Journal of Artificial Intelligence Research 45 (2012) 731-759 Submitted 06/12; published 12/12

Tractable Set Constraints

Manuel Bodirsky BODIRSKY@LIX.POLYTECHNIQUE.FR
École Polytechnique, LIX
(UMR 7161 du CNRS)
91128 Palaiseau, France

Martin Hils HILS@MATH.UNIV-PARIS-DIDEROT.FR
Institut de Mathématiques de Jussieu
(UMR 7586 du CNRS)
Université Paris Diderot Paris 7
UFR de Mathématiques
75205 Paris Cedex 13, France

Abstract

Many fundamental problems in artificial intelligence, knowledge representation, and verification involve reasoning about sets and relations between sets and can be modeled as *set constraint satisfaction problems (set CSPs)*. Such problems are frequently intractable, but there are several important set CSPs that are known to be polynomial-time tractable. We introduce a large class of set CSPs that can be solved in quadratic time. Our class, which we call $\mathcal{EI}$, contains all previously known tractable set CSPs, but also some new ones that are of crucial importance for example in description logics. The class of $\mathcal{EI}$ set constraints has an elegant universal-algebraic characterization, which we use to show that every set constraint language that properly contains all $\mathcal{EI}$ set constraints already has a finite sublanguage with an NP-hard constraint satisfaction problem.

1. Introduction

Constraint satisfaction problems are computational problems where, informally, the input consists of a finite set of variables and a finite set of constraints imposed on those variables; the task is to decide whether there is an assignment of values to the variables such that all the constraints are simultaneously satisfied. *Set constraint satisfaction problems* are special constraint satisfaction problems where the values are sets, and the constraints might, for instance, force that one set y includes another set x, or that one set x is disjoint to another set y. The constraints might also be ternary (or, more generally, of any finite arity), such as the constraint that the intersection of two sets x and y is contained in z, in symbols $(x \cap y) \subseteq z$.

To systematically study the computational complexity of constraint satisfaction problems, it has turned out to be a fruitful approach to consider constraint satisfaction problems CSP(Γ) where the set of allowed constraints is formed from a fixed finite set Γ of relations $R \subseteq D^k$ over a (possibly infinite) common domain D. For example, if D equals $\mathbb{Q}$, the rational numbers, and Γ is $\{<\}$ where $<$ is the usual order of the rationals, then CSP(Γ) is the problem of deciding whether a given set of (binary) constraints of the form $x < y$ has a common solution over the rational numbers. This way of parametrizing the constraint satisfaction problem by a *constraint language* Γ has led to many strong algorithmic results (e.g., Bulatov & Dalmau, 2006; Idziak, Markovic, McKenzie, Valeriote, & Willard, 2010; Barto & Kozik, 2009; Bodirsky & Kutz, 2007; Bodirsky & Kára, 2009), and to many powerful hardness conditions for large classes of constraint satisfaction problems (Schaefer, 1978; Bulatov, Krokhin, & Jeavons, 2005; Bulatov, 2003, 2006; Bodirsky & Kára, 2009).

A *set constraint language* Γ is a set of relations $R \subseteq (\mathcal{P}(\mathbb{N}))^k$ where the common domain $D = \mathcal{P}(\mathbb{N})$ is the set of all subsets of the natural numbers; moreover, we require that each relation R can be defined by a Boolean combination of equations over the signature $\sqcap$, $\sqcup$, c, $\mathbf{0}$, and $\mathbf{1}$, which are function symbols for intersection, union, complementation, the empty and full set, respectively. Details of the formal definition of set constraint languages can be found in Section 3. In Section 4, we give many examples of set constraint languages. The choice of $\mathbb{N}$ is just for notational convenience; we could have selected any infinite set for our purposes, e.g., $\mathbb{R}^n$ instead of $\mathbb{N}$, a more natural choice in spatial reasoning. One may even replace $\mathcal{P}(\mathbb{N})$ by any infinite Boolean algebra (see Theorem 28).

In the following, a *set constraint satisfaction problem (set CSP)* is a problem of the form CSP(Γ) for a *finite* set constraint language Γ. It has been shown by Marriott and Odersky that all set CSPs are contained in NP.

Drakengren and Jonsson (1998) initiated the search for set CSPs that can be solved in polynomial time. They showed that CSP($\{\subseteq, ||, \neq\}$) can be solved in polynomial time, where

- $x \subseteq y$ holds iff x is a subset of or equal to y;
- $x \mid\mid y$ holds iff x and y are disjoint sets; and
- $x \neq y$ holds iff x and y are distinct sets.

They also showed that CSP(Γ) can be solved in polynomial time if all relations in Γ can be defined by formulas of the form

$$x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x_0 \subseteq y_0$$

or of the form

$$x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x_0 \mid\mid y_0$$

where $x_0, \ldots, x_k, y_0, \ldots, y_k$ are not necessarily distinct variables (Drakengren & Jonsson, 1998, Thm. 20). We will call the set of all relations that can be defined in this way *Drakengren and Jonsson's set constraint language*. It is easy to see that the algorithm they present runs in time quadratic in the size of the input. On the other hand, they also show that if Γ contains the relation defined by $x_1 \neq y_1 \vee x_2 \neq y_2$ and the relation defined by $x_1 \subseteq x_0 \vee x_2 \subseteq x_0$ then the problem CSP(Γ) is NP-hard (Drakengren & Jonsson, 1998, Thm. 22).

1.1 Contributions and Outline.

We present a significant extension of Drakengren and Jonsson's (1998) set constraint language (Section 4) whose CSP can still be solved in quadratic time in the input size (Section 5); we call this set constraint language $\mathcal{EI}$. Unlike Drakengren and Jonsson's set constraint language, our language also contains the ternary relation defined by $(x \cap y) \subseteq z$, which is a relation that is of particular interest in description logics – we will discuss this below. Moreover, we show that any further extension of $\mathcal{EI}$ contains a finite sublanguage with an NP-hard set CSP (Section 6), using concepts from model theory and universal algebra. In this sense, we present a *maximal* tractable class of set constraint satisfaction problems.

Our algorithm is based on the concept of *independence* in constraint languages which was discovered several times independently in the 90's (Lassez & McAloon, 1989; Jonsson & Bäckström, 1998; Marriott & Odersky, 1996, see also Koubarakis, 2001; Broxvall, Jonsson, & Renz, 2002; Cohen, Jeavons, Jonsson, & Koubarakis, 2000); however, we apply this concept *twice* in a novel, nested way, which leads to a two level resolution procedure that can be implemented to run in quadratic time. The technique we use to prove the correctness of the algorithm is also an important contribution of our paper, and we believe that a similar approach can be applied in many other contexts; our technique is inspired by the already mentioned connection to universal algebra.

1.2 Application Areas and Related Literature

We mention three different contexts where set constraints appeared in the literature.

1.2.1 Set Constraints for Programming Languages.

Set constraints find applications in program analysis; here, a set constraint is of the form $X \subseteq Y$, where X and Y are set expressions. Examples of set expressions are $\mathbf{0}$ (denoting the empty set), set-valued variables, and union and intersection of sets, but also expressions of the form $f(Z_1, Z_2)$ where f is a function symbol and Z_1, Z_2 are again set expressions. Unfortunately, the worst-case complexity of most of the reasoning tasks considered in this setting is *very* high, often EXPTIME-hard (for a survey on this, see Aiken, 1994). More recently, it has been shown that the quantifier-free combination of set constraints (without function symbols) and cardinality constraints (quantifier-free Pressburger arithmetic) has a satisfiability problem in NP (Kuncak & Rinard, 2007). This logic (called QFBAPA) is interesting for program verification (Kuncak, Nguyen, & Rinard, 2006).

1.2.2 Tractable Description Logics.

Description logics are a family of knowledge representation formalisms that can be used to formalize and reason with concept definitions. The computational complexity of most of the computational tasks that have been studied for the various formalisms is usually quite high. However, in the last years a series of description logics, for example $\mathcal{EL}$, $\mathcal{EL}^{++}$, Horn-$\mathcal{F}L_0$, and various extensions and fragments (Küsters & Molitor, 2002; Baader, 2003; Baader, Brandt, & Lutz, 2005; Krötzsch, Rudolph, & Hitzler, 2006), has been discovered where crucial tasks such as e.g. entailment, concept satisfiability and knowledge base satisfiability can be decided in polynomial time.

