even for the non-Boolean case. For a domain D and a number n, an *n-ary relation over D* is a subset of D^n. In this paper, all domains are finite. A *constraint language Γ over D* is a finite set of relations over D. We say that a relation or a constraint language is *Boolean*, if the domain has cardinality 2. A *Γ-formula* is a conjunction of the form

$$\varphi = R_1(x_1^1, \dots, x_{n_1}^1) \wedge \dots \wedge R_k(x_1^k, \dots, x_{n_k}^k),$$

where the R_i are relations from Γ of arity n_i (we use the same symbol for the relation and its predicate). The set of occurring variables is denoted with $\mathrm{VAR}\,(\varphi)$. An assignment $I \colon \mathrm{VAR}\,(\varphi) \to D$ *satisfies* φ, or is a *solution of* φ, if for every $1 \leq i \leq k$, $(I(x_1^i), \dots, I(x_{k_i}^i)) \in R_i$ (we write $I \models \varphi$). The formula φ is *satisfiable* if there exists a solution of φ. The set of solutions of φ is denoted with $\mathrm{SOL}\,(\varphi)$. The problem $\mathsf{CSP}(\Gamma)$ is to decide whether a given Γ-formula is satisfiable. A constraint language Γ is tractable if $\mathsf{CSP}(\Gamma)$ can be solved in polynomial time. The study of the complexity of the problem $\mathsf{CSP}(\Gamma)$ has received much attention. For Boolean constraint languages Γ, Schaefer showed that $\mathsf{CSP}(\Gamma)$ is solvable in P or is NP-complete [Sch78]. This dichotomy also holds for the three-element case [Bul02].

We introduce some notation. For $\mathbf{v} \in D^n$, we write $\mathbf{v}[i]$ for the i-th component of $\mathbf{v}$, where $i \in \{1, \ldots, n\}$. Instead of $(I(x_1), \ldots, I(x_n))$, we write $I(x_1, \ldots, x_n)$. For one-element constraint languages $\Gamma = \{R\}$, we identify Γ and R, i.e., we say that R is tractable, speak about R-formulas, define $R^+ := \{R\}^+$, etc. For a formula φ and strings t_1 and t_2, $\varphi[t_1/t_2]$ is obtained from φ by simultaneously replacing every occurrence of t_1 with t_2. For example, if $\varphi = R(x_1, x_2, x_3)$, then $\varphi[x_2/0] = R(x_1, 0, x_3)$, and $\varphi[R/S] = S(x_1, x_2, x_3)$. For a set D and $a, b \in D$, the function $f_{a \to b} \colon D \to D$ is defined as $f(a) = b$, and $f(\alpha) = \alpha$ for all $\alpha \in D \setminus \{a\}$. For $f \colon D \to D$, $n \in \mathbb{N}$, we define: for $\mathbf{v} \in D^n$, let $f(\mathbf{v}) := (f(\mathbf{v}[1]), \ldots, f(\mathbf{v}[n]))$. For a relation $R \subseteq D^n$, let $f(R) := \{f(\mathbf{v}) \mid \mathbf{v} \in R\}$. For a formula $\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{l} R_i(x_1^i, \ldots, x_{k_i}^i)$ let $f(\varphi)(x_1, \ldots, x_n) = \bigwedge_{i=1}^{l} f(R_i)(x_1^i, \ldots, x_{k_i}^i)$, and finally for an assignment $I \colon \mathrm{VAR}(\varphi) \to D$, let $f(I)$ be the assignment defined as $f(I)(x) := f(I(x))$. For a set $C \subseteq D$, and a function $f \colon D^n \to D$, we say that f is *conservative on* C if $\alpha_1, \ldots, \alpha_n \in C$ implies $f(\alpha_1, \ldots, \alpha_n) \in C$.

Definition 2.1. *Let $f \colon D^k \to D$, and let R be an n-ary relation over D. We say R is* closed under *f, or f is a* polymorphism *of R, if for all $\mathbf{v_1}, \ldots, \mathbf{v_k} \in R$, it holds that*

$$\Big(f\big(\mathbf{v_1}[1], \ldots, \mathbf{v_k}[1]\big),\ f\big(\mathbf{v_1}[2], \ldots, \mathbf{v_k}[2]\big),\ \ldots,\ f\big(\mathbf{v_1}[n], \ldots, \mathbf{v_k}[n]\big)\Big) \in R,$$

i.e., the tuple obtained from applying f coordinate-wise to $\mathbf{v_1}, \ldots, \mathbf{v_k}$ is in R.

We denote the set of polymorphisms of R with $\mathrm{Pol}(R)$. For a constraint language Γ, $\mathrm{Pol}(\Gamma)$ is the set of functions which are polymorphisms of all relations in Γ. For the constraint satisfaction problem, it is known that the set $\mathrm{Pol}\Gamma$ determines the complexity of $\mathsf{CSP}\Gamma$ up to logspace reductions [ABI+05]. In the enumeration context, this can be shown not to be the case. A counter-example is omitted for space reasons.

Polymorphisms are related to a closure operator on the sets of relations over D, which gives an interesting Galois correspondence:

Definition 2.2. *Let Γ be a constraint language over D. The* co-clone generated by Γ, *denoted by $\langle\Gamma\rangle$, contains every relation R over D which can be expressed as a formula of the form $\exists x_1 \ldots \exists x_k \varphi$, where $x_1, \ldots, x_k \in \mathit{VAR}(\varphi)$, and φ is a $\Gamma \cup \{=\}$-formula.*

We say R can be expressed with Γ if $R \in \langle\Gamma\rangle$.

Proposition 2.3 ([JCG97]). *Let Γ_1 and Γ_2 be constraint languages. Then $\Gamma_1 \subseteq \langle\Gamma_2\rangle$ if and only if* $\mathrm{Pol}(\Gamma_2) \subseteq \mathrm{Pol}(\Gamma_1)$.

This means that the set of polymorphisms defines the expressive power of a constraint language. It is easy to see that for any constraint language, its set of polymorphism contains the identity, and is closed under composition, permutation, and identification of variables. Sets of functions closed under these operators are called *clones*. For the Boolean case, the set of clones and their inclusion structure have been identified by Emil Post [Pos41]. For domains of larger cardinality, this structure is unknown. This is one of the reasons why the study of CSP-related problems for the Boolean case is often considerably easier than for other domains. We define a Boolean relation or constraint language to be *Schaefer*, if it has a polymorphismwhich is not a constant and is not essentially unary, i.e., a polymorphism which depends on at least two of its variables.

2.2 Enumeration

For even the simplest type of formulas, we have, in general, an exponential number of solutions. Therefore for an enumeration algorithm to be considered efficient, we do not require it to give all solutions in polynomial time, but to generate the solutions with *polynomial delay* [JPY88]: for a formula φ, the algorithm has to enumerate all solutions of φ in such a way that the time between each pair of assignments, between the start of the algorithm and the first solution, and between the last solution and the termination of the algorithm is polynomial in the input size. We also require each solution to be printed exactly once. Note that such an algorithm is, in particular, output-polynomial, that means the running time is polynomial in the size of the output. We say that a constraint language Γ has an *efficient enumeration algorithm*, if there is a polynomial delay algorithm which, when given a Γ-formula φ as input, enumerates the set SOL (φ) with polynomial delay.

A slightly weaker condition is the following: if we allow the algorithm to print each assignment I up to c times, for a constant number c, then we say the algorithm is semi-efficient. The notion of semi-efficient algorithms is not of interest on its own, but arises in this work for technical reasons. For the Boolean case, semi-efficient algorithms exist if and only if efficient algorithms exist. It is easy to see that for any class of formulas, if there is a semi-efficient enumeration algorithm, then this class of formulas has a polynomial-time satisfiability problem - checking satisfiability can be done by starting the enumeration algorithm and waiting for either the first solution or the termination. Similarly, the question if there is a non-constant solution, or if there are at least $p(|\varphi|)$ many different solutions, for a polynomial p, can be answered in polynomial time using such an algorithm.

Let us consider one of the simplest types of enumeration algorithms conceivable, suggested in [Val79]. For a formula φ with VAR $(\varphi) = \{x_1, \ldots, x_n\}$, and for each $\alpha \in D$, we check if $\varphi \wedge (x_1 = \alpha)$ is satisfiable. If yes, we recursively enumerate the solutions of $\varphi[x_1/\alpha]$ augmented with the assignment $x_1 = \alpha$. If the satisfiability tests in this approach can be performed in polynomial time, then this is a polynomial-delay enumeration algorithm for the solutions of φ. Obviously, if $\mathrm{P} = \mathrm{NP}$, then all satisfiability tests of this nature can be done in polynomial time, and we always have an efficient enumeration algorithm. Therefore, for this paper we assume $\mathrm{P} \neq \mathrm{NP}$. The algorithm outlined above yields the following theorem. For a constraint language Γ over a domain D, let $\Gamma^+ := \Gamma \cup \{\{(\alpha)\} \mid \alpha \in D\}$. This can also be seen as the single relation $R \times \{(\alpha_1)\} \times \cdots \times \{(\alpha_n)\}$, where $D = (\alpha_1, \ldots, \alpha_n)$.

Theorem 2.4 ([Coh04]). *If* $\mathsf{CSP}(\Gamma^+) \in \mathrm{P}$*, then* Γ *has an efficient enumeration algorithm.*

There is a large class of constraint languages Γ which are tractable, but for which the tests of the form above are NP-complete. However, in the Boolean case it turns out that if some constraint language Γ has any (semi-) efficient enumeration algorithm, then the algorithm outlined above works [CH97]. The proof for this result makes use of a special version of the satisfiability problem, which we will define next.

Definition 2.5. *Let* Γ *be a constraint language over some domain* D*. Then* $\mathsf{CSP}^*(\Gamma)$ *is the problem to determine if a given* Γ*-formula* φ *has a solution* I *which is not constant, i.e. there are variables* $x_1, x_2 \in VAR(\varphi)$ *such that* $I(x_1) \neq I(x_2)$*.*

The main technical result in [CH97] allowing to prove that a non-Schaefer constraint language cannot be enumerated efficiently in the Boolean case is the following:

Theorem 2.6 ([CH97]). *Let* Γ *be a constraint language over the Boolean domain which is not Schaefer. Then* $\mathsf{CSP}^*(\Gamma)$ *is* NP*-complete.*

To summarize the above, for a constraint language Γ to have a polynomial-delay algorithm it is required that $\mathsf{CSP}(\Gamma) \in \mathrm{P}$. Furthermore, if $\mathsf{CSP}(\Gamma^+) \in \mathrm{P}$, such an algorithm is guaranteed to exist.In this case we even can, to a large extend, choose the order in which we want the algorithm to print out its solutions (see Theorem 3.25). Therefore, in the remainder of this paper, we are only interested in constraint languages Γ such that $\mathsf{CSP}(\Gamma) \in \mathrm{P}$, and $\mathsf{CSP}(\Gamma^+) \notin \mathrm{P}$. We show that there is a rich class of these languages which still have a polynomial-delay enumeration algorithm.

Definition 2.7. *Let* Γ *be a constraint language over a domain* $D = \{0, \ldots, k-1\}$*, and let* f *be a unary polymorphism of* Γ*, such that* $f \circ f = f$*, and* f *has minimal range. Then* $\Gamma_f^{id} := f(\Gamma) \cup \{\{f(0)\}, \ldots, \{f(k-1)\}\}$*.*

Intuitively, Γ_f^{id} is obtained from Γ by applying f to Γ, and then adding all of the constant relations $\{(\alpha)\}$ for elements α which still appear in any element of $f(\Gamma)$. It only depends on the constraint languages Γ_f^{id} if there is a polynomial time algorithm for $\mathsf{CSP}(\Gamma)$:

Proposition 2.8 ([BKJ00]). *For a constraint language* Γ *and a unary polymorphism* $f \in \mathrm{Pol}(\Gamma)$ *with minimal range, such that* $f \circ f = f$*, it holds that* $\mathsf{CSP}(\Gamma) \in \mathrm{P}$ *if and only if* $\mathsf{CSP}(\Gamma_f^{id}) \in \mathrm{P}$*.*

It can easily be seen that Γ always has a polymorphism f which has minimal range, and which fulfills $f \circ f = f$: let the range of f be minimal. Then f is a permutation on its range, and thus $f^{k!}$ acts identically on $f(D)$, where $k := |f(D)|$. Now the function $f^{k!}$ meets the conditions.

2.3 Consequences

We obtain a simple corollary from Proposition 2.8: Since any Γ with an efficient enumeration algorithm is tractable, we are only interested in languages Γ such that for any unary polymorphism $f \in \mathrm{Pol}(\Gamma)$ with minimal range, and $f \circ f = f$, Γ_f^{id} is tractable. Now assume that all unary polymorphisms f of Γ are injective, i.e., the minimal range is the cardinality of the domain. Then the identity function id is one of these polymorphisms, and thus $\Gamma_{\mathrm{id}}^{\mathrm{id}} = \Gamma^+$ is tractable. But if Γ^+ is tractable, we know that Γ has an efficient enumeration algorithm due to Theorem 2.4. Therefore, we are only interested in constraint languages Γ which have a non-injective unary polymorphism. For the three-element case, this implies that in the cases we consider, we have a constant polymorphism or a polymorphism of the form $f_{a \to b}$.

Proposition 2.9. *Let Γ be a constraint language such that $\mathsf{CSP}(\Gamma) \in \mathrm{P}$, and $\mathsf{CSP}(\Gamma^+) \notin \mathrm{P}$. Then Γ has a non-injective unary polymorphism f which, restricted to its range, is the identity.*

3 Enumeration Algorithms

3.1 Tools

It is often convenient to look at only one relation instead of a constraint language. The following lemma shows that we can make this restriction. It implies that if we can decide, for any one-element constraint language, if it has a polynomial-delay enumeration algorithm, then we can decide this question for arbitrary finite constraint languages. For the remainder of this paper, we restrict our study to single relations.