Two of the basic assertions that can be made in $\mathcal{EL}^{++}$ and Horn-$\mathcal{F}L_0$ are $C_1||C_2$ (*there is no* C_1 *that is also* C_2) and $C_1 \cap C_2 \subseteq C_3$ (*every* C_1 *that is* C_2 *is also* C_3), for concept names C_1, C_2, C_3. These are $\mathcal{EI}$ set constraints, and the latter has not been treated in the framework of Drakengren and Jonsson. None of the description logics with a tractable knowledge base satisfiability problem contains all $\mathcal{EI}$ set constraints.

1.2.3 Spatial Reasoning.

Several spatial reasoning formalisms (like RCC-5 and RCC-8) are closely related to set constraint satisfaction problems. These formalisms allow to reason about relations between *regions*; in the fundamental formalism RCC-5 (see, e.g., Jonsson & Drakengren, 1997), one can think of a region as a non-empty set, and possible (binary) relationships are containment, disjointness, equality, overlap, and disjunctive combinations thereof. Thus, the exclusion of the empty set is the most prominent difference between the set constraint languages studied by Drakengren and Jonsson (1998) — contained in the class of set constraint languages considered here — and RCC-5 and its fragments. We will see in Section 3 that the CSP for RCC-5 and the CSPs for all its reducts are set CSPs (Proposition 2).

2. Constraint Satisfaction Problems

To use existing terminology in logic and model theory, it will be convenient to describe constraint languages by *structures* (see, e.g., Hodges, 1993). A *structure* Γ is a tuple $(D; f_1^\Gamma, f_2^\Gamma, \ldots, R_1^\Gamma, R_2^\Gamma, \ldots)$ where D is a set (the *domain* of Γ), each f_i^Γ is a function from $D^{k_i} \to D$ (where k_i is called the *arity* of f_i^Γ), and each R_i^Γ is a relation over D, i.e., a subset of D^{l_i} (where l_i is called the *arity* of R_i^Γ). For each function f_i^Γ we assume that there is a *function symbol* which we denote by f_i, and for each relation R_i^Γ we have a *relation symbol* which we denote by R_i. Constant symbols will be treated as 0-ary function symbols. The set τ of all relation and function symbols for some structure Γ is called the *signature* of Γ, and we also say that Γ is a τ*-structure*. If the signature of Γ only contains relation symbols and no function symbols, we also say that Γ is a *relational structure*. In the context of constraint satisfaction, relational structures Γ are also called *constraint languages*, and a constraint language Γ' is called a *sublanguage* (or *reduct*) of a constraint language Γ if the relations in Γ' are a subset of the relations in Γ (and Γ is called an *expansion* of Γ').

Let Γ be a relational structure with domain D and a *finite* signature τ. The *constraint satisfaction problem* for Γ is the following computational problem, also denoted by CSP(Γ): given a finite set of variables V and a conjunction Φ of finitely many atomic formulas of the form $R(x_1, \ldots, x_k)$, where $x_1, \ldots, x_k \in V$ and $R \in \tau$, is Φ *satisfiable* in Γ; that is, does there exist an assignment $s\colon V \to D$ such that for every constraint $R(x_1, \ldots, x_k)$ in the input we have that $(s(x_1), \ldots, s(x_k)) \in R^\Gamma$?

The mapping s is also called a *solution* to the *instance* Φ of CSP(Γ), and the conjuncts of Φ are called *constraints*. Note that we only introduce constraint satisfaction problems CSP(Γ) for *finite constraint languages*, i.e., relational structures Γ with a *finite* relational signature.

Example 1. *The problem* CSP$((\mathbb{Q}; <))$ *is the problem of deciding whether a given set of constraints of the form* $x < y$ *has a solution that simultaneously satisfies all constraints.*

3. Set Constraint Languages

In this section, we give formal definitions of set constraint languages. Let $\mathfrak{S}$ be the structure with domain $\mathcal{P}(\mathbb{N})$, the set of all subsets of natural numbers, and with signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$, where

- $\sqcap$ is a binary function symbol that denotes intersection, i.e., $\sqcap^{\mathfrak{S}} = \cap$;
- $\sqcup$ is a binary function symbol for union, i.e., $\sqcup^{\mathfrak{S}} = \cup$;
- c is a unary function symbol for complementation, i.e., $c^{\mathfrak{S}}$ is the function that maps $S \subseteq \mathbb{N}$ to $\mathbb{N} \backslash S$;
- $\mathbf{0}$ and $\mathbf{1}$ are constants (treated as 0-ary function symbols) denoting the empty set $\varnothing$ and the full set $\mathbb{N}$, respectively.

Sometimes, we simply write $\sqcap$ for the function $\sqcap^{\mathfrak{S}}$ and $\sqcup$ for the function $\sqcup^{\mathfrak{S}}$, i.e., we do not distinguish between a function symbol and the respective function. We use the symbols $\sqcap, \sqcup$ and not the symbols $\cap, \cup$ to prevent confusion with meta-mathematical usages of $\cap$ and $\cup$ in the text.

A *set constraint language* is a relational structure whose relations have a quantifier-free definition in $\mathfrak{S}$. We always allow equality in first-order formulas, and the equality symbol $=$ is always interpreted to be the true equality relation on the domain of the structure. We write $x \sqsubseteq y$ as an abbreviation for $x \sqcap y = x$.

Example 2. *The ternary relation* $\{(x, y, z) \in \mathcal{P}(\mathbb{N})^3 \mid x \sqcap y \sqsubseteq z\}$ *has the quantifier-free definition* $z \sqcap (x \sqcap y) = x \sqcap y$ *over* $\mathfrak{S}$.

Theorem 1 (See Marriott & Odersky, 1996, Proposition 5.8)**.** *Let* Γ *be a set constraint language with a finite signature. Then* $\mathrm{CSP}(\Gamma)$ *is in NP.*

It is easy to see that there are NP-hard set CSPs, as shown in the next example.

Example 3. *Consider the set constraint language* Γ *that contains the eight relations*

$$\begin{aligned}
&\{(x, y, z) \mid x = \mathbf{1} \vee y = \mathbf{1} \vee z = \mathbf{1}\}\\
&\{(x, y, z) \mid x = \mathbf{1} \vee y = \mathbf{1} \vee \bar{z} = \mathbf{1}\}\\
&\{(x, y, z) \mid x = \mathbf{1} \vee \bar{y} = \mathbf{1} \vee z = \mathbf{1}\}\\
&\{(x, y, z) \mid x = \mathbf{1} \vee \bar{y} = \mathbf{1} \vee \bar{z} = \mathbf{1}\}\\
&\{(x, y, z) \mid \bar{x} = \mathbf{1} \vee y = \mathbf{1} \vee z = \mathbf{1}\}\\
&\{(x, y, z) \mid \bar{x} = \mathbf{1} \vee y = \mathbf{1} \vee \bar{z} = \mathbf{1}\}\\
&\{(x, y, z) \mid \bar{x} = \mathbf{1} \vee \bar{y} = \mathbf{1} \vee z = \mathbf{1}\}\\
&\{(x, y, z) \mid \bar{x} = \mathbf{1} \vee \bar{y} = \mathbf{1} \vee \bar{z} = \mathbf{1}\}\ .
\end{aligned}$$

Then the set CSP for those relations is the well-known 3-SAT problem, which is NP-complete (Garey & Johnson, 1978).

It is well-known that the structure $(\mathcal{P}(\mathbb{N}); \sqcup, \sqcap, c, \mathbf{0}, \mathbf{1})$ is a Boolean algebra, with

- $\mathbf{0}$ playing the role of false, and $\mathbf{1}$ playing the role of true;
- c playing the role of $\neg$;
- $\sqcap$ and $\sqcup$ playing the role of $\wedge$ and $\vee$, respectively.

We refer to the work of Koppelberg (1989) for background on Boolean algebras.

To not confuse logical connectives with the connectives of Boolean algebras, we always use the symbols $\sqcap$, $\sqcup$, and c instead of the usual function symbols $\wedge$, $\vee$, and $\neg$ in Boolean algebras. To facilitate the notation, we also write $\bar{x}$ instead of $c(x)$, and $x \neq y$ instead of $\neg(x = y)$.

We assume that all terms t over the functional signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$ are written in *(inner) conjunctive normal form (CNF)*, i.e., as $t = \bigsqcap_{i=1}^{n} \bigsqcup_{j=1}^{n_i} l_{ij}$ where l_{ij} is either of the form $\bar{x}$ or of the form x for a variable x. Note that every term over $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$ can be rewritten into an equivalent term of this form, using the usual laws of Boolean algebras (Boole, 1847). We allow the special case $n = 0$ (in which case t becomes $\mathbf{1}$), and the special case $n_i = 0$ (in which case $\bigsqcup_{j=1}^{n_i} l_{ij}$ becomes $\mathbf{0}$). We refer to $c_i := \{l_{ij} \mid 1 \leqslant j \leqslant n_i\}$ as an *(inner) clause* of t, and to l_{ij} as an *(inner) literal* of c_i. We say that a set of inner clauses is *satisfiable* if there exists an assignment from $V \to \mathcal{P}(\mathbb{N})$ such that for all inner clauses, the union of the evaluation of all literals equals $\mathbb{N}$ (this is the case if and only if the formula $\bigsqcap_{i=1}^{n} c_i = \mathbf{1}$ has a satisfying assignment).

Example 4. *Inequality $x \neq y$ on $\mathcal{P}(\mathbb{N})$ can be equivalently written as $(y \sqcup \bar{x}) \sqcap (x \sqcup \bar{y}) \neq \mathbf{1}$; in this formula, we have two inner clauses, each with a positive and a negative inner literal.*

We assume that all quantifier-free formulas ϕ over the signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$ are written in *(outer) conjunctive normal form (CNF)*, i.e., as $\phi = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m_i} L_{ij}$ where L_{ij} is either of the form $t = \mathbf{1}$ (a *positive (outer) literal*) or of the form $t \neq \mathbf{1}$ (a *negative (outer) literal*). Again, it is well-known and easy to see that we can for every quantifier-free formula find a formula in this form which is equivalent to it in every Boolean algebra. We refer to $C_i := \{L_{ij} \mid 1 \leqslant j \leqslant m_i\}$ as an *(outer) clause* of ϕ, and to L_{ij} as an *(outer) literal* of C_i. Whenever convenient, we identify ϕ with its set of clauses.

Example 5. *Consider the formula $(x = y \wedge y \neq z) \vee (x \neq y \wedge y = z)$. It can be rewritten as $(x = y \vee y = z) \wedge (x \neq y \vee y \neq z)$. When we subsequently replace inequality literals $x \neq y$ by $(y \sqcup \bar{x}) \sqcap (x \sqcup \bar{y}) \neq \mathbf{1}$ (see Example 4), we arrive at a formula which is in the discussed normal form: it has two outer clauses, one with two positive outer literals, and the other with two negative outer literals.*

As mentioned in the introduction, all CSPs for reducts of RCC-5 (and therefore also many reducts of RCC-8) can be formulated as set CSPs. The network satisfaction problem for RCC-5 can be seen as the CSP for the following structure $\mathcal{RCC}$-5 with domain $\mathcal{P}(\mathbb{N})\backslash\{\varnothing\}$ and all binary relations given as follows: for each relation R of $\mathcal{RCC}$-5 there exists a quantifier-free $\{\sqcup, \sqcap, \mathbf{c}, \mathbf{0}, \mathbf{1}\}$-formula $\phi(x_1, x_2)$ such that $(s_1, s_2) \in R$ if and only if both s_1 and s_2 are non-empty, and $\phi(s_1, s_2)$ is true in $\mathfrak{S}$. It is clear that there are only finitely many inequivalent quantifier-free formulas over the language $\{\sqcup, \sqcap, \mathbf{c}, \mathbf{0}, \mathbf{1}\}$, and hence $\mathcal{RCC}$-5 has only finitely many relations.

Proposition 2. *Let Γ be a reduct of $\mathcal{RCC}$-5. Then there exists a set constraint language Δ such that* CSP(Γ) *and* CSP(Δ) *are the same problem.*

Proof. Let $\phi_1, \ldots, \phi_n$ be the formulas the define the relations of Γ (which has domain $\mathcal{P}(\mathbb{N})\backslash\{\varnothing\}$). Let Δ be the structure with domain $\mathcal{P}(\mathbb{N})$ and relations $R_1, \ldots, R_n$ defined by $\phi_1, \ldots, \phi_n$ in $\mathfrak{S}$. That is, the only difference between Γ and Δ is the additional element $\{\varnothing\}$, which appears in none of the relations of Δ. We have to prove that every finite conjunction Φ of atomic formulas of the form $R_i(x_1, \ldots, x_k)$ is satisfiable in Γ if and only if the conjunction is satisfiable in Δ. If Φ is satisfiable over Δ then it is clearly also satisfiable over Γ since Δ is an induced substructure of Γ. Conversely, if Φ is satisfiable over Γ, then any solution must have the property that if $s(x_i) = \varnothing$, then x_i does not appear in any constraint (since the constraints force that their arguments are non-empty). Hence, setting $s(x_i)$ to any non-empty subset of $\mathbb{N}$ will still be a solution, which implies the claim. $\square$

Proposition 2 shows that the class of set CSPs contains the class of all CSPs for reducts of $\mathcal{RCC}$-5. The inclusion is clearly strict: there are set CSPs that cannot be formulated as CSPs for reducts of $\mathcal{RCC}$-5, since relations in set constraint languages can have arbitrary arity, while reducts of $\mathcal{RCC}$-5 only contain binary relations.

4. Horn-Horn Set Constraints

In this section, we study Horn-Horn set constraints, a class of set constraints which admits an intuitive syntactic description and which is thus easy to define. Universal algebraic considerations on this class in Section 4.2 lead us to another class of set constraints, called $\mathcal{EI}$ and introduced in Section 4.3. This class strictly contains the class of Horn-Horn set constraints. Its universal algebraic description is the key for the maximality result which we will prove in Section 6. As for tractability, set constraint languages from $\mathcal{EI}$ allow for a (linear-time) reduction to satisfiability of Horn-Horn clauses (see Proposition 22). We note that these classes of set constraints (Horn-Horn and $\mathcal{EI}$) have not been studied before.

4.1 Horn-Horn Relations

Definition 3. A quantifier-free formula is called *Horn-Horn* if

1. every outer clause is *outer Horn*, i.e., contains at most one positive outer literal, and
2. every inner clause of positive outer literals is *inner Horn*, i.e., contains at most one positive inner literal.

A relation $R \subseteq \mathcal{P}(\mathbb{N})^k$ is called

- *outer Horn* if it can be defined over $\mathfrak{S}$ by a conjunction of outer Horn clauses;
- *inner Horn* if it can be defined over $\mathfrak{S}$ by a formula of the form $(c_1 \sqcap \cdots \sqcap c_k) = \mathbf{1}$ where each c_i is inner Horn;
- *Horn-Horn* if it can be defined by a Horn-Horn formula over $\mathfrak{S}$.

Example 6. *Inequality $\neq$ is Horn-Horn: recall that it may defined by $(y \sqcup \bar{x}) \sqcap (x \sqcup \bar{y}) \neq \mathbf{1}$.*

Example 7. *Using the previous example, the relation* $\{(x, y, u, v) \mid x \neq y \vee u = v\}$ *can easily be seen to be Horn-Horn, too.*

Example 8. *The ternary relation* $\{(x, y, z) \mid x \sqcap y \sqsubseteq z\}$*, which we have encountered above, has the Horn-Horn definition* $\bar{x} \sqcup \bar{y} \sqcup z = \mathbf{1}$*.*

Proposition 4. *Drakengren and Jonsson's set constraint language only contains Horn-Horn relations.*

Proof. The disjointness relation $||$ has the definition $\bar{x} \sqcup \bar{y} = \mathbf{1}$, so is inner Horn. The inequality relation $\neq$ is Horn-Horn since both inner clauses in its definition $(y \sqcup \bar{x}) \sqcap (x \sqcup \bar{y}) \neq \mathbf{1}$ have only one positive inner literal. The inclusion relation $x \subseteq y$ has the definition $y \sqcup \overline{x} = \mathbf{1}$, so is inner Horn.

Horn-Horn is preserved under adding additional outer disequality literals to the outer clauses, so all relations considered in Drakengren and Jonsson's language are Horn-Horn. $\square$

4.2 Universal Algebraic Preliminaries

As we will see in this section, the class of Horn-Horn formulas is preserved by several important functions defined on the set of subsets of natural numbers.