Lemma 3.1. *Let $\Gamma = \{R_1, \dots, R_n\}$ be a finite constraint language. Then Γ has a polynomial-delay enumeration algorithm (semi-efficient enumeration algorithm, resp.) if and only if for every subset $S = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$, the relation $R_S := R_{i_1} \times R_{i_2} \times \cdots \times R_{i_k}$ has such an algorithm.*

Proof. If Γ has an enumeration algorithm of any of this type, then this also holds for the relations R_S, since we can easily express clauses using R_S as conjunctions of R_i relations. For the other direction, let $\varphi = \bigwedge_{i=1}^{l} R_{t_i}(x_1^i, \dots, x_{n_{t_i}}^i)$ be a Γ-formula (where $n_{t_i} = \mathrm{ar}(R_{t_i})$). Let $S := \{t_1, \dots, t_l\}$, i.e., S contains the indices of the relations appearing in the formula φ. Without loss of generality, let $S = \{1, \dots, k\}$ for some $k \leq n$, and let $t_1 = 1, \dots, t_k = k$. For $i \in \{1, \dots, l\}$, let C_i be the following clause:

$$C_k = \left(\bigwedge_{t \in S \setminus \{t_i\}} R_t(x_1^t, \dots, x_{n_t}^t) \right) \wedge R_{t_i}(x_1^i, \dots, x_{n_{t_i}}^i),$$

which is a conjunction or C_i and other clauses which also appear in φ. Then each C_i is a clause containing exactly one application of each relation R_t for $t \in S$. Therefore, each clause C_i can be expressed as one application of the relation R_S. Now consider the formula

$$\psi := \bigwedge_{i=1}^{l} C_i.$$

Then each clause of ψ appears in the original formula φ, and each clause from φ appears in ψ, i.e., the formulas are equivalent. Therefore we can apply the enumeration algorithm to the formula ψ to enumerate the solutions of φ. □

Definition 3.2. *Let φ be a formula over some constraint language Γ. We consider the clauses of the formula as an undirected graph, where two clauses are connected if they share a variable. We say that φ is* connected*, if this graph is connected.*

Proposition 3.3. *Let Γ be a constraint language, such that there is an efficient enumeration algorithm for connected Γ-formulas. Then Γ has an efficient enumeration algorithm.*

Proof. Let φ be a formula which is not connected. Then φ can be written as $\varphi_1 \wedge \cdots \wedge \varphi_k$, where φ_i is connected for $i \in \{1, \ldots, k\}$. This representation can be computed in polynomial time, by identifying the connected components of the formula graph. By our prerequisites, the solutions of φ_i can be enumerated with polynomial delay. Therefore we can, without loss of generality, assume that all of the φ_k are satisfiable: if one of these is unsatisfiable, then φ is unsatisfiable as well, and we can test this in polynomial time. Since for $i \neq j$, φ_i and φ_j do not share variables, it is obvious that

$$\mathrm{SOL}\left(\varphi\right) = \left\{I_1 \cup \cdots \cup I_k, \text{ such that } I_j \models \varphi_j\right\},$$

where the union of functions with disjoint domains is defined as usual. Therefore the solutions of φ can easily be obtained from the solutions of the φ_j, for which we have efficient algorithms available. □

The Galois correspondence explained above does not apply to the enumeration problems we study, as we will see in Sect. 5. However, it is easy to see that a restricted closure operator helps here: this is a weaker version of the usual co-clone closure operator, where we do not allow the introduction of existentially quantified variables. Many aspects of the standard algebraic approach used in the context of constraint satisfaction problems turn out to be useful for our purposes as well.

Definition 3.4. *Let Γ be a constraint language. Then $\langle\Gamma\rangle_{\nexists}$ contains all relations R which can be defined with $\Gamma \cup \{=\}$-formulas.*

It is easy to see, given the definition of $\langle\Gamma\rangle$, that the following proposition holds:

Proposition 3.5. *Let Γ be a constraint language.*

1. *Let Γ' be a finite subset of $\langle\Gamma\rangle_{\nexists}$. If Γ has an efficient enumeration algorithm, then Γ' has an efficient enumeration algorithm as well.*
2. *Let Γ' be a finite subset of $\langle\Gamma\rangle$. Then $\mathsf{CSP}(\Gamma') \leq_m^{\log} \mathsf{CSP}(\Gamma)$.*

Proof. The proof is very easy:

1. Simply convert Γ'-formulas to Γ-formulas. Any occurring equalities can be dealt with using variable identification. This gives a polynomial time transformation.
2. This follows from Proposition 3.5. The logspace reduction is a corollary from Reingold's discovery that search in undirected graphs can be performed in LOGSPACE [Rei05]. A proof can be found in [ABI+05].

□

For the enumeration problem, another closure operator is helpful. This closure operator lies between the previously defined operators: for a relation R, it holds that $\langle R\rangle_{\nexists} \subseteq \langle R\rangle_{\text{cons}} \subseteq \langle R\rangle$. While the previously defined operator $\langle.\rangle_{\nexists}$ corresponds to efficient enumerability, the operator $\langle.\rangle_{\text{cons}}$ deals with the case of semi-efficient enumeration algorithms.

Definition 3.6. *Let R and S be relations over some domain D. Assume that $R \in \langle S\rangle$, i.e. there is some constant c such that R can be written as*

$$R(x_1, \ldots, x_n) \iff \exists y_1 \ldots \exists y_c S(a_1^1, \ldots, a_m^1) \wedge \cdots \wedge S(a_1^p, \ldots, a_m^p),$$

where $a_j^i \in \{y_1, \ldots, y_c, x_1, \ldots, x_n\}$. We write $C^R(x_1, \ldots, x_n)$ for the formula on the right side of this equivalence. We say $R \in \langle S\rangle_{cons}$, if the y_i can be "re-used" in the following way: for every R-formula $\varphi = \bigwedge_{i=1}^{l} R(x_1^i, \ldots, x_n^i)$, it holds that

$$\varphi \iff \exists y_1 \ldots \exists y_c \bigwedge_{i=1}^{l} C^R(x_1^i, \ldots, x_n^i).$$

This operator does not necessarily preserve efficient enumerability, but it does preserve semi-efficient enumerability. Therefore, this is very useful when considerung reductions to the Boolean case, since here, these enumeration notions coincide (Proposition 3.26).

Proposition 3.7. *Let R be a relation over some domain. Then R has a semi-efficient enumeration algorithm if and only if every relation from $\langle R\rangle_{cons}$ has one.*

Proof. This is obvious given the definition. Let $R' \in \langle R\rangle_{\text{cons}}$. Then we can transform every R'-formula φ' to an R-formula φ adding only constant-many new existentially quantified variables. Now enumerating the solutions of φ, where we do not print the assignments for the new existentially quantified variables, gives each solution of φ' at most $c \cdot |D|^k$ often, where D is the domain of the relation, k is the number of existentially quantified variables in φ, and c is the constant from the semi-efficient algorithm for R. Therefore, R' has a semi-efficient enumeration algorithm. The converse is trivial, since $R \in \langle R\rangle_{\text{cons}}$. □

3.2 An Example

We give an example for a relation R which has a polynomial-delay enumeration algorithm, and where R^+ is not tractable—i.e., we show that there are more efficiently enumerable relations than the ones covered by Theorem 2.4. We then show how the ideas highlighted in this example can be generalized.

Example 3.8. Let $R := \{(2,0,0,2),(2,0,2,0),(2,2,0,0),(1,0,0,1),(1,0,1,0),(1,1,0,0),(1,0,0,0)\}$. R is efficiently enumerable, and R^+ is not tractable.

Proof Sketch. $\mathsf{CSP}(R^+)$ is NP-complete: it is widely known that CSP(1-in-3) is NP-complete (this follows from [Sch78]), where 1-in-3 $= \{(1,0,0),(0,1,0),(0,0,1)\}$. CSP(1-in-3) reduces to $\mathsf{CSP}(R^+)$ by forcing one additional variable to 2.

To see that R has an efficient enumeration algorithm, consider the following approach: it can be verified that $f_{2\to1} \in \mathrm{Pol}(R)$, and that $f_{2\to1}(R)$ is closed under the Boolean AND operator. Thus, $f_{2\to1}(R)$ is Schaefer, and has an efficient enumeration algorithm due to [CH97]. It can be seen that $f_{2\to1}(R) = R \cap \{0,1\}^4$. Therefore enumerating, for a given R-formula φ, the solutions of $f_{2\to1}(\varphi)$, is the same as enumerating all solutions of φ which assign each of the variables $x \in \mathrm{VAR}(\varphi)$ one of the values 0 and 1.

If we can enumerate, with polynomial delay, for each solution I as above, all "compatible" solutions J for which $f_{2\to1}(J) = I$, then we can enumerate all solutions of φ. This is because since $f_{2\to1} \in \mathrm{Pol}(R)$, for an arbitrary solution J of φ, $f_{2\to1}(J)$ is a solution of φ as well. Therefore the solution J appears in this enumeration scheme.

It remains to prove that for each solution $I \models \varphi$, $I\colon \mathrm{VAR}(\varphi) \to \{0,1\}$, we can enumerate the set of all J fulfilling the above conditions efficiently. This is, in essence, a Boolean problem: given such an assignment I, we want to exchange some of the occurring 1s with 2s, such that the assignment J obtained this way satisfies φ. Variables x such that $I(x) = 0$ are left unmodified. Therefore, this is a Boolean problem involving the values 1 and 2. It is natural that there is a Boolean constraint language, which we will later introduce as $\Gamma_R^{E_1\to E_2}$, that can be used to express the "possibilities of changing 1s into 2s". Intuitively, $\Gamma_R^{E_1\to E_2}$ is obtained as follows: for each $\mathbf{v} \in \{0,1\}^4$, consider the relation $R_{\mathbf{v}}$, containing all tuples $\mathbf{v}' \in R$ such that $f_{2\to1}(\mathbf{v}') = \mathbf{v}$. This relation describes the combinations of 2s and 1s that are "allowed". Since we are not interested in the occurring 0s here—they are fixed and we do not change these assignments—we only look at those components of the relation in which 1s and 2s appear. We will introduce the constraint language $\Gamma_R^{E_1\to E_2}$ arising here formally, and prove that R has an efficient enumeration algorithm in Corollary 3.16. □

3.3 Partial enumerability

A central idea in our algorithms are *partial assignments*. These assign, to each variable, a set of possible values, where the appearing subsets form a partition of the domain. We introduce some notation on partitions and equivalence relations.

Let D be a domain, and E a partition of D. Trivially, partitions and equivalence relations correspond to each other. We often identify E with its corresponding equivalence relation, denoted by $\sim_E$. The *discrete partition of* D, D^{disc}, corresponding to the equality predicate on D, is defined as $D^{\text{disc}} = \{\{\alpha\} \mid \alpha \in D\}$. Its opposite, the *indiscrete partition of* D, D^{indisc}, corresponds to the equivalence relation on D where all elements are equivalent, i.e., $D^{\text{indisc}} = \{D\}$. We often identify a partition E and the set of unary relations representing the classes in E.

Definition 3.9.
- *Let E be a partition of the domain D. Then f^E is the function assigning each $\alpha \in D$ its equivalence class., i.e. the unique function $f^E\colon D \to E$ such that $\alpha \in f(\alpha)$ for all $\alpha \in D$. We say that f^E is the* canonical *homomorphism from D to E.*
- *For a relation R, R/E is defined as $f^E(R)$.*
- *For partitions E_1 and E_2 of a domain D we say E_2 is a refinement of E_1 ($E_2 \leq E_1$) if $\alpha \sim_{E_2} \beta$ implies $\alpha \sim_{E_1} \beta$.*

Let Γ be a constraint language over D, E_1 a partition of D, and φ a Γ-formula. We say $I\colon \text{VAR}(\varphi) \to E_1$ is a *partial E_1-assignment.* If E_2 is a refinement of E_1 and J is a partial E_2-assignment, then we say J is *compatible with* I if for all $x \in \text{VAR}(\varphi)$, $J(x) \subseteq I(x)$. We also apply this notion to tuples, i.e. if $\mathbf{v_1} \in E_1^n$, $\mathbf{v_2} \in E_2^n$, then we say $\mathbf{v_2}$ is compatible with $\mathbf{v_1}$ if for all $i \in \{1, \ldots, n\}$, $\mathbf{v_2}[i] \subseteq \mathbf{v_1}[i]$. We identify partial D^{disc}-assignments J for φ and assignments $J\colon \text{VAR}(\varphi) \to D$, i.e. for such an assignment we say J is compatible with I if $J(x) \in I(x)$ for all $x \in \text{VAR}(\varphi)$. A partial assignment I is a *partial E_1-solution of* φ if there exists some $J\colon \text{VAR}(\varphi) \to D, J \models \varphi$ such that J is compatible with I. We denote the set of partial E_1-solutions of φ with $\text{SOL}^{E_1}(\varphi)$.

Definition 3.10. *Let E_1 and E_2 be partitions of D, such that E_2 is a refinement of E_1, and R a relation over D.*

- *R is efficiently* E_1-enumerable*, if there is a polynomial delay algorithm which, given an R-formula φ, enumerates $SOL^{E_1}(\varphi)$.*
- *R is efficiently* $E_1 \to E_2$-enumerable*, if there exists a polynomial-delay algorithm which, given an R-formula φ and an assignment $I\colon VAR(\varphi) \to E_1$, enumerates all partial solutions $J \in SOL^{E_2}(\varphi)$ which are compatible with I.*
- *R is efficiently* $E_1 \to D$-enumerable*, if R is efficiently $E_1 \to D^{disc}$-enumerable.*

The following theorem is one of our main results, and shows how our approach can be used to obtain enumeration algorithms. In Sect. 3.4, we will show that the prerequisites for this theorem are met by a large class of relations.

Theorem 3.11. *Let D be a domain, E_1 and E_2 partitions of D such that E_2 is a refinement of E_1, and let R be a relation over D.*

1. *If R is efficiently E_1-enumerable and efficiently $E_1 \to E_2$-enumerable, then R is efficiently E_2-enumerable.*
2. *If $\mathsf{CSP}(\{R\} \cup E_1) \in \mathrm{P}$ and R is $E_1 \to D$-enumerable, then R has an efficient enumeration algorithm.*

Proof.
1. Let φ be an R-formula. We enumerate $\text{SOL}^{E_1}(\varphi)$. For each partial solution I printed by this algorithm, because R is $E_1 \to E_2$-enumerable, we can enumerate all partial solutions $J\colon \text{VAR}(\varphi) \to E_2$ such that J is compatible with I.
It is obvious that this algorithm has polynomial delay, since both of the nested algorithms work with polynomial delay, and for each of the solutions $I \in \text{SOL}^{E_1}(\varphi)$, there is a compatible E_2-solution to φ which is printed out. No solution is printed twice, since if $I_1, I_2 \in \text{SOL}^{E_1}(\varphi)$, $I_1 \neq I_2$, then no E_2-solution J can be compatible with both I_1 and I_2. Every solution printed out is indeed a partial E_2-solution of φ, since the algorithm securing the E_2-enumerability condition only prints partial solutions of the formula. Each E_2-solution J of φ is printed, since we can, in a canonical manner, obtain a partial solution $I \in \text{SOL}^{E_1}(\varphi)$ such that J is compatible with I. Therefore, the enumeration algorithm works correctly.
2. This follows immediately from part 1 and Theorem 3.25.

□

This theorem generalizes Theorem 2.4: R^+ is tractable if and only if $\{R\} \cup D^{\text{disc}}$ is. Further, every relation is trivially efficiently $D^{\text{disc}} \to D$ enumerable.

3.4 Nested problems

In this section we present results about new types of enumeration algorithms. The goal here is to find conditions for partial enumerability, i.e. to present conditions which guarantee that some relation is efficiently E-enumerable or efficiently $E_1 \to E_2$-enumerable: We show here that there is a rich class of relations meeting the conditions required in Theorem 3.11. The conditions we give can be verified by looking at a constraint language over a smaller domain, thus giving an inductive approach. The result for partial E-enumerability is quite easy, but needs special prerequisites, which we define now.

Definition 3.12. *Let R be a relation over D, and E a partition of D. We say that E' is a* representation system of E compatible with R, *if E' contains exactly one element of each equivalence class in E, and $f^{E'} \circ f^E \colon D \to E' \in \mathrm{Pol}(R)$, where $f^{E'} \colon E \to E'$ is the function assigning each equivalence class its corresponding value in E'.*

Note that $f^{E'} \circ f^E$ is the canonical function assigning each $\alpha \in D$ its representative in E'. If this function is a polymorphism, then for each partial E-solution I of an R-formula φ, $f^{E'}(I)$ is a solution of φ. This is used in the proof for the following Lemma:

Lemma 3.13. *Let R be a relation over D, and let E be a partition of D such that there is a representation system of E compatible with R. Then R is efficiently E-enumerable if and only if $^R/_E$ is efficiently enumerable.*

Proof. The function assigning each R-formula φ its corresponding $^R/_E$-formula by exchanging each clause $R(x_1, \dots, x_n)$ with the clause $^R/_E(x_1, \dots, x_n)$ clearly is a bijection between R- and $^R/_E$-formulas which is computable in polynomial time, as is its inverse. Therefore, to prove the lemma, it suffices to show that for some R-formula φ, the partial E-assignment I is a partial E-solution of φ if and only if it is a solution of $\varphi\,[R/{}^R/_E]$. Let E' be the representation system.

Fist, let $I \in \mathrm{SOL}^E(\varphi)$. Then there exists a solution $J \colon \mathrm{VAR}(\varphi) \to D$ such that $J \models \varphi$ and J is compatible with I. We show that $I \models \varphi\,[R/{}^R/_E]$. Let $^R/_E(x_1, \dots, x_n)$ be a clause in $\varphi\,[R/{}^R/_E]$. Then, since $J \models \varphi$, we know that $J(x_1, \dots, x_n) \in R$. Thus, by definition of $^R/_E$, we know that $f^E(J(x_1, \dots, x_n)) \in {}^R/_E$. Since J is compatible with I, we know that $f^E(J) = I$. This implies that $I \models \varphi\,[R/{}^R/_E]$.

Now, let $I \models \varphi\,[R/{}^R/_E]$, and let $J \colon \mathrm{VAR}(\varphi) \to E'$ be defined as $J = f^{E'}(I)$. Then J is compatible with I by definition of $f^{E'}$. We show that $J \models \varphi$, implying that $I \in \mathrm{SOL}^E(\varphi)$. For this, let $R(x_1, \dots, x_n)$ be a clause in φ. Since $I \models \varphi\,[R/{}^R/_E]$, we know that there is some $\mathbf{v} \in R$, such that $f^E(\mathbf{v}) = I(x_1, \dots, x_n)$. Thus, since $f^{E'} \circ f^E$ is a polymorphism of R, we know that $f(\mathbf{v}) \in R$. By choice of J, this implies that $f^{E'}(\mathbf{v}) = J(x_1, \dots, x_n) \in R$, concluding the proof. □

This concept can be illustrated at the relation R given in Example 3.8. Recall that R has the polymorphism $f_{2\to1}$. This gives a canonical partition of the domain $D = \{0, 1, 2\}$ of R: let $E := \{\{0\}, \{1, 2\}\}$. Since $f_{2\to1}$ is a polymorphism of R, $E' := \{0, 1\}$ is a representation system of E compatible with R: $f^{E'} \circ f^E$ is just the function $f_{2\to1}$.

We showed above that $f_{2\to1}(R)$ is Schaefer. With the above, this is the same relation as $^R/_E$. Therefore, $^R/_E$ has an efficient enumeration algorithm due to Proposition 3.26, and since E' is a representation system of E compatible with R, we know that R is efficiently E-enumerable by Lemma 3.13. This can also be seen directly: if we consider the relation $f_{2\to1}(R) = R \cap \{0,1\}^4$, then this enables us, for a given R-formula φ, to enumerate all solutions $I \colon \mathrm{VAR}(\varphi) \to \{0, 1\}$ (cp. the proof sketch of Example 3.8). Now for any solution $J \models \varphi$, the "Boolean solution" $f_{2\to1}(J)$ is also a solution of φ, and it can easily be seen that $f_{2\to1}(J) \sim_E J$.

The corresponding criterion for $E_1 \to E_2$-enumerability is more technical, but holds without prerequisites: to decide whether a relation R is efficiently $E_1 \to E_2$ enumerable, it suffices to consider a constraint language $\Gamma_R^{E_1 \to E_2}$, which we define now. This is the language Γ mentioned in the proof sketch for Example 3.8, allowing us, for each "Boolean solution" I, to enumerate all solutions J such that $f_{2\to1}(J) = I$. The definition is more general: we do not necessarily want to enumerate, to a given partial solution I, all "fitting" solutions J, but we also allow these solutions J to be partial solutions, with respect to a refinement E_2 of E_1. Remember that in Example 3.8, we were not interested in those parts of the relation which are set to 0, we only wanted to get all possible combinations of 1 and 2. The natural generalization is that we are not interested in classes from E_1 which are not partitioned further in E_2. For partial E_1-solutions I and partial E_2-solutions J compatible with I, and variables x such that $I(x) = D_i \in E_1$, if $D_i \in E_2$, then $J(x) = D_i$ has to hold as well. Therefore, these aspects of the relation R are not interesting when determining possible E_2 solutions compatible with a given E_1 solution. Hence, the corresponding components of the relation R are disregarded in the following definition:

Definition 3.14. *1. Let R be an n-ary relation, and let $I \subseteq \{1, \ldots, n\}$, such that $I = \{i_1, \ldots, i_k\}$. Let $R_I(x_{i_1}, \ldots, x_{i_k})$ be the relation obtained from $R(x_1, \ldots, x_n)$ by existentially quantifying all of the variables x_j such that $j \notin I$, i.e.*

$$R_I(x_{i_1}, \ldots, x_{i_k}) \Leftrightarrow \exists j_1 \ldots \exists j_{n-k} R(x_1, \ldots, x_n),$$

where $\{1, \ldots, n\} \setminus I = \{j_1, \ldots, j_{n-k}\}$.

2. Let R be an n-ary relation over a domain D. Let E_1 and E_2 be partitions of D such that E_2 is a refinement of E_1. For $\mathbf{v} \in E_1^n$, we define

$$\mathbf{v}^{E_1 \to E_2} := \{\mathbf{t} \in E_2^n \mid \mathbf{t} \text{ compatible with } \mathbf{v}, \text{ there is a } \mathbf{u} \in R \text{ compatible with } \mathbf{t}\},$$

$$I_{\mathbf{v}} = \{i \in \{1, \ldots, n\} \mid \mathbf{v}[i] \notin E_2\}, \qquad R_{\mathbf{v}}^{E_1 \to E_2} := \mathbf{v}_{I_{\mathbf{v}}}^{E_1 \to E_2},$$

and finally, let $\Gamma_R^{E_1 \to E_2} := \left\{R_{\mathbf{v}}^{E_1 \to E_2} \mid \mathbf{v} \in E_1^n\right\}$.

The relation $\mathbf{v}_{I_{\mathbf{v}}}^{E_1 \to E_2}$ describes sets of solutions compatible with a given partial solution: for a constraint application $R(x_1, \ldots, x_n)$ and a tuple $v \in E_1^n$, the set $\mathbf{v}^{E_1 \to E_2}$ contains the partial solutions $J \colon \{x_1, \ldots, x_n\} \to E_2$ compatible with $\mathbf{v}$. As explained above, in those cases where the equivalence classes from E_2 also appear in E_1, the corresponding values are disregarded by existentially quantifying over the corresponding parts of the relation $\mathbf{v}^{E_1 \to E_2}$. Observe that $\Gamma_R^{E_1 \to E_2}$ is a constraint language over the domain $\{D_2 \in E_2 \mid D_2 \notin E_1\}$, i.e. the domain containing those equivalence classes from E_2 which are finer than the equivalence classes in E_1.

An important case is when $E_2 = D^{\text{disc}}$. Then the relation $R_{\mathbf{v}}^{E_1 \to E_2}$ describes the possible "real" solutions which are compatible with a given partial E_1 assignment. Here, equivalence classes which only contain one element play a crucial role. Since D^{disc} cannot refine these, elements in such classes "disappear". A very important special case is when $E_2 = D^{\text{disc}}$, and all but one of the equivalence classes in E_1 are singletons. In particular, if $D_1 = \{a, b\} \in E_1$ for some $a \neq b \in D$, and $|D_i| = 1$ for all other $D_i \in E_1$, we say R *is $\{a, b\}$-Schaefer* if $\Gamma_R^{E_1 \to D^{\text{disc}}}$ is Schaefer. Note that this language is over the Boolean domain $\{a, b\}$, and thus this terminology can be applied. From Proposition 3.26, it is immediate that if R is $\{a, b\}$-Schaefer, then $\Gamma_R^{E \to D^{\text{disc}}}$ has an efficient enumeration algorithm (where E is the partition containing the set $\{a, b\}$ and the remaining elements from D as singletons).

We take another look at the relation R from Example 3.8, and construct the constraint language $\Gamma_R^{E \to D}$. Remember that $D = \{0, 1, 2\}$, and we chose the partition $E = \{\{0\}, \{1, 2\}\}$. We denote the equivalence class $\{0\}$ with $\overline{0}$, and the class $\{1, 2\}$ with $\overline{1}$. Let $\mathbf{v_1} := (\overline{1}, \overline{0}, \overline{0}, \overline{1})$, $\mathbf{v_2} := (\overline{1}, \overline{0}, \overline{1}, \overline{0})$, $\mathbf{v_3} := (\overline{1}, \overline{1}, \overline{0}, \overline{0})$, $\mathbf{v_4} := (\overline{1}, \overline{0}, \overline{0}, \overline{0})$. Every tuple in R is E-equivalent to one of these four tuples. Therefore, for any $\mathbf{v} \notin \{\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \mathbf{v_4}\}$, the relation $\mathbf{v}^{E \to D}$ is empty. By definition, the following equations hold:

$$\begin{array}{llll}
\mathbf{v_1}^{E \to D} & = \{(2,0,0,2), (1,0,0,1)\} \qquad & R_{\mathbf{v_1}}^{E \to D} & = \{(2,2), (1,1)\} \\
\mathbf{v_2}^{E \to D} & = \{(2,0,2,0), (1,0,1,0)\} & R_{\mathbf{v_1}}^{E \to D} & = \{(2,2), (1,1)\} \\
\mathbf{v_3}^{E \to D} & = \{(2,2,0,0), (1,1,0,0)\} & R_{\mathbf{v_1}}^{E \to D} & = \{(2,2), (1,1)\} \\
\mathbf{v_4}^{E \to D} & = \{(1,0,0,0)\} & R_{\mathbf{v_1}}^{E \to D} & = \{(1)\}
\end{array}$$

Therefore the constraint language $\Gamma_R^{E \to D}$ only contains the relations $\{(2,2), (1,1)\}$ and $\{(1)\}$. If we view these as relations over the Boolean domain by e.g. identifying the occurring 2s with the Boolean 0, then this language language is closed under the Boolean AND operator. Therefore, $\Gamma_R^{E \to D}$ is Schaefer, thus this language has an efficient enumeration algorithm. This implies that R is $E \to D$-enumerable: for a relation R it holds that $\Gamma_R^{E_1 \to E_2}$ is enumerable if and only if R is $E_1 \to E_2$ enumerable, as shown by the following Lemma 3.15. The characterization it gives can be used, with Theorem 3.11, to prove the existence of efficient enumeration algorithms inductively. The proof uses the ideas explained in the proof sketch for Example 3.8, and in the discussion of the language $\Gamma_R^{E_1 \to E_2}$ above.

Lemma 3.15. *Let R be an n-ary relation over a domain D, and let E_1, E_2 be partitions of D such that E_2 is a refinement of E_1. then The following holds:*

1. *The following conditions are equivalent:*
   (i) $\Gamma_R^{E_1\to E_2}$ is efficiently enumerable
   (ii) R is efficiently $E_1 \to E_2$-enumerable.
2. *If $\{R\} \cup E1$ is tractable, then $\Gamma_R^{E_1\to E_2}$ is tractable*

Proof. We start with proving a strong connection between pairs R-formulas and partial E_1-assignments on the one hand, and $\Gamma_R^{E_1\to E_2}$-formulas on the other. Let

$$\varphi = R(\mathbf{x_1}) \wedge \cdots \wedge R(\mathbf{x_m})$$

be an R-formula, where $\mathbf{x_i}$ are vectors of variables, and let I_1 be a partial E_1-assignment for φ. Let ψ be the $\Gamma_R^{E_1\to E_2}$-formula defined by

$$\psi = R_{I_1(\mathbf{x_1})}^{E_1\to E_2}(\mathbf{x_1}_{\{i|I_1(\mathbf{x_1}[i])\notin E_2\}}) \wedge \cdots \wedge R_{I_1(\mathbf{x_m})}^{E_1\to E_2}(\mathbf{x_m}_{\{i|I_1(\mathbf{x_m}[i])\notin E_2\}}),$$

where $I_1(\mathbf{x_j}) = (I_1(\mathbf{x_j}[1]), \ldots, I_1(\mathbf{x_j}[n]))$ and $\mathbf{x_j}_{\{i|I_1(\mathbf{x_j}[i])\notin E_2\}}$ is the tuple containing the components from $\mathbf{x_j}$ that are not equivalence classes from E_2.