Definition 5.
- Let $i\colon (\mathcal{P}(\mathbb{N}))^2 \to \mathcal{P}(\mathbb{N})$ be the function that maps a pair of sets (S_1, S_2) to the set $\{2a \mid a \in S_1\} \cup \{2a + 1 \mid a \in S_2\}$;
- denote by Fin^* the set of finite non-empty subsets of $\mathbb{N}$, and let $F\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathrm{Fin}^*)$,
$$F(S) := \{S_0 \subseteq S \mid S_0 \text{ is finite and non-empty}\};$$
- let $G\colon \mathbb{N} \to \mathrm{Fin}^*$ be a bijection (since both sets are countable, such a bijection exists);
- let $e\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ be defined by
$$e(S) = \{G^{-1}(T) \mid T \in F(S)\};$$
- let ei be the function defined by $ei(x, y) = e(i(x, y))$.

Intuitively, the functions e and ei are designed so that they 'forget unions' (this will be formalized in Definition 34), while preserving the other basic operations in $\mathfrak{S}$ (see Lemma 11 and Proposition 8 for the case of e).

Definition 6. Let $f\colon (\mathcal{P}(\mathbb{N}))^k \to \mathcal{P}(\mathbb{N})$ be a function, and $R \subseteq \mathcal{P}(\mathbb{N})^l$ be a relation. Then we say that f *preserves* R if the following holds: for all $a^1, \ldots, a^k \in (\mathcal{P}(\mathbb{N}))^l$ we have that $(f(a^1_1, \ldots, a^k_1), \ldots, f(a^1_l, \ldots, a^k_l)) \in R$ if $a^i \in R$ for all $i \leqslant k$. If f does not preserve R, we also say that f *violates* R. We say that f *strongly preserves* R if for all $a^1, \ldots, a^k \in (\mathcal{P}(\mathbb{N}))^l$ we have that $(f(a^1_1, \ldots, a^k_1), \ldots, f(a^1_l, \ldots, a^k_l)) \in R$ if and only if $a^i \in R$ for all $i \leqslant k$. If ϕ is a first-order formula that defines a relation R over $\mathfrak{S}$, and f preserves (strongly preserves) R, then we also say that f *preserves (strongly preserves)* ϕ. Finally, if $g\colon (\mathcal{P}(\mathbb{N}))^l \to \mathcal{P}(\mathbb{N})$ is a function, we say that f *preserves (strongly preserves)* g if it preserves (strongly preserves) the graph of g, i.e., the relation $\big\{(x_1, \ldots, x_l, g(x_1, \ldots, x_l)) \mid x_1, \ldots, x_l \subseteq \mathbb{N}\big\}$.

Note that if an injective function f preserves a function g, then it also strongly preserves g.

Example 9. *Consider the function* $f\colon (\mathcal{P}(\mathbb{N}))^2 \to \mathcal{P}(\mathbb{N})$, $(x,y) \mapsto x \sqcup y$. *Then* f *preserves* $\sqsubseteq$, *since* $x \sqcup y \sqsubseteq x' \sqcup y'$ *whenever* $x \sqsubseteq x'$ *and* $y' \sqsubseteq y$. *On the other hand,* f *does not strongly preserve* $\sqsubseteq$, *as is shown by* $f(\mathbb{N}, \varnothing) = f(\varnothing, \mathbb{N})$.

Fact 7. *The mapping* i *is an isomorphism between* $\mathfrak{S}^2$ *and* $\mathfrak{S}$.

Proof. The mapping i can be inverted by the mapping that sends $S \subseteq \mathbb{N}$ to $\big(\{a \mid 2a \in S\}, \{a \mid 2a+1 \in S\}\big)$. It is straightforward to verify that i strongly preserves $\mathbf{0}$, $\mathbf{1}$, c, $\sqcup$, $\sqcap$.

- Clearly, $i(x,y) = \varnothing$ if and only if $x = y = \varnothing$.
- Similarly, since the natural numbers are partitioned by the even and odd numbers, $i(x,y) = \mathbb{N}$ if and only if $x = y = \mathbb{N}$.
- Let S_1 and S_2 be subsets of $\mathbb{N}$. To verify that i preserves c we have to show that $i(c(S_1,S_2))$, which is by definition equal to $i(\overline{S_1}, \overline{S_2})$, equals $c(i(S_1,S_2))$. Suppose that $a = 2a'$. Then:

$$\begin{aligned} a \in i(\overline{S_1}, \overline{S_2}) &\Leftrightarrow a' \in \overline{S_1} \\ &\Leftrightarrow a' \notin S_1 \\ &\Leftrightarrow 2a' \notin i(S_1,S_2) \\ &\Leftrightarrow a \in \overline{i(S_1,S_2)} \end{aligned}$$

  The argument for $a = 2a'+1$ is analogous. Thus, i preserves c. Since i is injective, it even strongly preserves c.

- Let (S_1,S_2) and (T_1,T_2) be from $(\mathcal{P}(\mathbb{N})^2$. We have to show that $i((S_1,S_2) \sqcup (T_1,T_2))$, which is by definition equal to $i(S_1 \sqcup T_1, S_2 \sqcup T_2)$, equals $i(S_1,S_2) \sqcup i(T_1,T_2)$. With $a = 2a'$ as before:

$$\begin{aligned} a \in i(S_1 \sqcup T_1, S_2 \sqcup T_2) &\Leftrightarrow a' \in (S_1 \sqcup T_1) \\ &\Leftrightarrow 2a' \in i(S_1,S_2) \sqcup i(T_1,T_2) \end{aligned}$$

  The argument for $a = 2a'+1$ is again analogous. Thus, i preserves $\sqcup$. Since i is injective, it even strongly preserves $\sqcup$.

The verification for $\sqcap$ is similar to that for $\sqcup$. ☐

Proposition 8. *The function* e *has the following properties.*

- *e is injective,*
- *e strongly preserves* $\mathbf{1}$, $\mathbf{0}$, *and* $\sqcap$, *and*
- *for* $x,y,z \in \mathcal{P}(\mathbb{N})$ *such that* $x \sqcup y = z$, *not* $x \sqsubseteq y$, *and not* $y \sqsubseteq x$, *we have that* $e(x) \sqcup e(y) \subsetneqq e(z)$.

Proof. We verify the properties one by one. Since G is bijective, $e(x) = e(y)$ if and only if x and y have the same finite subsets. This is the case if and only if $x = y$, and hence e is injective. Thus, to prove that e *strongly* preserves $\mathbf{1}$, $\mathbf{0}$, and $\sqcap$, it suffices to check that e preserves $\mathbf{1}$, $\mathbf{0}$, and $\sqcap$.

Since G is bijective, we have that $G(\mathbb{N})$ equals the set of all finite subsets of $\mathbb{N}$, and hence $e(\mathbb{N}) = \mathbb{N}$, which shows that e preserves $\mathbf{1}$. We also compute $e(\varnothing) = G^{-1}(F(\varnothing)) = G^{-1}(\varnothing) = \varnothing$.

Next, we verify that for all $x, y \in \mathcal{P}(\mathbb{N})$ we have $e(x) \sqcap e(y) = e(x \sqcap y)$. Let $a \in \mathbb{N}$ be arbitrary. We have $a \in e(x) \sqcap e(y)$ if and only if $G(a) \in F(x) \cap F(y)$. By definition of F and since $G(a)$ is a finite subset of $\mathbb{N}$, this is the case if and only if $G(a) \in F(x \sqcap y)$. This is the case if and only if $a \in e(x \sqcap y)$, which concludes the proof that e preserves $\sqcap$.

We verify that if $x \sqcup y = z$, not $x \sqsubseteq y$, and not $y \sqsubseteq x$, then $e(x) \sqcup e(y) \subsetneqq e(z)$. First observe that for all $u, v \subseteq \mathbb{N}$ with $u \sqsubseteq v$ we have $e(u) \sqsubseteq e(v)$ since e preserves $\sqcap$. This implies that $e(x) \sqcup e(y) \sqsubseteq e(z)$. Since $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$, there are a, b such that $a \in x$, $a \notin y$, $b \in y$, $b \notin x$. Then we have that $\{a, b\} \in F(z)$, but $\{a, b\} \notin F(x) \cup F(y)$. Hence, $G^{-1}(\{a, b\}) \in e(z)$, but $G^{-1}(\{a, b\}) \notin e(x) \sqcup e(y)$. This shows that $e(z) \neq e(x) \sqcup e(y)$. □

Note that in particular e preserves $\sqsubset$, $\sqsubseteq$, and $||$. Moreover, $e(c(x)) \sqsubseteq c(e(x))$: this follows from preservation of $||$, since $x||c(x)$, and therefore $e(x)||e(c(x))$, which is equivalent to the inclusion above. Both e and i strongly preserve $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and therefore also ei strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$.