The intuition is that for a clause $R(\mathbf{x_j})$ in the original formula φ, the clause $R_{I_1(\mathbf{x_j})}^{E_1\to E_2}(\mathbf{x_j}_{\{i|I_1(\mathbf{x_j}[i])\notin E_2\}})$ describes the possible values that a partial E_2-solution I_2 which is compatible with I_1 can assign to the variables, and still satisfy the original clause. In the new clause, only those variables from $\mathrm{VAR}\,(\varphi)$ appear whose values are "more determined" by the partial E_2-solutions we are interested in, than by the partial E_1-assignment that we are given. These are exactly those components of the vector $\mathbf{x_j}$ for which $I_1(\mathbf{x_j})$ contains an equivalence class from E_1 that is properly refined by E_2.

We show the following: every partial E_2-solution of φ which is compatible with I_1 can be restricted to a solution of ψ, and every solution of ψ can be extended to exactly one partial E_2-solution of φ compatible with I_1.

Let $I_2 \colon \mathrm{VAR}\,(\varphi) \to E_2$ a partial E_2-assignment for φ. If $I_2 \in \mathrm{SOL}^{E_2}\,(\varphi)$ and if I_2 is compatible with I_1, then there is some $I_3 \in \mathrm{SOL}\,(\varphi)$ such that $I_3(x) \in I_2(x) \subseteq I_1(x)$ for every $x \in \mathrm{VAR}\,(\varphi)$. That means $I_3(\mathbf{x_i})$ is compatible with $I_2(\mathbf{x_i})$ and that is compatible with $I_1(\mathbf{x_i})$ for every $i \in \{1, \ldots, m\}$. Hence, $I_2(\mathbf{x_i}) \in I_1(\mathbf{x_i})^{E_1\to E_2}$. It follows that $I_2(\mathbf{x_i})_{\{j|I_1(\mathbf{x_i}[j])\notin E_2\}} \in R_{I_1(\mathbf{x_i})}^{E_1\to E_2}$ for all $i \in \{1, \ldots, m\}$, which means I_2 restricted to $\mathrm{VAR}\,(\psi)$ is from $\mathrm{SOL}\,(\psi)$.

For the other direction let $I_2 \colon \mathrm{VAR}\,(\psi) \to E_2$ be a solution of ψ. We define $I_2' \colon \mathrm{VAR}\,(\varphi) \to E_2$ by $I_2'(x) = I_2(x)$ for $x \in \mathrm{VAR}\,(\psi)$ and $I_2'(x) = I_1(x)$ otherwise (note that $I_1(x) \in E_2$ in these cases). Since $I_2'(\mathbf{x_i})_{\{j|I_1(\mathbf{x_i}[j])\notin E_2\}} = I_2(\mathbf{x_i}_{\{j|I_1(\mathbf{x_i}[j])\notin E_2\}}) \in R_{I_1(\mathbf{x_i})}^{E_1\to E_2}$, it holds that $I_2'(\mathbf{x_i}) \in I_1(\mathbf{x_i})^{E_1\to E_2}$. Therefore I_2' is compatible with I_2 and it exist a solution of φ that is compatible with I_2', which means that $I_2' \in \mathrm{SOL}^{E_2}\,(\varphi)$. Since for every partial E_2 assignment I_2 compatible with I_1 holds that I_1 and I_2 restricted on $\mathrm{VAR}\,(\varphi) \setminus \mathrm{VAR}\,(\psi)$ are equal, this is the only possibility to extend I_2 to a partial E_2-assignment of φ.

Now we prove the lemma:

1. Let $\Gamma_R^{E_1\to E_2}$ be efficiently enumerable, and let φ be an R-formula and I_1 a partial E_1 assignment for φ. Then the solutions of the $\Gamma_R^{E_1\to E_2}$-formula ψ constructed from φ and I_1 as above can be enumerated efficiently. So the following describes an algorithm with polynomial delay: enumerate all solutions of ψ and as soon as as such a solution I_2 is generated print out I_2'. Due to the above this enumerates all partial E_2-solutions of φ compatible with I_1 exactly once, that means R be efficiently $E_1 \to E_2$-enumerable.
   Now let R be efficiently $E_1 \to E_2$-enumerable. Let

   $$\psi = R_{\mathbf{v_1}}^{E_1\to E_2}(\mathbf{x_1}) \wedge \cdots \wedge R_{\mathbf{v_m}}^{E_1\to E_2}(\mathbf{x_m}),$$

   where $\mathbf{v_i} \in E_1^n$ and $\mathbf{x_i}$ is a vector of variables with the same length as $\mathbf{v_i}_{\{j|\mathbf{v_i}[j]\notin E_2\}}$ be a $\Gamma_R^{E_1\to E_2}$-formula.
   Since $R_{\mathbf{v_i}}^{E_1\to E_2}$ is a projection of $\mathbf{v_i}^{E_1\to E_2}$ which is build from E_2-vectors that are compatible with the E_1-vector $\mathbf{v_i}$, it holds that every solution must map $\mathbf{x_i}[j]$ to a subset of $\mathbf{v_i}_{\{k|\mathbf{v_i}[k]\notin E_2\}}[j]$
   We present an enumerating algorithm for ψ with polynomial delay.

(a) For every $\mathbf{x_{i_1}}[j_1] = \mathbf{x_{i_2}}[j_2]$ check if $\mathbf{v_{i_1}}_{\{k|\mathbf{v_{i_1}}[k]\notin E_2\}}[j_1] = \mathbf{v_{i_2}}_{\{k|\mathbf{v_{i_2}}[k]\notin E_2\}}[j_2]$. If this does not hold, then these sets are disjoint, and therefore ψ is unsatisfiable, STOP.

(b) Let $\mathbf{y_i} \in E_2^n$ such that $\mathbf{y_i}_{\{k|\mathbf{v_i}[k]\notin E_2\}} = \mathbf{x_i}$ and the other positions are filled with new distinctive variables. Let $\varphi = R(\mathbf{y_1}) \wedge \cdots \wedge R(\mathbf{y_m})$ and $I\colon \mathrm{VAR}\,(\varphi) \to E_1$ defined by $I(\mathbf{y_i}[j]) = \mathbf{v_i}[j]$ be a partial E_1-assignment for φ. Note that in step (a) we assured that I is well defined and that ψ can be obtained from φ and I by the construction above.

(c) Enumerate all partial E_2-solutions compatible with I of φ, restrict them to $\mathrm{VAR}\,(\psi)$, and print them out.

Due to what we showed above, this algorithm is correct.

2. Let $\{R\} \cup E_1$ be tractable and ψ be a $\Gamma_R^{E_1\to E_2}$-formula. The forementioned algorithm where step (c) is exchanged with the following decides ψ.

(c) Determine whether the $\{R\} \cup E_1$-formula

$$\varphi \wedge \bigwedge_{i=1,j=1}^{i=m,j=n} (\mathbf{y_i}[j] \in \mathbf{v_i}[j])$$

is satisfiable and print out the result.

□

Corollary 3.16. *Let R be a relation over the domain D such that $a, b \in D$, $f_{a\to b} \in \mathrm{Pol}(R)$, $f_{a\to b}(R)$ has an efficient enumeration algorithm, and R is $\{a,b\}$-Schaefer. Then R has an efficient enumeration algorithm.*

For the Boolean case, the question whether a constraint language Γ has a polynomial-delay enumeration algorithm is answered by Proposition 3.26. Therefore, Corollary 3.16 gives an inductive criteria for the existence of efficient algorithms.

Proof. Let $D_1 := \{a,b\}$, and let $D = D_1 + \cdots + D_k$ where $|D_i| = 1$ for $2 \le i \le k$. Since $f_{a\to b} \in \mathrm{Pol}(R)$, it follows that $I\colon \mathrm{VAR}\,(\varphi) \to \{\{D_1\}, \ldots, \{D_k\}\}$ is a partial solution of φ if and only if $I' \models \varphi$, where $I'(x) \in I(x)$, and $I'(x) = b$ if $I(x) = \{a,b\}$. Since $f_{a\to b}(R)$ has an efficient enumeration algorithm, we can enumerate all solutions $I\colon \mathrm{VAR}\,(\varphi) \to D \setminus \{a\}$. By the above, this means that we can enumerate all partial solutions $I\colon \mathrm{VAR}\,(\varphi) \to \{D_1, \ldots, D_k\}$. Since R is $\{a,b\}$-Schaefer, by Lemma 3.15, R is $\{D_1, \ldots, D_k\} \to D$-enumerable. From Theorem 3.11, it follows that R has a polynomial delay enumeration algorithm. □

3.5 Partial Enumeratability: The Brute Force approach

We previously demonstrated one way of how the $E \to D$-enumerability property can be used to get an enumeration algorithm: in the case of Theorem 3.11, we have two "nested" problems, both of which are enumerable. Another method to use this property is when we have exponentially many solutions of one type available, buying us enough time to use a brute force search on the other component of the problem. Theorem 3.18 states this formally, but first we give an example fullfilling this condition.

Example 3.17. Let $R := \{1,2\}^4 \cup \{(0,0,0,2), (0,0,2,0), (0,2,0,0), (0,0,0,1), (0,0,1,0), (0,1,0,0)\}$. R is efficiently enumerable and R^+ is not tractable.

Proof Sketch. It is obvious that $\mathsf{CSP}(R^+)$ is NP-complete, since $\mathsf{CSP}(\text{1-in-3})$ reduces to $\mathsf{CSP}(\Gamma^+)$, by forcing a new variable to 0. Therefore, Theorem 2.4 cannot be used to prove that this relation has an efficient enumeration algorithm. It can be verified that there is no non-trivial partition E of the domain $D = \{0,1,2\}$ such that R is efficiently E-enumerable and efficiently $E \to D$-enumerable (for the latter, remember that testing this simply means testing if the language $\Gamma_R^{E\to D}$ is Schaefer), and thus, Theorem 3.11 does not apply either. The existence of an efficient enumeration algorithm for R follows from the next theorem. The following Lemma 3.20 gives evidence that the rather artificial situation in the example above, that we have the full relation $\{1,2\}^4$ as a subset of the relation, is necessary for this approach. □

Theorem 3.18. *Let E be a partition of the domain D, and let $D_1 \in E$, where $|D_1| \geq |E|$. Let $R \subseteq D^n$ such that $D_1^n \subseteq R$ and R is efficiently $E \rightarrow D$-enumerable. Then R has an efficient enumeration algorithm.*

Proof. Let $D = \{0, \ldots, l-1\}$, let $D_1 := \left\{\alpha_1, \ldots, \alpha_{|D_1|}\right\}$, and let $E = \{D_1, \ldots, D_k\}$. Let φ be an R-formula. For a function $f\colon \mathrm{VAR}(\varphi) \rightarrow \{1, \ldots, k\}$, define $I^f(x) := D_{f(x)}$, and define $J^f(x) := \alpha_{f(x)}$. By definition, the set of all partial E-assignments is equal to the set $\left\{I^f \mid f\colon \mathrm{VAR}(\varphi) \rightarrow \{1, \ldots, k\}\right\}$, and for each $f\colon \mathrm{VAR}(\varphi) \rightarrow \{1, \ldots, k\}$, the assignment J^f is a solution of φ. Furthermore, for $f_1 \neq f_2$, it holds that $J^{f_1} \neq J^{f_2}$, and $I^{f_1} \neq I^{f_2}$.

```
for all $f\colon \mathrm{VAR}(\varphi) \rightarrow \{1, \ldots, k\}$ do
    Unless $f \equiv 1$, enumerate all solutions compatible with $I^f$
    Print $J^f$
end for
Enumerate all assignments $J\colon \mathrm{VAR}(\varphi) \rightarrow D_1$ which contain some value not in $\{\alpha_1, \ldots, \alpha_k\}$.
```

Fig. 1. Enumeration algorithm for the proof of Theorem 3.18

We claim that the algorithm presented in Fig. 1 is an efficient enumeration algorithm for R. It is obvious that this algorithm works with polynomial delay, since R is $E \rightarrow D$-enumerable. Every solution J for φ which not only assigns values from D_1 is printed, because there is a partial solution I^f such that J is compatible with I^f (and f is not the constant 1). Every solution which only assigns values from D_1 is printed as well: the solutions involving only the values $\alpha_1, \ldots, \alpha_k$ are printed in the **FOR** loop of the algorithm, the others are printed in the last statement (it is obvious that these can be enumerated with polynomial delay).

No solution J is printed twice: if J assigns values different from those in D_1, then this cannot happen, since for $f_1 \neq f_2$, the set of solutions compatible with I^{f_1} and I^{f_2} are disjoint. It is also obvious that every solution assigning only values from D_1 is only printed once. □

For the three-element domain, the interesting case here is the following:

Corollary 3.19. *Let R be an n-ary relation over $\{a, b, c\}$, such that $\{a, b\}^n \subseteq R$ and R is $\{a, b\}$-Schaefer. Then there exists a polynomial delay algorithm for R*

The following lemma leads to the conjecture that the condition in Theorem 3.18 requiring the full relation D_1^n to be a subset of R cannot be weakened: the approach in Theorem 3.18 is that because we have an exponentially large "guaranteed" set of solutions, we have enough time for a complete search on the partial assignments. We now show that for a Boolean relation, as soon as it is not the full relation, we cannot expect to have enough solutions to buy us the required time. We expect a similar proposition to hold for the non-Boolean case, and therefore believe that as soon as the condition $D_1^n \subseteq R$ does not hold, the approach from Theorem 3.18 cannot be used.

Lemma 3.20. *Let $R \subsetneqq \{0, 1\}^n$ be an n-ary Boolean relation, then for any $m \in \mathbb{N}$ there exists a CSP($\{R\}$)-formula φ with $|\varphi| \geq m$ such that φ has at most 2 solutions.*

Proof. Since $R \subsetneqq \{0, 1\}^n$ there is a $\mathbf{v} = (v_1, \ldots, v_n) \in \{0, 1\}^n$ such that $\mathbf{v} \notin R$. Without loss of generality let $v_1, \ldots, v_k = 0$ and $v_{k+1}, \ldots, v_n = 1$ with $1 \leq k \leq n$. Let

$$\varphi(x_0, \ldots, x_m) = R(\mathbf{x_1}) \wedge \cdots \wedge R(\mathbf{x_m}) \wedge R(\mathbf{y_1}) \wedge \cdots \wedge R(\mathbf{y_m}),$$

where

$$\mathbf{x_i}[j] \equiv \begin{cases} x_{i-1} & \text{if } v_j = 0 \\ x_i & \text{if } v_j = 1 \end{cases} \quad \text{and} \quad \mathbf{y_i}[j] \equiv \begin{cases} x_i & \text{if } v_j = 0 \\ x_{i-1} & \text{if } v_j = 1 \end{cases}.$$

We prove that all solutions of φ are constant. Assume $I : \{x_0, \ldots, x_m\} \rightarrow \{0, 1\}$ is a non constant solution of φ. Then there exists an $i \in \{1, \ldots, m\}$ such that $I(x_{i-1}) \neq I(x_i)$. That means either $I(\mathbf{x_i}) = \mathbf{v}$ or $I(\mathbf{y_i}) = \mathbf{v}$, but that is a contradiction because $\mathbf{v} \notin R$. Note that this covers also the case where $k = 1$ or $k = n$. So every solution of φ is constant and since $|\varphi| \geq m$ the lemma is proven. □

Partial Enumeratability: Arbitrary nesting In the previous sections, we showed that there are different possible algorithms which can be applied to solve a given enumeration problem, and that these algorithms can be nested to solve enumeration problems over larger domains. We now show a Lemma which gives evidence that arbitrary combinations of these nestings can be done.