The following is a direct consequence of the fact that isomorphisms between Γ^k and Γ preserve Horn formulas over Γ; since the simple proof is instructive for what follows, we give it for the special case that is relevant here.

Proposition 9. *Outer Horn relations are preserved by i.*

Proof. Let ϕ be a conjunction of outer Horn clauses with variables V. Let $\{t_0 = \mathbf{1}, t_1 \neq \mathbf{1}, \ldots, t_k \neq \mathbf{1}\}$ be an outer clause of ϕ. Let $u, v \colon V \to \mathcal{P}(\mathbb{N})$ be two assignments that satisfy this clause. Let $w \colon V \to \mathcal{P}(\mathbb{N})$ be given by $x \mapsto i(u(x), v(x))$. Suppose that w satisfies $t_j = \mathbf{1}$ for all $1 \leqslant j \leqslant k$. Since i is injective we must have that $t_j = \mathbf{1}$ for both u and v for $1 \leqslant j \leqslant k$, and therefore neither assignment satisfies the negative literals. Hence, u and v must satisfy $t_0 = \mathbf{1}$. Since i is an isomorphism between $\mathfrak{S}^2$ and $\mathfrak{S}$, it preserves in particular $t_0 = \mathbf{1}$, and hence w also satisfies $t_0 = \mathbf{1}$. □

Proposition 10. *Inner Horn relations are strongly preserved by e.*

Proof. Observe that $x \sqcup (\bigsqcup_j \overline{y}_j) = \mathbf{1}$ is equivalent to $x \sqcap (\bigsqcap_j y_j) = \bigsqcap_j y_j$, which is strongly preserved by e since e strongly preserves $\sqcap$. This clearly implies the statement. □

Note that Proposition 9 and Proposition 10 imply that ei strongly preserves inner Horn relations. We later also need the following.

Lemma 11. *Let $x_1, \ldots, x_k, y_1, \ldots, y_l \subseteq \mathbb{N}$, where $k \geqslant 1$. Then the following are equivalent.*

1. $e(x_1) \sqcup \cdots \sqcup e(x_k) \sqcup \overline{e(y_1)} \sqcup \cdots \sqcup \overline{e(y_l)} = \mathbf{1}$.
2. *there exists an $i \leqslant k$ such that* $x_i \sqcup (\bigsqcup_j \overline{y}_j) = \mathbf{1}$.

3. *there exists an* $i \leqslant k$ *such that* $e(x_i) \sqcup (\bigsqcup_j \overline{e(y_j)}) = \mathbf{1}$.

For $k = 0$, *we have that* $\bigsqcup_j \overline{y}_{j \leqslant l} = \mathbf{1}$ *if and only if* $\bigsqcup_{j \leqslant l} \overline{e(y_j)} = \mathbf{1}$.

Proof. For the implication from (1) to (2), suppose that there is for every $i \leqslant k$ an $a_i \in \mathbb{N}$ such that $a_i \notin X_i := x_i \sqcup (\bigsqcup_j \overline{y}_j)$. Let c be $G^{-1}(\{a_1, a_2, \ldots, a_k\})$. Then for each $i \leqslant k$, we have that $c \notin e(x_i) \sqcup \bigsqcup_{j \leqslant l} \overline{e(y_j)}$. To see this, first observe that $a_i \in \bigsqcap_{j \leqslant l} y_j \sqcap \overline{x}_i$. Therefore, $\{a_1, \ldots, a_k\} \in \bigsqcap_{j \leqslant l} F(y_j) \sqcap \overline{F(x_i)}$ for all $i \leqslant k$. We conclude that $c \notin e(x_1) \sqcup \cdots \sqcup e(x_k) \sqcup \overline{e(y_1)} \sqcup \cdots \sqcup \overline{e(y_l)}$.

The implication (2) $\Rightarrow$ (3) follows directly from Proposition 10. The implication (3) $\Rightarrow$ (1) is trivial. The second statement is a direct consequence of Proposition 10. $\square$

Proposition 12. *Every Horn-Horn relation is preserved by* e *and* i, *and so in particular by* ei.

Proof. Suppose that R has a Horn-Horn definition ϕ over $\mathfrak{S}$ with variables V. Since R is in particular outer Horn, it is preserved by i by Proposition 9.

Now we verify that R is preserved by e. Let $u \colon V \to \mathcal{P}(\mathbb{N})$ be an assignment that satisfies ϕ. That is, u satisfies at least one literal in each outer clause of ϕ. It suffices to show that the assignment $v \colon V \to \mathcal{P}(\mathbb{N})$ defined by $x \mapsto e(u(x))$ satisfies the same outer literal. Suppose first that the outer literal is positive; because ϕ is Horn-Horn, it is of the form $x \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$ or of the form $\overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$, which is preserved by e by Lemma 11.

Now, suppose that the outer literal is negative, that is, of the form $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} \neq \mathbf{1}$ for some $k \geqslant 0$. We will treat the case $k \geqslant 1$, the other case being similar. Suppose for contradiction that $v(x_1) \sqcup \cdots \sqcup v(x_k) \sqcup \overline{v(y_1)} \sqcup \cdots \sqcup \overline{v(y_l)} = \mathbf{1}$. By Lemma 11, there exists an $i \leqslant k$ such that $u(x_i) \sqcup (\bigsqcup_j u(\overline{y}_j)) = \mathbf{1}$. But then we have in particular that $u(x_1) \sqcup \cdots \sqcup u(x_k) \sqcup u(\overline{y_1}) \sqcup \cdots \sqcup u(\overline{y_l}) = \mathbf{1}$, in contradiction to the assumption that u satisfies ϕ. $\square$

4.3 $\mathcal{EI}$ Set Constraints

In this section we introduce the class of $\mathcal{EI}$ set constraints, show that it strictly contains all Horn-Horn relations, give several examples and non-examples. Then we present an algorithmic reduction from CSPs for $\mathcal{EI}$ set constraints to satisfiability for finite sets of Horn-Horn clauses.

Definition 13. The set of all relations with a quantifier-free definition over $\mathfrak{S}$ that are preserved by the operation ei is denoted by $\mathcal{EI}$.

Remark. Note that the definition of the operation ei (Definition 5) involved a bijection G between $\mathbb{N}$ and Fin*; we will see later (Proposition 36 and Proposition 37) that the class $\mathcal{EI}$ is independent from the precise choice of G.

Recall from Proposition 12 that $\mathcal{EI}$ contains all Horn-Horn relations. We now present examples of relations that are not from $\mathcal{EI}$, and examples of relations that are in $\mathcal{EI}$ but not Horn-Horn.

Example 10. *We give an example of a relation that is clearly* not *from $\mathcal{EI}$. The relation $R = \{(x,y) \mid x \sqcup y = \mathbf{1}\}$ is violated by ei: consider $S_1 = \{2a \mid a \in \mathbb{N}\}$ and $S_2 = \{2a + 1 \mid a \in \mathbb{N}\}$. Then $(S_1, S_2) \in R$, and since i is an isomorphism between $\mathfrak{S}^2$ and $\mathfrak{S}$ we also have that $(i(S_1,S_1), i(S_2,S_2)) \in R$. Since neither $i(S_1,S_1) \sqsubseteq i(S_2,S_2)$ nor $i(S_2,S_2) \sqsubseteq i(S_1,S_1)$, we get that $e(i(S_1,S_1)) \sqcup e(i(S_2,S_2)) \sqsubsetneq e(\mathbf{1}) = \mathbf{1}$ by Proposition 8. Therefore, $(ei(S_1,S_1), ei(S_2,S_2)) \notin R$ which is what we wanted to show.*

Example 11. *The relation $R = \{(x,y,z) \mid (x = y) \vee (y = z)\}$ is also not preserved by ei: note that $(\mathbf{0},\mathbf{1},\mathbf{1}), (\mathbf{0},\mathbf{0},\mathbf{1}) \in R$, but $ei(\mathbf{0},\mathbf{0})$, $ei(\mathbf{1},\mathbf{0})$, and $ei(\mathbf{1},\mathbf{1})$ are pairwise distinct since ei is injective.*

Example 12. *The formula*

$$\begin{aligned} &(x \sqcap y \neq x) \\ \wedge\ &(x \sqcap y \neq y) \\ \wedge\ &(v = \mathbf{1} \ \vee\ u = \mathbf{1} \ \vee\ x \sqcup y \neq \mathbf{1}) \end{aligned}$$

is clearly not Horn-Horn. However, the relation defined by the formula is from $\mathcal{EI}$: if (x_1, y_1, u_1, u_2) und (x_2, y_2, u_2, v_2) are from that relation, then neither $i(x_1,x_2) \sqsubseteq i(y_1,y_2)$ nor $i(y_1,y_2) \sqsubseteq i(x_1,x_2)$. By Proposition 8, $(ei(x_1,x_2), ei(y_1,y_1), ei(u_1,u_2), ei(v_1,v_2))$ satisfies the formula. There is no equivalent Horn-Horn formula, since the formula is not preserved by i.

Example 13. *The formula $((x \sqcup y \neq \mathbf{1}) \vee (u \sqcup v = \mathbf{1})) \wedge (\bar{x} \sqcup y \neq \mathbf{1}) \wedge (x \sqcup \bar{y} \neq \mathbf{1})$ is not Horn-Horn. However, it is preserved by e and by i: the reason is that one of its clauses has the negative literal $x \sqcup y \neq \mathbf{1}$, and the conjuncts $\{\bar{x} \sqcup y \neq \mathbf{1}\}$ and $\{x \sqcup \bar{y} \neq \mathbf{1}\}$. Therefore, for every tuple $t \in R$ the tuple $e(t)$ satisfies $x \sqcup y \neq \mathbf{1}$ and is in R as well. By Proposition 9, R is preserved by i. In this case, the authors suspect that there is no equivalent Horn-Horn formula. More generally, it is open whether there exist formulas that are preserved by e and i, but that are* not *equivalent to a Horn-Horn formula.*

Corollary 14. *The class of Horn-Horn relations is a proper subclass of $\mathcal{EI}$.*

Proof. Proposition 12 shows that $\mathcal{EI}$ contains all Horn-Horn relations. Example 12 shows that the inclusion is strict. □

We prepare now some results that can be viewed as a partial converse of Proposition 12.

Definition 15. A quantifier-free formula ϕ (in the syntactic form described at the end of Section 3) is called *reduced* if if every formula obtained from ϕ by removing an outer literal is not equivalent to ϕ over $\mathfrak{S}$.

We note that a slightly different notion of a reduced formula has been introduced by Bodirsky, Chen, and Pinsker (2010). The variant we are using here is better suited for our purposes.

Lemma 16. *In the structure $\mathfrak{S}$, every quantifier-free formula is equivalent to a reduced formula.*

Proof. It is clear that every quantifier-free formula can be written as a formula ϕ in CNF and in the form as we have discussed it after Theorem 1. We now remove successively outer literals as long as this results in an equivalent formula. ☐

We first prove a partial converse of Proposition 9.

Proposition 17. *Let ϕ be a reduced formula that is preserved by i. Then each outer clause of ϕ is Horn.*

Proof. Let V be the set of variables of ϕ. Assume for contradiction that ϕ contains an outer clause with two positive literals, $t_1 = \mathbf{1}$ and $t_2 = \mathbf{1}$. If we remove the literal $t_1 = \mathbf{1}$ from its clause C, the resulting formula is inequivalent to ϕ, and hence there is an assignment $s_1 \colon V \to \mathcal{P}(\mathbb{N})$ that satisfies none of the literals of C except for $t_1 = \mathbf{1}$. Similarly, there is an assignment $s_2 \colon V \to \mathcal{P}(\mathbb{N})$ that satisfies none of the literals of C except for $t_2 = \mathbf{1}$. By injectivity of i, and since i strongly preserves $c, \sqcap, \sqcup$, and $\mathbf{1}$, the assignment $s \colon V \to \mathcal{P}(\mathbb{N})$ defined by $x \mapsto i(s_1(x), s_2(x))$ does not satisfy the two literals $t_1 = \mathbf{1}$ and $t_2 = \mathbf{1}$. Since i strongly preserves c, $\sqcup$, $\sqcap$, none of the other literals in C is satisfied by those mappings as well, in contradiction to the assumption that ϕ is preserved by i. ☐

Definition 18. Let V be a set of variables, and $s \colon V \to \mathcal{P}(\mathbb{N})$ be a mapping. Then a function from $V \to \mathcal{P}(\mathbb{N})$ of the form $x \mapsto e(s(x))$ is called a *core assignment*.

Lemma 19. *For every quantifier-free formula ϕ there exists a formula ψ such that all inner clauses are inner Horn, and such that ϕ and ψ have the same satisfying core assignments. If ϕ is preserved by ei, then the set of all satisfying core assignments of ψ is closed under ei.*

Proof. Suppose that ϕ has an outer clause C with a positive outer literal $t = \mathbf{1}$ such that t contains an inner clause $c := x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l$ that is not Horn, i.e., $k \geqslant 2$. Then we replace the outer literal $t = \mathbf{1}$ in ϕ by k literals $t_1 = \mathbf{1}, \ldots, t_k = \mathbf{1}$ where t_i is obtained from t by replacing c by $x_i \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l$.

We claim that the resulting formula ϕ' has the same set of satisfying core assignments. Observe that $x_i \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \sqsubseteq c$, and hence $t_i = \mathbf{1}$ implies $t = \mathbf{1}$. An arbitrary satisfying assignment of ϕ' satisfies either one of the positive outer literals $t_i = \mathbf{1}$, in which case that observation shows that it also satisfies ϕ, or it satisfies one of the other outer literals of C, in which case it also satisfies this literal in ϕ. Hence, ϕ' implies ϕ. Conversely, let s be a satisfying core assignment of ϕ. If s satisfies a literal from C other than $t = \mathbf{1}$, then it also satisfies this literal in ϕ', and s satisfies ϕ'. Otherwise, s must satisfy $t = \mathbf{1}$, and hence $s(x_1) \sqcup \cdots \sqcup s(x_k) \sqcup \overline{s(y_1)} \sqcup \cdots \sqcup \overline{s(y_l)} = \mathbf{1}$. Since s is a core assignment, Lemma 11 implies that there exists an $i \leqslant k$ such that $s(x_i) \sqcup \overline{s(y_1)} \sqcup \cdots \sqcup \overline{s(y_l)} = \mathbf{1}$. So s satisfies ϕ'.

Suppose that ϕ has an outer clause C with a negative outer literal $t \neq \mathbf{1}$ such that t contains an inner clause $c := x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l$ that is not Horn, i.e., $k \geqslant 2$. Then we replace the clause C in ϕ by k clauses $C_1, \ldots, C_k$ where C_k is obtained from C by replacing c with $x_i \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l$.

We claim that the resulting formula ϕ' has the same set of satisfying core assignments. Observe that $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$ implies that $x_i \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$, for every $i \leqslant k$. The observation shows that an arbitrary assignment of ϕ is also an assignment of ϕ'.

Conversely, let s be a satisfying core assignment of ϕ'. If s satisfies one of the other literals of C other than $t \neq \mathbf{1}$, then s satisfies ϕ. Otherwise, s must satisfy $x_i \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$ for all $i \leqslant k$, and by Lemma 11 we have that s also satisfies $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$.

We perform these replacements until we obtain a formula ϕ' where all inner clauses are Horn; this formula satisfies the requirements of the first statement of the lemma.

To prove the second statement, let $u, v \colon V \to \mathcal{P}(\mathbb{N})$ be two satisfying core assignments of ϕ'. Since ϕ' and ϕ have the same satisfying core assignments, u and v also satisfy ϕ. Then the mapping $w \colon V \to \mathcal{P}(\mathbb{N})$ given by $x \mapsto ei(u(x), v(x))$ is a core assignment, and because ei preserves ϕ, the mapping w satisfies ϕ. Since ϕ and ϕ' have the same core assignments, w is also a satisfying assignment of ϕ', which proves the statement. □

We now single out a technical condition which guarantees, under some extra condition (see Proposition 21) that formulas satisfying a certain universl algebraic property are Horn-Horn. This will allow us to perform a reduction from a CSP associated to a (finite) set constraint languages from $\mathcal{EI}$ to satisfiability of Horn-Horn clauses.

Definition 20. A quantifier-free formula ϕ (in the syntactic form described at the end of Section 3) is called *strongly reduced* if every formula obtained from ϕ by removing an outer literal does not have the same set of satisfying core assignments over $\mathfrak{S}$.