Definition 3.21. *Let $R_1 \subseteq D_1^n$ and $R_2 \subseteq D_2^n$ be relations. We say R_1 is isomorphic to R_2, $R_1 \cong R_2$, if there is a bijection $f\colon D_1 \to D_2$ such that for all $(\alpha_1, \dots, \alpha_n) \in D_1^n$,*

$$(\alpha_1, \dots, \alpha_n) \in R_1 \Leftrightarrow (f(\alpha_1), \dots, f(\alpha_n)) \in R_2,$$

i.e. if $f(R_1) = R_2$.

Lemma 3.22. *Let $R_1 \subseteq D_1^n$, $R_2 \subseteq D_2^m$, and let there be some $\alpha \in D_1$, and some $\mathbf{v} \in R_1, \mathbf{v} = (\alpha_1, \dots, \alpha_n)$, such that $|\{i \in \{1, \dots, n\} \mid \alpha_i = \alpha\}| = m$, and such that $R_2 \neq \emptyset$. Then there exists a relation $R \subseteq (D_1 \cup D_2)^n$, and a partition E of $D_1 \cup D_2$, such that the following holds:*

- *$R/E \cong R_1$,*
- *There is some $S \in \Gamma_E^R$ such that $S \cong R_2$.*

Further, the following connections hold:

- *R is efficiently $E \to D$-enumerable if and only if R_2 is efficiently enumerable.*
- *If R_2 has a constant polymorphism, then R is efficiently E-enumerable if and only if R_1 is efficiently enumerable.*

Proof. Let $D_1 = \{\alpha_1, \dots, \alpha_k\}$, and let $D_2 = \{\beta_1, \dots, \beta_l\}$. Assume that $D_1 \cap D_2 = \emptyset$. Without loss of generality, assume that if R_2 has a constant polymorphism, then R_2 contains the tuple $(\beta_l, \dots, \beta_l)$. Further, assume that the α from the prerequisites is α_m. We now construct a new domain $D_2' = (D_2 \setminus \{\beta_1\}) \cup \{\alpha_m\}$, and exchange, in every component of every tuple of R_2, the element β_l with the element α_m. Since we are only interested in the relations up to isomorphism, we call this domain D_2 again. In short, we can assume, without loss of generality, the following:

- $D_1 = \{\alpha_1, \dots, \alpha_k\}$, $D_2 = \{\beta_1, \dots, \beta_l\}$, where $D_1 \cap D_2 = \{\alpha_k\} = \{\beta_l\}$.
- If R_2 has a constant polymorphism, then $(\beta_l, \dots, \beta_l) \in R_2$
- There is some $\mathbf{v} \in R_1$, such that $\mathbf{v} = (\underbrace{\alpha_k, \dots, \alpha_k}_{m}, \gamma_{m+1}, \dots, \gamma_n)$ and $\gamma_{m+1}, \dots, \gamma_n \in D_1 \setminus \{\alpha_k\}$.

We construct a partition E on $D_1 \cup D_2$, as follows: $E := \{\{\alpha_1\}, \dots, \{\alpha_{k-1}\}, D_2\}$, i.e. all elements in D_2 are equivalent, and all elements from $D_1 \setminus D_2$ form one-element equivalence classes.

Let f^E be the canonical homomorphism assigning each element in $D_1 \cup D_2$ its equivalence class in E. We now define relations as follows:

$R_1' := f^E(R_1) \setminus \{f^E\mathbf{v}\}$,
$R_2' := \{(\delta_1, \dots, \delta_m, \gamma_{m+1}, \dots, \gamma_n) \mid (\delta_1, \dots, \delta_m) \in R_2\}$,
$R := f^{E^{-1}}(R_1') \cup R_2'$,

where $f^{E^{-1}}(R_1') = \{\mathbf{u} \in D_1^n \mid f^E(\mathbf{u}) \in R_1'\}$.

Observe that $f^{E^{-1}}(R_1') = \{\mathbf{w} \in (D_1 \cup D_2)^n \mid (f^E(\mathbf{w}) \in f^E(R_1)) \wedge (f^E(\mathbf{w}) \neq f^E(\mathbf{v}))\}$, and that for any $\mathbf{w} \in R_2'$, it holds that $f^E(\mathbf{w}) = \mathbf{v}$. Therefore, it follows for the relation R:

$$\begin{aligned} R = &\{\mathbf{w} \in (D_1 \cup D_2)^n \mid (f^E(\mathbf{w}) \neq f^E(\mathbf{v})) \wedge (f^E(\mathbf{w}) \in f^E(R_1))\} \\ \cup &\{\mathbf{w} \in (D_1 \cup D_2)^n \mid (f^E(\mathbf{w}) = f^E(\mathbf{v})) \wedge ((\mathbf{w}[1], \dots, \mathbf{w}[m]) \in R_2)\} \end{aligned}$$

Let f be the restriction of f^E to D_1. Since in every equivalence from E, exactly one element from D_1 appears, $f\colon D_1 \to E$ is a bijection. We show that $f(R_1) = R/E$, proving that $R_1 \cong R/E$. To prove this, we show that for $\delta_1, \dots, \delta_n \in D_1$, it holds that

$$(\delta_1, \dots, \delta_n) \in R_1 \iff f^E(\delta_1, \dots, \delta_n) \in f^E(R).$$

First, let $(\delta_1, \dots, \delta_n) \in R_1$. If $(\delta_1, \dots, \delta_n) \neq \mathbf{v}$, then $f^E(\delta_1, \dots, \delta_n) \in R_1'$, therefore, $(\delta_1, \dots, \delta_n) \in f^{E^{-1}}(R_1') \subseteq R$. Therefore, assume that $(\delta_1, \dots, \delta_n) = \mathbf{v}$. In particular, this implies $f^E(\delta_1, \dots, \delta_n) = f^E(\mathbf{v})$. Since $R_2 \neq \emptyset$, we know that $R_2' \neq \emptyset$. Let $\mathbf{u} \in R_2'$. From the above it follows that $f^E(\mathbf{u}) = f^E(\mathbf{v}) = f^E(\delta_1, \ldots, \delta_n)$. Since $R'_2 \subseteq R$, it follows that $\mathbf{u} \in R$, and since $f^E(\delta_1, \ldots, \delta_n) = f^E(\mathbf{u})$, we know that $f^E(\delta_1, \ldots, \delta_n) \in f^E(R)$.

For the converse, let $f^E(\delta_1, \ldots, \delta_n) \in f^E(R)$. If $(\delta_1, \ldots, \delta_n) = \mathbf{v}$, then, by choice of $\mathbf{v}$, $(\delta_1, \ldots, \delta_n) \in R_1$. Now, assume that $(\delta_1, \ldots, \delta_n) \neq \mathbf{v}$. Since f^E restricted to D_1 is injective, and $\delta_1, \ldots, \delta_n \in D_1$, it follows that $f^E(\delta_1, \ldots, \delta_n) \neq f^E(\mathbf{v})$. Since for all $\mathbf{w} \in R'_2$, $f^E(\mathbf{w}) = f^E(\mathbf{v})$, we know that $f^E(\delta_1, \ldots, \delta_n) \neq f^E(R'_2)$. Therefore, by definition of R, it follows that $f^E(\delta_1, \ldots, \delta_n) \in f^E(f^{E^{-1}})(R'_1) = R'_1 \subseteq f^E(R_1)$. Since f^E restricted to D_1 is injective, it follows that $(\delta_1, \ldots, \delta_n) \in R_1$.

We now study the constraint language Γ_R^E, i.e. we look at the relations $R_\mathbf{u}^E$ for each $\mathbf{u} \in E^n$. We make a case distinction:

Case 1: $\mathbf{u} = f^E(\mathbf{v})$. Observe that D_2 is the only equivalence class in E containing more than one element. Therefore, by definition, it holds that

$$\begin{aligned} R^E_{f^E(\mathbf{v})} &= \left\{\mathbf{u} \in R \mid f^E(\mathbf{u}) = f^E(\mathbf{v})\right\}_{\{i \mid \mathbf{v}[i] \in D_2\}} \\ &= \{\mathbf{u} \in R \mid (\mathbf{u}[1], \ldots, \mathbf{u}[m]) \in R_2\}_{\{1,\ldots,m\}} \\ &= R_2. \end{aligned}$$

Case 2: $\mathbf{u} \neq f^E(\mathbf{v})$, and there is no $\mathbf{w} \in R$ compatible with $\mathbf{u}$. Then, by definition, $R^E_{f^E(\mathbf{v})} = \emptyset$.

Case 3: $\mathbf{u} \neq f^E(\mathbf{v})$, and there is some $\mathbf{w} \in R$ compatible with $\mathbf{u}$. Since $\mathbf{w}$ is compatible with $\mathbf{u}$, we know that $f^E(\mathbf{w}) = \mathbf{u}$. Since $\mathbf{u} \neq f^E(\mathbf{v})$, we know that $f^E(\mathbf{w}) \neq f^E(\mathbf{v})$. Thus, since $f^E(\mathbf{s}) = f^E(\mathbf{v})$ for all $\mathbf{s} \in R'_2$, we know that $\mathbf{w} \notin R'_2$. Therefore, by definition of R, $\mathbf{w} \in f^{E^{-1}}(R'_1)$.
We claim that $R^E_\mathbf{u}$ is the full relation over D_2 for some arity. By definition, this is the case if for all $\mathbf{s} \in (D_1 \cup D_2)^n$, if $f^E(\mathbf{s}) = f^E(\mathbf{w})$, then $\mathbf{s} \in R$. This is certainly the case, since $\mathbf{w} \in f^{E^{-1}}(R'_1)$, and therefore, $f^{E^{-1}}(\mathbf{w}) \subseteq R$.

We conclude that the constraint language Γ_R^E only contains the relation R_2, the empty relations and full relations over D_2. Any formula over this language in which a $\emptyset$-clause appears is immediately unsatisfiable. In any other formula over this language, every clause involving a full relation of some arity can be removed without changing the set of solutions of the formula. Therefore, Γ_R^E has an efficient enumeration algorithm if and only if there is an efficient enumeration algorithm for R_2.

Now for the last equivalence, let R_2 have a constant polymorphism. As argued above, we can assume that $(\alpha_k, \ldots, \alpha_k) \in R_2$. We claim that D_1 is a representation system for E which is compatible with R. Following Lemma 3.13, this implies that R is efficiently E-enumerable if and only if R/E is efficiently enumerable. Since, by our construction, $R_1 \cong R/E$, this proves the lemma. By definition, if $f^{D_1} \colon D \rightarrow D_1$ defined as

$$f^{D_1}(\alpha) = \begin{cases} \alpha, & \alpha \in D_1 \\ \alpha_k, & \alpha \in D_2 \end{cases}$$

is a polymorphism of R, then D_1 is a representation system for E compatible with R. Therefore, let $\mathbf{u} \in R$. We show that $f^{D_1}(\mathbf{u}) \in R = f^{E^{-1}}(R'_1) \cup R'_2$. If $\mathbf{u} \in f^{E^{-1}}(R'_1)$, then $f^E(\mathbf{u}) \in R'_1$. Obviously, it holds that $f^E(\mathbf{u}) = f^E(f^{D_1}(\mathbf{u})) \in R'_1$, since f^{D_1} assigns each value α of the domain a value $f^{D_1}(\alpha)$ which is E-equivalent to α. Therefore, $f^{D_1}(\mathbf{u}) \in f^{E^{-1}}(R'_1)$. If $\mathbf{u} \in R'_2$, then, by definition of R'_2 and f^{D_1}, we know that $f^{D_1}(\mathbf{u}) = (\alpha_k, \ldots, \alpha_k, \gamma_{m+1}, \ldots, \gamma_n) \in R'_2$, since $(\alpha_k, \ldots, \alpha_k) \in R_2$. Therefore, f^{D_1} is a polymorphism of R, and the lemma is proven. □

From this Lemma, it follows that we can have arbitrary "nestings" of algorithms to solve a given enumeration problem. We can, by inductively applying the construction used in the proof for Lemma 3.22, construct a relation R on a domain D with partitions $E_1, \ldots, E_l$, such that for each partial enumeration step $E_i \rightarrow E_{i+1}$, an arbitrary type of enumeration algorithm applies.

In other words, this Lemma proves that the methods to enumerate the distinct steps in a chain of partial enumeration algorithms are, with some technical requirements, completely independent of each other: arbitrary combinations exist.

3.6 Partitioned Relations

We now show tractability conditions for special kinds of relations. These are of interest to us, since relations appearing in constraint languages $\Gamma_R^{E_1 \rightarrow E_2}$ are of this form.

Definition 3.23. *Let $R \subseteq D^n$, and let $E = \{D_1, \dots, D_k\}$ be a partition of D. We say that R is* E-partitioned *if $R \subseteq D_{i_1} \times \dots \times D_{i_n}$ for $i_1, \dots, i_n \in \{1, \dots, k\}$.*

Note that if $R_1, \dots, R_n$ are E-partitioned relations, then $R_1 \times \dots \times R_n$ is E-partitioned as well. Therefore, to study enumeration algorithms for constraint languages containing only E-partitioned relations, Lemma 3.1 shows that we can again restrict ourselves to one-element constraint relations in these cases. The following Lemma can be used to show that some constraint language over a large domain has an efficient enumeration algorithm, using knowledge about a constraint language over a smaller domain.

Lemma 3.24. *Let $R \subseteq D^n$, E a partition of D such that R is E-partitioned. Let $E = \{D_1, \dots, D_k\}$, and let $|D_i| \leq m$ for all i. Let $D_i = \left\{\alpha_0^i, \dots, \alpha_{d_i}^i\right\}$.*

Define a function $f \colon D \to \{0, \dots, m-1\}$ as $f(\alpha_j^i) = j$. If $f(R)$ has an efficient enumeration algorithm, then R has an efficient enumeration algorithm.