Proposition 21. *Let ϕ be a strongly reduced formula all of whose inner clauses are Horn. If the set of satisfying core assignments of ϕ is closed under ei, then ϕ is Horn-Horn.*

Proof. Let V be the set of variables of ϕ. It suffices to show that all clauses of ϕ are outer Horn. Assume for contradiction that ϕ contains an outer clause with two positive literals, $t_1 = \mathbf{1}$ and $t_2 = \mathbf{1}$. If we remove the literal $t_1 = \mathbf{1}$ from its clause C, the resulting formula has strictly less satisfying core assignments; this shows the existence of a core assignment $s_1 \colon V \to \mathcal{P}(\mathbb{N})$ that satisfies none of the literals of C except for $t_1 = \mathbf{1}$. Similarly, there exists a core assignment $s_2 \colon V \to \mathcal{P}(\mathbb{N})$ that satisfies none of the literals of C except for $t_2 = \mathbf{1}$. By assumption, the inner clauses of t_1 and t_2 are Horn. We claim that the assignment $s \colon V \to \mathcal{P}(\mathbb{N})$ defined by $x \mapsto ei(s_1(x), s_2(x))$ does not satisfy the clause C. Since ei strongly preserves inner Horn clauses, we have that s does not satisfy $t_1 = \mathbf{1} \vee t_2 = \mathbf{1}$. For the same reasons s does not satisfy any other literals in C; this contradicts the assumption that the satisfying core assignments for ϕ are preserved by ei. □

Satisfiability of Horn-Horn clauses is the computational problem to decide whether, given a finite set S of Horn-Horn clauses, there is a satisfying assignment for S.

Proposition 22. *Let Γ be a finite set constraint language from $\mathcal{EI}$. Then* $\mathrm{CSP}(\Gamma)$ *can be reduced in linear time to satisfiability of Horn-Horn clauses.*

Proof. Let Φ be an instance of $\mathrm{CSP}(\Gamma)$, and let V be the set of variables that appear in Φ. For each constraint $R(x_1, \ldots, x_k)$ from Φ, let ϕ_R be the definition of R over $\mathfrak{S}$. By Lemma 19, there exists a formula ψ_R that has the same satisfying core assignments as ϕ_R and where all inner clauses are Horn; moreover, since ϕ_R is preserved by ei, the lemma asserts that the set of all satisfying core assignments of ψ_R is preserved by ei. We can assume without loss of generality that ψ_R is strongly reduced; this can be seen similarly to Lemma 16. By Proposition 21, the formula ψ_R is Horn-Horn.

Let Ψ be the set of all Horn-Horn clauses of formulas $\psi_R(x_1, \ldots, x_k)$ obtained from constraints $R(x_1, \ldots, x_k)$ in Φ in the described manner. We claim that Φ is a satisfiable instance of CSP(Γ) if and only if Ψ is satisfiable. This follows from the fact that for each constraint $R(x_1, \ldots, x_k)$ in Φ, the formulas ϕ_R and ψ_R have the same satisfying core assignments, and that both ϕ_R and ψ_R are preserved by ei (for ψ_R this follows from Proposition 12), so in particular by the function $x \mapsto ei(x, x)$. $\Box$

Note that in Proposition 22 we reduce satisfiability for $\mathcal{EI}$ to satisfiability for a proper subclass of Horn-Horn set constraints: while for general Horn-Horn set constraints we allow that inner clauses of negative outer literals are not Horn, the reduction only produces Horn-Horn clauses where *all* inner clauses are Horn.

5. Algorithm for Horn-Horn Set Constraints

We present an algorithm that takes as input a set Φ of Horn-Horn clauses and decides satisfiability of Φ over $\mathfrak{S} = (\mathcal{P}(\mathbb{N}); \sqcup, \sqcap, c, \mathbf{0}, \mathbf{1})$ in time quadratic to the length of the input. By Proposition 22, this section will therefore conclude the proof that CSP(Γ) is tractable when all relations in Γ are from $\mathcal{EI}$.

As mentioned in the introduction, our algorithm is based on two procedures, both resolution-like. The inner procedure is essentially the well-known positive unit resolution procedure for Horn-SAT, and the outer procedure is basically an algorithm that has been used in the literature about *independence* in constraint satisfaction (see, e.g., Jonsson & Bäckström, 1998; Koubarakis, 2001; Broxvall et al., 2002; Cohen et al., 2000). Our contribution in this section is the way how to nest these two algorithms to obtain a polynomial-time decision procedure for satisfiability of Horn-Horn clauses.

We start by discussing the first procedure of our algorithm, which we call the *inner resolution algorithm*. As in the case of Boolean positive unit resolution (Dowling & Gallier, 1984) one can implement the procedure Inner-Res such that it runs in linear time in the input size.

Lemma 23. *Let Φ be a finite set of inner Horn clauses. Then the following are equivalent.*

1. $\bigsqcap \Phi = \mathbf{1}$ *is satisfiable over* $\mathfrak{S}$.
2. *Inner-Res*(Φ) *from Figure 1 accepts.*
3. $\bigsqcap \Phi = \mathbf{1}$ *has a solution whose image is contained in* $\{\emptyset, \mathbb{N}\}$.

Proof. It is obvious that $\bigsqcap \Phi = \mathbf{1}$ is unsatisfiable when Inner-Res(Φ) rejects; in fact, for all inner clauses c derived by Inner-Res from Φ, the formula $c = \mathbf{1}$ is logically implied by $\bigsqcap \Phi = \mathbf{1}$. Conversely, if the algorithm accepts then we can set all eliminated variables to $\mathbb{N}$ and all remaining variables to $\emptyset$, which satisfies all clauses: in the removed clauses the positive literal is satisfied, and in the remaining clauses we have at least one negative literal at the final stage of the algorithm, and all clauses with negative literals at the final stage of the algorithm are satisfied. $\Box$

The proof of the previous lemma shows that $\bigsqcap \Phi = \mathbf{1}$ is satisfiable over $\mathfrak{S}$ if and only if $\bigsqcap \Phi = \mathbf{1}$ is satisfiable over the two-element Boolean algebra. As we will see in the following,

```
Inner-Res(Φ)
// Input: A finite set Φ of inner Horn clauses
// Accepts iff ⊓Φ = 1 is satisfiable
During the entire algorithm:
    if Φ contains an empty clause, then reject.
Repeat := true
While Repeat = true do
    Repeat := false
    If Φ contains a positive unit clause {x} then
        Repeat := true
        Remove all clauses where the literal x occurs.
        Remove the literal x̄ from all clauses.
    End if
Loop
Accept
```

Figure 1: Inner Resolution Algorithm.

this holds more generally (and not only for inner Horn clauses). The following should be well-known, and can be shown with the same proof as given by Koppelberg (1989) for the weaker Proposition 2.19 there. We give the proof here for the convenience of the reader.

Fact 24. *Let t_1, t_2 be terms over $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$. Then the following are equivalent:*

1. $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ *is satisfiable over the two-element Boolean algebra;*
2. $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ *is satisfiable over* all *Boolean algebras;*
3. $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ *is satisfiable over some Boolean algebra;*
4. $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ *is satisfiable over some finite Boolean algebra.*

Proof. Obviously, (1) implies (2), and (2) implies (3).

For (3) implies (4), assume that $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ has a satisfying assignment in some Boolean algebra $\mathfrak{C}$. Let $x_1, \ldots, x_n$ be the variables which occur in t_1 or t_2, and let $x_i \mapsto c_i$ be a satisfying assignment. Then $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ is satisfiable in the Boolean sub-algebra $\mathfrak{C}'$ of $\mathfrak{C}$ generated by $\{c_1, \ldots, c_n\}$, and $\mathfrak{C}'$ is finite (it has at most $2^{(2^n)}$ elements).