Proof. Let $R \subseteq D_{i_1} \times \dots \times D_{i_n}$, and let φ be an R-formula,

$$\varphi = R(x_1^1, \dots, x_n^1) \wedge \dots \wedge R(x_1^l, \dots, x_n^l).$$

First, we check if there is some variable $x \in \mathrm{VAR}\,(\varphi)$ such that there are i_1, j_1, i_2, j_2 with $x_{j_1}^{i_1} = x_{j_2}^{i_2} = x$, and $D_{j_1} \neq D_{j_2}$. If this is the case, then φ is not satisfiable, because D_{j_1} and D_{j_2} are disjoint, and x is forced to be assigned a value from $D_{j_1} \cap D_{j_2}$. This condition can be tested in polynomial time. Therefore, without loss of generality, assume that this does not happen, and that there is a function $d \colon \mathrm{VAR}\,(\varphi) \to E$ assigning each variable from $\mathrm{VAR}\,(\varphi)$ its domain in such a way that for every solution $I \models \varphi$, $I(x) \in d(x)$. Note that this function can be computed in polynomial time. This is achieved by simply checking at which position of an R-application the variable x appears, and then looking up which equivalence class from E correspponds to the column of R.

We claim that there is a one-to-one correspondence between solutions $I \models \varphi$ and $J \models f(\varphi)$. To be more precise: an assignment $J \colon \mathrm{VAR}\,(\varphi) \to \{0, \dots, m-1\}$ is a solution of $f(\varphi)$ if and only there is a unique $I \colon \mathrm{VAR}\,(\varphi) \to D$ such that $I \models \varphi$, and $f(I) = J$. Since for each $J \models f(\varphi)$, the unique solution $I \models \varphi$ such that $f(I) = J$ can be computed in polynomial time, this shows that when we can enumerate the solutions of $f(\varphi)$, then we can also enumerate the solutions of φ.

First, let $I \models \varphi$ for some assignment $I \colon \mathrm{VAR}\,(\varphi) \to D$. We claim that $f(I) \models f(\varphi)$. Let $R(x_1^i, \dots, x_n^i)$ be a clause in φ. Since $I \models \varphi$, $I(x_1^i, \dots, x_n^i) \in R$. Therefore, $f(I(x_1^i, \dots, x_n^i)) \in f(R)$.

Now, let $J \models f(\varphi)$ for $J \colon \mathrm{VAR}\,(\varphi) \to \{0, \dots, m-1\}$. We define an assignment $I \colon \mathrm{VAR}\,(\varphi) \to D$ such that $f(I) = J$: for a variable x such that $d(x) = D_i \in E$, and $J(x) = j \in \{0, \dots, m-1\}$, define $I(x) = \alpha_j^i$. Then $f(I) = J$. Let $R(x_1^i, \dots, x_n^i)$ be a clause in φ. Since $f(I) \models f(\varphi)$, it follows that $(a_1, \dots, a_n) := f(I(x_1^i, \dots, x_n^i)) \in R$. Since f, restricted to D_i, is injective, it follows that $I(x_1^i, \dots, x_m^i) = (\alpha_{a_1}^{d(x_1^i)}, \dots, \alpha_{a_n}^{d(x_n^i)}) \in R$.

Now assume that for a given $J \models \varphi$, there are two different assignments I_1 and I_2 such that $f(I_1) = f(I_2) = J$, and $I_1, I_2 \models \varphi$. Let $I_1(x_j^i) \neq I_2(x_j^i)$. Since $f(I_1) = f(I_2)$, it follows that $f(I_1(x_j^i)) = f(I_2(x_j^i))$. Since $I_1, I_2 \models \varphi$, it follows that $I_1(x_j^i), I_2(x_j^i) \in d(x_j^i)$. Since the function f restricted to $d(x_j^i)$ is injective, it follows that $f(I_1(x_j^i)) \neq f(I_2(x_j^i))$. This is a contradiction. □

Note that the converse of the above Lemma probably does not hold, i.e. negative results do not translate to bigger domains. Assume that $R \subseteq D_1 \times D_2$, where $D_1 \cap D_2 = \emptyset$, $|D_1| = |D_2|$. Then $R(x, x)$ is not satisfiable, but $f(R)(x, x)$ may be satisfiable. So there is no canonical correspondence which allows us to get the solutions to an $f(R)$-formula from the solutions to the corresponding R-formula.

3.7 Partial solutions and lexicographical orderings

We generalize the notion of lexicographical orderings in two ways. A lexicographical ordering on the solutions of some formula φ uses the same order of the domain D for each variable. But in a more general setting, we might want to demand that for the variable x_1, the value $a \in D$ is smaller than b, but for x_2, we might want $b < a$. Further, we generalize lexicographical orderings to partial solutions in the obvious way. Let $<_{\mathrm{var}}$ be a linear order on $\mathrm{VAR}\,(\varphi)$, and for every

$x \in \mathrm{VAR}(\varphi)$, let $<_x$ be a linear order on a partition E on D. We define a linear order $<_\varphi$ on the partial E-assignments of φ as follows:

$I_1 <_\varphi I_2$ if there is some $x \in \mathrm{VAR}(\varphi)$ such that for all $y \in \mathrm{VAR}(\varphi)$ with $y <_{\mathrm{var}} x$, it holds that $I_1(y) = I_2(y)$, and $I_1(x) <_x I_2(x)$. Such an order is called a *variable E-lexicographical order* on $\mathrm{SOL}^E(\varphi)$.

A *variable E-lexicographical enumeration algorithm for Γ* is an algorithm which, when given a Γ-formula φ and a variable lexicographical order $<_\varphi$ as defined above as input, prints each partial E-solution of φ exactly once and in the order defined by $<_\varphi$. We say that a constraint language Γ over the domain D has an efficient variable lexicographical enumeration algorithm if there is an efficient variable lexicographical D^{disc} enumeration algorithm for Γ, i.e. if we can efficiently enumerate the "real" (not partial) solutions of Γ-formulas in a variable lexicographical order. (We identify partial solutions where each equivalence class has only one element and "real" solutions here.)

This definition is the formalization of the requirement that we can tell the algorithm in which order we want the (partial) solutions to be printed. Note that in this context, the notion of semi-efficient enumeration algorithms does not make sense: if we allow such an algorithm to print every solution a constant number of times, then the algorithm would have to print all of these copies directly after each other, due to the lexicographical ordering condition. In this case, polynomial delay is still maintained if such an algorithm would be modified to print each solution only once (and in order to achieve this, it would only have to store the most recently printed solution).

Theorem 3.25. *Let E be a partition on D, and let Γ be a constraint language over D. There exists a variable E-lexicographical enumeration algorithm for Γ if and only if $\Gamma \cup E$ is tractable.*

Proof. Let $E = \{D_1, \dots, D_k\}$, and first assume that $\Gamma \cup E$ is tractable. Let φ be a Γ-formula, and let $<_\varphi$ be a variable lexicographical order. The procedure Generate (Fig. 2) generates, on input φ and a $\{D_1, \dots, D_k\}$-formula ψ such that $\mathrm{VAR}(\psi) \subseteq \mathrm{VAR}(\varphi)$, all partial solutions of $\varphi \wedge \psi$ in the order defined by $<_\varphi$ with polynomial delay. To enumerate all partial solutions of φ, start Generate with φ and the empty formula ψ. This algorithm is a generalization of the algorithms in the proof of Theorem 9 in [Coh04] and in section 3 of [CH97].

```
Input: Formula $\varphi$ and conjunction $\psi$ of constraints of the form $D_i(x_j)$, where $\mathrm{VAR}(\psi) \subseteq \mathrm{VAR}(\varphi)$
  if $\varphi \wedge \psi \notin$ SAT then
    exit
  end if
  if $\mathrm{VAR}(\varphi) = \mathrm{VAR}(\psi)$ then
    Print the partial assignment defined by $\psi$
    exit
  end if
  Let $x$ be the $<_{\mathrm{var}}$-smallest variable in $\mathrm{VAR}(\varphi) \setminus \mathrm{VAR}(\psi)$
  Let $D_{i_1} <_x D_{i_2} <_x \cdots <_x D_{i_k}$
  for $j = 1$ to $j = k$ do
    Generate($\varphi, \psi \wedge D_{i_j}(x)$)
  end for
```

Fig. 2. The procedure Generate

The algorithm works in polynomial time, since the satisfiability test used in the algorithm can be performed in polynomial time by our prerequisites.

Now assume that Γ has a variable E-lexicographical enumeration algorithm, and let $\varphi \wedge \psi$ be a $\Gamma \cup E$-formula. Without loss of generality, assume that the (not necessarily distinct) variables $x_1^i, \dots, x_l^i$ are constrained to the subset D_i, i.e. there are clauses $D_i(x_1^i), \dots, D_i(x_l^i)$ for all i, and no other applications of D_i. Let $X := \{x_1^1, \dots, x_l^1, x_1^2, \dots, x_l^k\}$ be the set of variables which appear in one of the D_i constraint applications, and let $Y := \mathrm{VAR}(\varphi) \setminus X$. Now, define $<_{\mathrm{var}}$ to be a linear order on the variables such that for all $x \in X, y \in Y$, it holds that $x <_{\mathrm{var}} y$. For each $x_j^i \in X$, define the order $<_x$ in such a way that D_i is the smallest element of $\{D_1, \dots, D_k\}$ with respect to $<_x$. Let φ' be the Γ-formula obtained from φ by deleting all D_i-clauses. Now, enumerate the solutions of φ' according to the order as defined above. By construction of the order, the formula φ is satisfiable if and only if if the first partial solution returned by the algorithm satisfies the clauses $D_i(x_1^i), \dots, D_i(x_k^i)$. This gives a polynomial-time decision procedure for $\mathsf{CSP}(\Gamma \cup \{D_1, \dots, D_k\})$. □

This theorem generalizes the "extreme" cases: Γ^+ tractable is equivalent to $\Gamma \cup D^{\mathrm{disc}}$ tractable. For this case, the theorem implies that the solutions of some Γ-formula can be printed in variable lexicographical order if and only if Γ^+ is tractable. On the other hand, $\Gamma \cup \{D\}$ tractable is equivalent to Γ tractable. Therefore, the notion of tractability combined with partitions of the domain is a natural generalization of previous cases.

For the Boolean case, all of our various enumeration cases are equivalent: For the non-Boolean case, a similar proposition does not hold, unless $\mathrm{P} = \mathrm{NP}$, as illustrated by Example 3.8.

Proposition 3.26. *Let R be a relation over the Boolean domain. Then the following propositions are equivalent:*

1. *R has a semi-efficient enumeration algorithm,*
2. *R has an efficient enumeration algorithm,*
3. *R has an efficient variable lexicographical order enumeration algorithm,*
4. *R^+ is tractable,*
5. *R is Schaefer, or $\mathrm{P} = \mathrm{NP}$.*

Proof. The implications $4 \rightarrow 3$, $3 \rightarrow 2$ and $2 \rightarrow 1$ are clear, and all hold for the general case as well. The only one of these which does not follow directly from the definition is $4 \rightarrow 3$, this follows from Theorem 3.25. For the Boolean case and the implication $1 \rightarrow 4$, observe that if Γ is Schaefer, then Γ^+ is tractable. Otherwise, it cannot have a semi-efficient enumeration algorithm, because $\mathsf{CSP}^*(\Gamma)$ is NP-complete, and a semi-efficient enumeration algorithm immediately gives a polynomial-time decision procedure for this problem. The equivalence of 2 and 5 is the main result from [CH97]. □

4 Negative Cases

In this section we present negative results, i.e. conditions ensuring that relations do not have efficient enumeration algorithms (if $\mathrm{P} \neq \mathrm{NP}$ holds). We present a number of technical results. With current techniques, we were not yet able to show broad conditions for non-enumerability of relations. Note that due to Proposition 2.9, in the case where $|D| = 3$, we always have either a constant polymorphism, or a polymorphism of the form $f_{a \rightarrow b}$. The lemma deals with the latter case, with additional prerequisites.

We now show our first non-enumerability result:

Lemma 4.1. *Let D be a 3-element domain, and $R \subseteq D^n$, let $a, b \in D$, such that $f_{a \rightarrow b} \in \mathrm{Pol}(R)$, $f_{a \rightarrow b}(R)$ is not Schaefer, and one of the following cases occurs:*

1. *$(c, \dots, c) \in R$ and there is some $\mathbf{v} \in \{a, b\}^n$ with $\mathbf{v} \notin R$, $\mathbf{v}[a/c, b/b] \in R$, and $\mathbf{v}[a/b, b/c] \in R$.*
2. *$(b, \dots, b) \notin R$*
3. *$(a, \dots, a) \notin R$ and $(c, \dots, c) \in R$*
4. *R can express the "$\neq a$" condition (Conjecture[1])*
5. *$(a, \dots, a), (c, \dots, c) \notin R$, and there is a $\mathbf{v} \in \{a, b\}^n \setminus R$ with $f_{a \rightarrow c}(\mathbf{v}) \in R$*

Then R does not have a semi-efficient enumeration algorithm, unless $\mathrm{P} = \mathrm{NP}$.

Proof. Since $f_{a \rightarrow b}(R)$ is not Schaefer, the problem $\mathsf{CSP}^*(f_{a \rightarrow b}(R))$ is NP-complete, where $\mathsf{CSP}^*(f_{a \rightarrow b}(R))$ is the problem to decide, given a $f_{a \rightarrow b}(R)$-formula, if it has a non-constant solution [CH97]. Let φ be a $f_{a \rightarrow b}(R)$-formula, and let $\mathrm{VAR}(\varphi) = \{x_1, \dots, x_m\}$. We construct an R-formula

$$\psi := \varphi[f_{a \rightarrow b}(R)/R] \wedge \chi,$$

where χ is an R-formula and χ has the following properties:

[1] The proof for this is unclear.

1. $|\{I\colon \mathrm{VAR}\,(\chi) \to \{a,b\}, I \models \chi\}|$ is polynomial in $|\varphi|$
2. For all $I\colon \mathrm{VAR}\,(\varphi) \to \{b,c\}$, $I \models \varphi$ implies $I \models \chi$.

The first condition ensures that when enumerating the solutions of $\varphi \wedge \chi$, we do not have to wait too long until we get a solution which does not only contain as and bs. A solution which contains at least one c maps to a non-constant solution of φ. The second condition ensures that the formula χ does not interfere with the solutions of φ, which we are interested in. Hence, the existence of an efficient enumeration algorithm for R implies that $\mathsf{CSP}^*(f_{a\to b}(R))$ is decidable in polynomial time.

We now show that in each of the cases stated in the lemma, we can construct a formula χ which has this properties.

Case 1 Let
$$\chi(x_2,\ldots,x_m) = R(\mathbf{x_2}) \wedge \cdots \wedge R(\mathbf{x_m}) \wedge R(\mathbf{y_2}) \wedge \cdots \wedge R(\mathbf{y_m}),$$
where
$$\mathbf{x_i}[j] \equiv \begin{cases} x_{i-1} & \text{if } \mathbf{v}[j] = a \\ x_i & \text{if } \mathbf{v}[j] = b \end{cases} \quad \text{and} \quad \mathbf{y_i}[j] \equiv \begin{cases} x_i & \text{if } \mathbf{v}[j] = a \\ x_{i-1} & \text{if } \mathbf{v}[j] = b \end{cases}.$$
We prove that all solutions $I : \{x_1,\ldots,x_m\} \to \{a,b\}$ of χ are constant. Assume $I : \{x_0,\ldots,x_m\} \to \{a,b\}$ is a non constant solution of φ. Then there exists an $i \in \{1,\ldots,m\}$ such that $I(x_{i-1}) \neq I(x_i)$. That means either $I(\mathbf{x_i}) = \mathbf{v}$ or $I(\mathbf{y_i}) = \mathbf{v}$, but that is a contradiction because $\mathbf{v} \notin R$. Note that this covers also the case where $k = 1$ or $k = n$. So the first property is satisfied.
For the second property we assume without loss of generality that $(b,\ldots,b) \in R$, otherwise we can use the next case. Let $I : \mathrm{VAR}\,(\varphi) \to \{b,c\}$ be a solution of φ. Then $I(\mathbf{x_i}), I(\mathbf{y_i}) \in \{\mathbf{v}[a/c,b/b], \mathbf{v}[a/b,b/c], (b,\ldots,b), (c,\ldots,c)\}$ which is a subset of R due to our prerequisites. That means $I \vdash R$.

Case 2 Since $f_{a\to b}$ is a polymorphism of R, there is no tuple $\mathbf{v} \in R \cap \{a,b\}^n$. Therefore we can choose $\chi = \emptyset$.

Case 3 Choose $\chi = R(x_1,\ldots,x_1) \wedge R(x_2,\ldots,x_2) \wedge \cdots \wedge R(x_m,\ldots,x_m)$. The formula χ does not have any solution in which an a occurs (note that we can assume $(b,\ldots,b) \in R$, and therefore these clauses are equivalent to restricting all variables to take either the value b or c).

Case 5 In this case, we can force some additional variable t to take the value b by the constraint application $R(t,\ldots,t)$ (without loss of generality, we can assume $(b,\ldots,b) \in R$, because of case 2). Now assume that $v = (\underbrace{a,\ldots,a}_{k},\underbrace{b,\ldots,b}_{l})$, and add the constraint application $R(\underbrace{x_i,\ldots,x_i}_{k},\underbrace{t,\ldots,t}_{l})$ for each variable x_i. This forces the variables to take values different from a (note that $\mathbf{v}$ cannot be the constant $(a,\ldots,a)$ tuple, since $(c,\ldots,c) \notin R$).

□

Like most of our negative results, the following Lemma is a very special case. However, this is important since it allows us to show, in Theorem 5.1, that the Galois connection does not hold for the enumeration problem.

Lemma 4.2. *Let R be a relation over the domain $D = \{a,b,c\}$, let R not be $\{a,b\}$-Schaefer, and let $\{(c)\} \in \langle R\rangle$, $\{(a),(b)\} \in \langle R,\{(c)\}\rangle_{\nexists}$, and assume* $\mathrm{P} \neq \mathrm{NP}$. *Then R does not have a semi-efficient enumeration algorithm.*

Proof. let $E := \{\{a,b\},\{c\}\}$. Since R is not $\{a,b\}$-Schaefer, the constraint language $\Gamma_R^{E\to D}$ is not Schaefer (note that $\Gamma_R^{E\to D}$ is a constraint language over the Boolean domain $\{a,b\}$). Therefore, following [CH97], the problem $\mathsf{CSP}^*(\Gamma_R^{E\to D})$ is NP-complete, where $\mathsf{CSP}^*(\Gamma_R^{E\to D})$ is the problem, to determine for a $\Gamma_R^{E\to D}$-formula ψ, if ψ has a non-constant solution. Now assume that R' has a semi-enumeration algorithm, and let ψ be a $\Gamma_R^{E\to D}$-formula,
$$\psi = \bigwedge_{i=1}^{l} R^E_{\mathbf{v_{t_i}}}(x^i_1,\ldots,x^i_{\mathrm{ar}(R^E_{\mathbf{v_{t_i}}})}),$$

where $\mathbf{v_{t_i}} \in E^n$ for all $1 \leq i \leq l$. We construct a formula φ of R-clauses which is "equivalent" to ψ in some sense. This construction is very similar to the one in the proof for Lemma 3.15. Since ψ is a formula in which only relations over the Boolean domain $\{a, b\}$ appear, we first force all appearing variables in ψ to take values from this domain: let

$$\varphi_1 := \exists y_1, \ldots, y_k \bigwedge_{x \in \mathrm{VAR}(\psi)} (x \in \{a, b\}) \wedge (x_c = c).$$

A formula equivalent to φ_1 can be expressed as R-formula, since $\{c\} \in \langle R \rangle$, and $\{(a), (b)\} \in \langle R, \{(c)\} \rangle_{\nexists}$. Note that the number k of new existentially quantified variables does not depend on the length of φ: these are needed to express the clause $x_c = c$, since $\{(c)\}$ is not necessarily in $\langle R \rangle_{\nexists}$. This construction can result in additional existential variables. To further express the clauses $x \in \{a, b\}$ does not require any existential variables, since $\{(a), (b)\} \in \langle R, \{(c)\} \rangle_{\nexists}$.

Now, let $C = R_v^E(x_1, \ldots, x_m)$ be a clause in ψ, where $x_1, \ldots, x_m \in \mathrm{VAR}(\psi)$ and $\mathbf{v} \in E^n$. Without loss of generality, let $\mathbf{v} = (\underbrace{\{c\}, \ldots, \{c\}}_{p}, \underbrace{\{a, b\}, \ldots, \{a, b\}}_{m})$. Let $C' := R(\underbrace{x_c, \ldots, x_c}_{p}, x_1, \ldots, x_m)$. Then the solutions to $C' \wedge \psi_1$ are exactly the solutions to C with the additional assignment $x_c = c$. Now, let φ be the conjunction of φ_1 and C' for all clauses C' in ψ. Then the solutions I for ψ, $I \colon \mathrm{VAR}(\psi) \to \{a, b\}$ are exactly those assignments to $\mathrm{VAR}(\psi)$ which can be extended to assignments I' to $\mathrm{VAR}(\varphi) = \mathrm{VAR}(\psi) \cup \{x_c, y_1, \ldots, y_k\}$ such that $I' \models \varphi$. Note that the formula φ can be computed from ψ in polynomial time.

Assume that R does have a semi-efficient enumeration algorithm. Start this algorithm for φ. Since ψ can have at most 2 constant solutions, and for every solution of ψ, there is only a constant number of solutions to φ, and every solution of φ is printed only a constant number of times, we only have to wait for a constant number of outputs of this enumeration algorithm. Since the algorithm has polynomial delay, this is a polynomial-time decision procedure for $\mathsf{CSP}^*(\Gamma_R^{E \to D})$, which cannot exist due to the prerequisites and Proposition 3.26. $\square$

4.1 Implementation

In this section we show a canonical way to implement unary relations, and derive results using the $\langle . \rangle_{\mathrm{cons}}$ closure operator, which we can apply to the enumeration problem.

Definition 4.3. *Let R be a relation over the domain $D = \{\alpha_1, \ldots, \alpha_k\}$. For $\mathbf{v} \in R$, we define*

$$C_{\mathbf{v}} := R(a_1^{\mathbf{v}}, \ldots, a_n^{\mathbf{v}}),$$

where $a_i^{\mathbf{v}} = x_{\mathbf{v}[i]}$, for distinct variables $x_{\alpha_1}, \ldots, x_{\alpha_k}$. Further, we define

$$\varphi_R = \bigwedge_{\mathbf{v} \in R} C_{\mathbf{v}}.$$

The idea behind this definition is to try to force the variable x_α to take the value $\alpha \in D$ for all satisfying assignments of φ_R. We prove that if we cannot force these values with φ_R, then we cannot do this with applications of R at all.

Lemma 4.4. *1. There is a one-to-one correspondence between solutions $I \colon \{x_{\alpha_1}, \ldots, x_{\alpha_k}\} \to \{\alpha_1, \ldots, \alpha_k\}$ and unary polymorphisms of R. To be precise, $f \colon D \to D$ is a polymorphism of R if and only if I_f is a solution of φ_R, where $I_f(x_\alpha) = f(\alpha)$.*

2. Let R be a relation over D, and let $\alpha \in D$. The following are equivalent:
   (a) $\{(\alpha)\} \in \langle R \rangle$,
   (b) For all $I \models \varphi_R$, it holds that $I(x_\alpha) = \alpha$.

3. Let $D = \{a, b, c\}$ be the three-element domain, let $E = \{\{a, b\}, \{c\}\}$, and let $f_{a \to b}$ be a polymorphism of R, and let all unary polymorphisms of R be conservative on the classes in E. Then $E \subseteq \langle R \rangle_{cons}$.

Proof. 1. Let f be a polymorphism of R. By construction of φ_R, it is obvious that I_{id}, where id is the identity, is a solution of φ. Since $f \in \text{Pol}(R)$, we know that $f(I_{\text{id}})$ is a solution of φ as well (because only R-clauses appear in φ_R). Now for $\alpha \in D$, it holds that $f(I_{\text{id}})(x_\alpha) = f(I_{\text{id}}(x_\alpha)) = f(\alpha)$, i.e. $f(I_{\text{id}}) = I_f \models \varphi_R$.
For the converse, assume that I_f is a solution of φ_R, and let $\mathbf{v} \in R$. We show that $f(\mathbf{v}) \in R$. Since $I_f \models \varphi_R$, it holds that $I_f \models C_{\mathbf{v}}$, i.e. $I_f((a_1^{\mathbf{v}}, \ldots, a_n^{\mathbf{v}})) \in R$. We show that $I_f((a_1^{\mathbf{v}}, \ldots, a_n^{\mathbf{v}})) = f(\mathbf{v})$. It holds that $I_{\text{id}}(a_1^{\mathbf{v}}, \ldots, a_n^{\mathbf{v}}) = \mathbf{v}$, by definition of $C_{\mathbf{v}}$. Obviously, $f(I_{\text{id}})(a_1^{\mathbf{v}}, \ldots, a_n^{\mathbf{v}}) = f(\mathbf{v}) \in R$.

2. The direction 2*b* → 2*a* follows per definition. On the other hand, assume $\{(\alpha)\} \in \langle R \rangle$. By Proposition 2.3, this implies $\text{Pol}(R) \subseteq \text{Pol}(\{(\alpha)\})$. Now it can easily be verified that the polymorphisms of $\{(\alpha)\}$ are exactly those functions $f\colon D \to D$ for which $f(\alpha, \ldots, \alpha) = \alpha$ holds. Therefore, this holds for every polymorphism f of R. Now, let I be a solution of φ_R. Following part 1, the function f defined by $f(\alpha) = I(x_\alpha)$ is a polymorphism of R. Therefore, $f(\alpha) = \alpha$, and thus $I(x_\alpha) = \alpha$.

3. We make a case distinction.

Case 1: $f_{b\to a} \notin \text{Pol}(R)$. We prove that $I\colon \{x_a, x_b, x_c\} \to \{a, b, c\}$ is a solution of φ_R if and only if $I(x_a) \in \{a, b\}$, $I(x_b) = b$, and $I(x_c) = c$.

First, let $I \models \varphi_R$. Following part 1, we know, since all unary polymorphisms of R are conservative on $\{a, b\}$ and on $\{c\}$, that $I(x_c) = c$, and $I(x_a), I(x_b) \in \{a, b\}$. Assume that $I(x_b) \neq b$, i.e., $I(x_b) = a$. If $I(x_a) = a$, then, due to part 1, we know that $f_{b\to a}$ is a polymorphism of R, which is a contradiction to our assumption. For the other case, if $I(x_a) = b$, then we know that f' is a polymorphism of R, where $f'(a) = b$, $f'(b) = a$, and $f(c) = c$. Since $f_{a\to b} \in \text{Pol}(R)$, we conclude that $f' \circ f_{a\to b} = f_{b\to a} \in \text{Pol}(R)$, which is again a contradiction.

Now, let $I(x_a) \in \{a, b\}$, $I(x_b) = b$, and $I(x_c) = c$. It is obvious that if $I(x_a) = a$, then $I \models \varphi_R$: this follows from part 1, since the identity is always a polymorphism. Therefore, assume that $I(x_a) = b$. Then, by definition of the assignment, it corresponds to the function $f_{a\to b}$, which is a polymorphism of R, and due to part 1, I is a solution of φ.

Now, let $\psi = \bigwedge_{i=1}^{l} (x_i = c) \wedge (y_i \in \{a, b\})$ be an E-formula (without loss of generality, assume that there are equally many variables which we force to be c and which we force to be in $\{a, b\}$. If not, we repeat some clauses in the construction below). Then, by the above,

$$\exists x_b \bigwedge_{i=1}^{l} \varphi_R[x_a/y_i, x_c/x_i] \Leftrightarrow \psi,$$

since for the existentially quantified variable x_b, the value b is the only one which can be chosen. Therefore, we conclude $E \subseteq \langle R \rangle_{\text{cons}}$.

Case 2: $f_{b\to a} \in \text{Pol}(R)$. We construct the formula $\psi(x, x_c) := \varphi_R\,[x_a/x, x_b/x]$. We show that $I\colon \{x, x_c\} \to \{a, b, c\}$ is a solution of ψ if and only if $I(x) \in \{a, b\}$ and $I(x_c) = c$.

First, let $I \models \psi$. Since $\psi = \varphi_R\,[x_a/x, x_b/x]$, it follows from part 1, that f defined as $f(a) = f(b) = I(x)$, $f(c) = c$ is a polymorphism of R. Since all unary polymorphisms are conservative on $\{a, b\}$ and on $\{c\}$, it follows that $I(x) \in \{a, b\}$, $I(x_c) = c$.

Now, let $I(x) \in \{a, b\}$ and $I(x_c) = c$. Since $f_{a\to b} \in \text{Pol}(R)$, it follows from part 1, that I_1 defined by $I_1(x_c) = c, I_1(x_b) = I_1(x) = b$ is a solution of φ_R. Therefore, I_1 restricted to x, x_c is a solution of ψ. The second possible solution fulfilling the requirements is I_2, defined as $I_2(x) = a, I_2(x_c) = c$. We extend this to a solution of φ_R by setting $I_2(x_a) = I_2(x_b) = a$. This is a solution of φ_R, because $f_{b\to a}$ is a polymorphism of R.

As above, let $\psi = \bigwedge_{i=1}^{l} (x_i = c) \wedge (y_i \in \{a, b\})$ be an E-formula. Then, by the above,

$$\bigwedge_{i=1}^{l} \psi[x/y_i, x_c/x_i] \Leftrightarrow \psi.$$

Therefore, we conclude $E \subseteq \langle R \rangle_{\text{cons}}$.

□

4.2 Complete classification of the conservative case

With our results, we obtain a complete classification of the case where all polymorphisms are conservative on some partition $E = \{\{a,b\},\{c\}\}$, in the three-element case. Note that this result is not an algebraic characterization in the usual sense, since $\Gamma_R^{E\to D}$ being Schaefer does not only depend on the set of polymorphisms of R.

Lemma 4.5. *Let R be a relation over some domain D, and let E be a partition of D. If $E \subseteq \langle R\rangle_{cons}$, and R has an efficient enumeration algorithm, then R is semi-efficiently $E \to D$-enumerable.*

Proof. Let φ be an R-formula, and let $I\colon \mathrm{VAR}(\varphi) \to E$ be a partial E-assignment. Since $E \subseteq \langle R\rangle_{\mathrm{cons}}$ we can, using only constant-many new existentially quantified variables, force every variable $x \in \mathrm{VAR}(\varphi)$ to take values from $I(x)$ by adding the clause $(I(x))(x)$ (remember that $I(x)$ is one of the classes in E). Enumerating the solutions of this modified formula gives us all solutions of φ which are compatible with I, where each can appear c times for a constant c, depending on the constant number of existential quantifiers needed to express E. □

Corollary 4.6. *Let R be a relation on the three-element domain D, and let there be some partition $E = \{\{a,b\},\{c\}\}$ of D, such that all polymorphisms of R are conservative on the classes in E.*

- *If $f_{a\to b} \notin \mathrm{Pol}(R)$ and $f_{b\to a} \notin \mathrm{Pol}(R)$, then R has an efficient enumeration algorithm if and only if R is tractable.*
- *Otherwise, R has an efficient enumeration algorithm if and only if $\Gamma_R^{E\to D}$ is Schaefer, and R/E is Schaefer (or $\mathrm{P} = \mathrm{NP}$).*

Proof. First, note that since all polymorphisms of R are conservative on the classes in E, R cannot have a constant polymorphism. Now, assume that $f_{a\to b}$ and $f_{b\to a}$ are not polymorphisms of R. Then it follows that the only unary polymorphism of R which restricted to its range is the identity, is the identity itself. If R is tractable, it follows from Proposition 2.9 that R^+ is tractable, and therefore R has an efficient enumeration algorithm due to Theorem 2.4. If R is not tractable, then R obviously cannot have an efficient enumeration algorithm.

Now, assume that, without loss of generality, $f_{a\to b} \in \mathrm{Pol}(R)$. If R is not tractable, then R does not have an efficient enumeration algorithm. If R is tractable, then $f_{a\to b}(R)$ is tractable as well, since some R-formula φ is satisfiable if and only if $f_{a\to b}(\varphi)$ is. Since R does not have a constant polymorphism, it follows that $f_{a\to b}(R)$ is Schaefer. Since $f_{a\to b} \in \mathrm{Pol}(R)$, the set $E' = \{b,c\}$ is a representation system of E compatible with R. Therefore, R/E is efficiently enumerable if and only if R is efficiently E-enumerable, due to Lemma 3.13. From Lemma 4.4, we know that $E \subseteq \langle R\rangle_{\mathrm{cons}}$, and thus, Lemma 4.5 implies that R is semi-efficiently $E \to D$-enumerable. This implies, due to Lemma 3.15, that $\Gamma_R^{E\to D}$ is semi-efficiently enumerable. Since this is a Boolean constraint language, Proposition 3.26 implies that $\Gamma_R^{E\to D}$ is efficiently enumerable, and therefore, again due to Lemma 3.15, R is efficiently $E \to D$-enumerable.

On the other hand, if R is efficiently E-enumerable and efficiently $E \to D$-enumerable, then R has an efficient enumeration algorithm due to Theorem 3.11. □

5 The Galois connection

We show that the Galois connection does not work for the enumeration problem. This result shows that the three-element case behaves very differently from the Boolean case, since for the two-element domain, it follows from Proposition 3.26 that for a Boolean relation, the property of having an efficient enumeration algorithm only depends on the polymorphisms of the relation.

Theorem 5.1. *There exist relations R and R' such that $R' \in \langle R\rangle$, R has an efficient enumeration algorithm, and R' does not have a semi-efficient enumeration algorithm, unless* $\mathrm{P} = \mathrm{NP}$.

The idea of the proof is that the union of relations which are Schaefer usually is not Schaefer. With existential quantifiers, we achieve that there are tuples $\mathbf{v_1}, \mathbf{v_2}$ in R which are not E-equivalent for some partition E, but their corresponding tuples in R' are. The relation R in the construction we give is $\{a,b\}$-Schaefer. Therefore, the relations $\mathbf{v_1}^{E\to D}$ and $\mathbf{v_2}^{E\to D}$ are Schaefer, but their union is not. Since the corresponding tuples in R' are E-equivalent, this union appears in the constraint language $\Gamma_{R'}^{E\to D}$.

Proof. Let

$$R := \left(\begin{array}{l} \Big(\{a\}\times\{a\}\times\{b\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big)\times\{b\} \\ \cup\Big(\{a\}\times\{b\}\times\{a\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big)\times\{c\} \\ \cup\Big(\{b,c\}^7\setminus\{(c,c,c,c,c,c,b),(c,c,c,c,c,c,c)\}\Big) \end{array}\right)\times\{c\}$$
$$\cup\{(a,a,a,a,a,a,a,c)\}.$$

We first show that R does have an efficient enumeration algorithm, using Corollary 3.16. First note that $f_{a\to b}$ is a polymorphism of R: let $\mathbf{v}=(\alpha_1,\dots,\alpha_8)\in R$, and, without loss of generality, assume that at least one of the components of $\mathbf{v}$ is a (otherwise, $f_{a\to b}(\mathbf{v})=\mathbf{v}\in R$). In this case, by construction of R, the only components of $\mathbf{v}$ which can be c are α_7 and α_8. Since $R\subset\{a,b,c\}^7\times\{c\}$, $\alpha_8=c$ must hold. If $\alpha_7=b$, then $f_{a\to b}(\mathbf{v})=(b,b,b,b,b,b,b,c)\in R$. If $\alpha_7=c$, then $f_{a\to b}(\mathbf{v})=(b,b,b,b,b,b,c,c)\in R$. Therefore, $f_{a\to b}\in\mathrm{Pol}(R)$.

We now show that R is $\{a,b\}$-Schaefer. For this, let $\mathbf{v}=(\alpha_1,\dots,\alpha_8)\in R\cap\{b,c\}^8$. As above, we know that $\alpha_8=c$. If there is some $i\in\{1,\dots,6\}$ such that $\alpha_i=c$, then, by construction of R, we know that

$$R_{\mathbf{v}}^{a\to b}=\{(b,\dots,b)\}=:R_1.$$

Now assume $(\alpha_1,\dots,\alpha_6)=(b,\dots,b)$. If $\alpha_7=c$, then $\mathbf{v}=(b,b,b,b,b,b,c,c)$, and

$$R_{\mathbf{v}}^{a\to b}=\{a\}\times\{b\}\times\{a\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}=:R_2.$$

Finally, if $\alpha_7=b$ or $\alpha_7=a$, then $\mathbf{v}=(b,b,b,b,b,b,b,c)$, and

$$R_{\mathbf{v}}^{a\to b}=\{a\}\times\{a\}\times\{b\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\times\{b\}=:R_3.$$

We use the Boolean terminology to describe the relations R_1,R_2 and R_3. If we interpret b as 1, and a as 0, then it is easily seen that these relations are closed under the Boolean AND operator. It follows that R is $\{a,b\}$-Schaefer.

By definition of R it holds that

$$f_{a\to b}(R)=R\cap\{b,c\}^8=\Big(\{b,c\}^7\setminus\{(c,c,c,c,c,c,b),(c,c,c,c,c,c,c)\}\Big)\times\{c\}.$$

Again applying the Boolean vocabulary, when we interpret b as 1, and c as 0, then this relation is closed under the Boolean OR operator. Therefore, $f_{a\to b}(R)$ is Schaefer. Now, by Corollary 3.16, it follows that R has an efficient enumeration algorithm.

Now, let R' be defined by the formula

$$R'(x_1,\dots,x_7):\Longleftrightarrow\exists y R(x_1,\dots,x_6,y,x_7).$$

We claim that R' does not have an efficient enumeration algorithm. We show that if R' has a semi-efficient enumeration algorithm, then P = NP. For this, we first show that R' is not $\{a,b\}$-Schaefer.

By construction, it holds that

$$R'=\left(\begin{array}{l} \Big(\{a\}\times\{a\}\times\{b\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big) \\ \cup\Big(\{a\}\times\{b\}\times\{a\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big) \\ \cup\Big(\{b,c\}^6\setminus\{(c,c,c,c,c,c)\}\Big) \end{array}\right)\times\{c\}$$
$$\cup\{(a,a,a,a,a,a,c)\}.$$

Let $\mathbf{v}=(b,b,b,b,b,b,c)\in R'$. Then it is obvious that

$$R'^{a\to b}_{\mathbf{v}}=\begin{array}{l} \Big(\{a\}\times\{a\}\times\{b\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big) \\ \cup\Big(\{a\}\times\{b\}\times\{a\}\times\Big(\{a,b\}^3\setminus\{(b,b,b)\}\Big)\cup\{(b,b,b,b,b,b)\}\Big) \\ \cup\{(b,b,b,b,b,b)\}\cup\{(a,a,a,a,a,a)\}. \end{array}$$

We show that $R'^{a\to b}_{\mathbf{v}}$ is not Schaefer. Again, we identify the values with Boolean constants, we interpret b as 1 and a as 0. To show that $R'^{a\to b}_{\mathbf{v}}$ is not Schaefer, it suffices to show that it is not closed under the Boolean operators AND and OR, the ternary XOR, and the function $\mathrm{maj}(x, y, z) = xy \vee xz \vee yz$. To verify this, observe that the following equations hold:

$$\begin{pmatrix}0\\0\\1\\1\\0\\0\end{pmatrix} \vee \begin{pmatrix}0\\0\\1\\0\\1\\1\end{pmatrix} = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \notin R, \quad \begin{pmatrix}0\\0\\1\\1\\0\\0\end{pmatrix} \wedge \begin{pmatrix}0\\1\\0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\\0\\0\end{pmatrix} \notin R, \quad \begin{pmatrix}0\\0\\0\\0\\0\\0\end{pmatrix} \oplus \begin{pmatrix}0\\0\\1\\1\\0\\0\end{pmatrix} \oplus \begin{pmatrix}0\\1\\0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\1\\1\\1\\1\\0\end{pmatrix} \notin R,$$

$$\mathrm{maj}\left(\begin{pmatrix}0\\0\\1\\1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\\1\\1\end{pmatrix}\right) = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \notin R.$$

This shows that $R'^{a\to b}_{\mathbf{v}}$ is not Schaefer. In particular, R' is not $\{a, b\}$-Schaefer. Due to Lemma 4.2, R' does not have a semi-efficient enumeration algorithm, since we can express $x_c = c$ as $\exists y R'(y, y, y, y, y, y, x_c)$, and $x \in \{a, b\}$ as $(x_c = c) \wedge (x, x, x, x, x, x, x_c)$. □

6 Conclusion and future research

We have exhibited new enumeration algorithms for constraint languages.

The goal of our research is to completely answer the question "which relations allow for an efficient enumeration algorithm?", assuming we know for which relations the constraint satisfaction problem is solvable in polynomial time. The next logical step is to get broader criteria showing that some relation cannot be enumerated with polynomial delay. In light of Theorem 5.1, it is evident that new algebraic techniques are needed. In [SS06], we give first ideas of the algebraic concepts which hopefully can be applied to the enumeration problem.

There is a close correspondence between enumeration algorithms and the possible orderings they can produce, as shown by Theorem 3.25. Also, enumerating solutions with nested procedures for partial E-enumerability and $E_1 \to E_2$-enumerability defines an order on the solutions. We believe that the key to discover further enumeration algorithms might be to consider other possible orderings.

Another interesting question is to additionally demand that an efficient enumeration algorithm only needs polynomial space. For implementations in practice, this is a crucial requirement. All of the algorithms we presented here work with this restriction. We believe that for these algorithms, negative results can be achieved more easily, since the PSPACE bound ensures that the algorithm cannot simply calculate a big set of solutions in advance, printing them when needed.

Acknowledgment

We thank Nadia Creignou and Heribert Vollmer for very helpful discussions.

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.

BKJ00. A. Bulatov, A. Krokhin, and P. Jeavons. Constraint satisfaction problems and finite algebras. In *27th International Colloquium on Automata, Languages and Programming*, pages 272–282, 2000.

Bul02. A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings 43rd Symposium on Foundations of Computer Science*, pages 649–658. IEEE Computer Society Press, 2002.

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.

CH96. N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

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.

CKS01. N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* Monographs on Discrete Applied Mathematics. SIAM, 2001.

Coh04. D. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

Dal05. Victor Dalmau. Generalized majority-minority operations are tractable. In Prakash Panangaden, editor, *Proceedings of the Twentieth Annual IEEE Symp. on Logic in Computer Science, LICS 2005*, pages 438–447. IEEE Computer Society Press, June 2005.

DK06. V. Dalmau and A. Krokhin. Majority constraints have bounded pathwidth duality. Technical Report NI06017-LAA, Isaac Newton Institute for Mathematical Sciences, 2006.

FV98. T. Feder and M. Y. Vardi. The computational structure of monotone monadis SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

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

JPY88. D. Johnson, C. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.

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

Rei05. Omer Reingold. Undirected st-connectivity in log-space. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 376–385, New York, NY, USA, 2005. ACM Press.

RV00. S. Reith and H. Vollmer. Optimal satisfiability for propositional calculi and constraint satisfaction problems. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *Lecture Notes in Computer Science*, pages 640–649. Springer Verlag, 2000.

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. New algebraic tools for constraint satisfaction, 2006. *These Proceedings.*

Val79. L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):411–421, 1979.