For (4) implies (1), first note that any finite Boolean algebra is isomorphic to the Boolean algebra $(\mathcal{P}(X); \sqcap, \sqcup, c, \mathbf{0}, \mathbf{1})$ of subsets of some finite set X. If $x \in X$, consider the map $h_x : \mathcal{P}(X) \to \{\mathbf{0}, \mathbf{1}\}$, $h(Y) := \mathbf{1}$ if $x \in Y$, and $h(Y) = \mathbf{0}$ otherwise. Then h_x is a homomorphism of Boolean algebras. In particular, this shows that for every non-zero element a of a finite Boolean algebra $\mathfrak{C}$, there is a homomorphism h from $\mathfrak{C}$ to the two-element Boolean algebra such that $h(a) \neq \mathbf{0}$. Now suppose (4), and assume that $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ has a satisfying assignment in some finite Boolean algebra $\mathfrak{C}$. Let c be the element denoted by t_2 in $\mathfrak{C}$ under this assignment, so $c \neq \mathbf{1}$. Now let h be a homomorphism from $\mathfrak{C}$ to $\{\mathbf{0}, \mathbf{1}\}$ such that $h(\overline{c}) \neq \mathbf{0}$, i.e. $h(c) \neq \mathbf{1}$. By construction, the image of the satisfying assignment under h is a satisfying assignment of $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ in $\{\mathbf{0}, \mathbf{1}\}$. $\square$

The same statement for $t_1 = \mathbf{1}$ instead of $t_1 = \mathbf{1} \wedge t_2 \neq \mathbf{1}$ has been given in Proposition 2.19 (Koppelberg, 1989). Fact 24 has the following consequence that is crucial for the way how we use the inner resolution procedure in our algorithm.

Lemma 25. *Let Ψ be a finite set of inner Horn clauses. The following are equivalent:*

1. *Inner-Res*$(\Psi \cup \{\overline{x}_1, \ldots, \overline{x}_k, y_0, \ldots, y_l\})$ *rejects.*
2. $\bigsqcap \Psi = \mathbf{1}$ *implies that* $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l = \mathbf{1}$ *over* $\mathfrak{S}$.

Proof. $\bigsqcap \Psi = \mathbf{1}$ implies that $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l = \mathbf{1}$ if and only if $\bigsqcap \Psi = \mathbf{1} \wedge x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$ is unsatisfiable over $\mathfrak{S}$. By Fact 24, this is the case if and only if $\bigsqcap \Psi = \mathbf{1} \wedge x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}$ is unsatisfiable over the 2-element Boolean algebra, which is the case if and only if $\bigsqcap \Psi = \mathbf{1} \wedge x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l = \mathbf{0}$ is unsatisfiable over the two-element Boolean algebra. As we have seen in Lemma 23, this in turn holds if and only if Inner-Res$(\Psi \cup \{\overline{x}_1 = \mathbf{1}, \ldots, \overline{x}_k = \mathbf{1}, y_1 = \mathbf{1}, \ldots, y_l = \mathbf{1}\})$ rejects. $\square$

```
Outer-Res(Φ)
// Input: A finite set Φ of Horn-Horn clauses
// Accepts iff Φ is satisfiable over (𝒫(ℕ); ⊓, ⊔, c, 0, 1)
During the entire algorithm:
   if Φ contains an empty clause, then reject.
Repeat := true
While Repeat = true do
   Repeat := false
   Let Ψ be the set of all inner Horn clauses of terms t
      from positive unit clauses {t = 1} in Φ.
      If Inner-Res rejects Ψ, then reject.
   For each negative literal t ≠ 1 in clauses from Φ
      For each inner clause D = {x_1, ..., x_k, ȳ_1, ..., ȳ_l} of t
         Call Inner-Res on
            Ψ ∪ {x̄_1 = 1, ..., x̄_k = 1, y_0 = 1, ..., y_l = 1}
         If Inner-Res rejects then remove clause D from t
      End for
      If all clauses in t have been removed, then
         Remove outer literal t ≠ 1 from its clause
         Repeat := true
   End for
Loop
Accept
```

Figure 2: Outer Resolution Algorithm.

Theorem 26. *The algorithm 'Outer-Res' in Figure 2 decides satisfiability for sets of Horn-Horn clauses in quadratic time.*

Proof. We first argue that if the algorithm rejects Φ, then Φ has indeed no solution. First note that during the whole argument, the set of clauses Φ has the same satisfying tuples (i.e., the corresponding formulas are equivalent): Observe that only negative literals get removed from clauses, and that a negative literal $t \neq \mathbf{1}$ only gets removed from a clause when Inner-Res rejects $\Psi \cup \{\overline{x}_1 = \mathbf{1}, \dots, \overline{x}_k = \mathbf{1}, y_0 = \mathbf{1}, \dots, y_l = \mathbf{1}\}$ for each inner clause $\{x_1, \dots, x_k, \overline{y}_1, \dots, \overline{y}_l\}$ of t. By Lemma 25, if Inner-Res rejects $\Psi \cup \{\overline{x}_1 = \mathbf{1}, \dots, \overline{x}_k = \mathbf{1}, y_0 = \mathbf{1}, \dots, y_l = \mathbf{1}\}$ then Ψ implies that $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l = \mathbf{1}$. Hence, the positive unit clauses imply that $t = \mathbf{1}$ and therefore the literal $t \neq \mathbf{1}$ can be removed from the clause without changing the set of satisfying tuples. Now the algorithm rejects if either Inner-Res rejects Ψ or if it derives the empty clause. In both cases it is clear that Φ is not satisfiable.

Thus, it suffices to construct a solution when the algorithm accepts. Let Ψ be the set of all inner clauses of terms from positive unit clauses at the final stage, when the algorithm accepts. For each remaining negative outer literal $\{t \neq \mathbf{1}\}$ and each remaining inner clause $D = \{x_1, \dots, x_k, \overline{y}_1, \dots, \overline{y}_l\}$ of t there exists an assignment α_D from $V \to \mathcal{P}(\mathbb{N})$ that satisfies $\Psi \cup \{x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y}_1 \sqcup \cdots \sqcup \overline{y}_l \neq \mathbf{1}\}$: otherwise, by Lemma 25, the inner resolution algorithm would have rejected $\Psi \cup \{\overline{x}_1 = \mathbf{1}, \dots, \overline{x}_k = \mathbf{1}, y_0 = \mathbf{1}, \dots, y_l = \mathbf{1}\}$, and would have removed the inner clause D from t. Let $D_1, \dots, D_s$ be an enumeration of all remaining inner clauses D that appear in all remaining negative outer literals.

Write i_s for the s-ary operation defined by $(x_1, \dots, x_s) \mapsto i(x_1, i(x_2, \dots, i(x_{s-1}, x_s) \cdots))$ (where i is as in Fact 7). We claim that $s \colon V \to \mathcal{P}(\mathbb{N})$ given by

$$x \mapsto i_s(\alpha_{D_1}(x), \dots, \alpha_{D_s}(x))$$

satisfies all clauses in Φ. Let C be a clause from Φ. By assumption, at the final stage of the algorithm, the clause C is still non-empty. Also note that since all formulas in the input were Horn-Horn, they contain at most one positive literal. This holds in particular for C, and we therefore only have to distinguish the following cases:

- At the final state of the algorithm, C still contains a negative literal $t \neq \mathbf{1}$. Since $t \neq \mathbf{1}$ has not been removed, there is a remaining inner clause $D = \{x_1, \dots, x_k, \overline{y}_1, \dots, \overline{y}_l\}$ of t. Observe that $s(x_1) \sqcup \cdots \sqcup s(x_k) \sqcup \overline{s(y_1)} \sqcup \cdots \sqcup \overline{s(y_l)} = \mathbf{1}$ if and only if $\alpha_{D_j}(x_1) \sqcup \cdots \sqcup \alpha_{D_j}(x_k) \sqcup \overline{\alpha_{D_j}(y_1)} \sqcup \cdots \sqcup \overline{\alpha_{D_j}(y_l)} = \mathbf{1}$ for all $1 \leqslant j \leqslant s$. Hence, and since $\alpha_D(x_1) \sqcup \cdots \sqcup \alpha_D(x_k) \sqcup \overline{\alpha_D(y_1)} \sqcup \cdots \sqcup \overline{\alpha_D(y_l)} \neq \mathbf{1}$, s satisfies $t \neq \mathbf{1}$. This shows that s satisfies C.

- All negative literals have been removed from C during the algorithm. The positive literal $t_0 = \mathbf{1}$ of C is such that the inner clauses of t_0 are Horn. They will be part of Ψ, and therefore $t_0 = \mathbf{1}$ is satisfied by s. Indeed, by assumption the assignments α_{D_j} satisfy Ψ, and Ψ is preserved by i.

We conclude that s is a solution to Φ. The inner resolution algorithm has a linear time complexity; the outer resolution algorithm performs at most a linear number of calls to the inner resolution algorithm, and it is straightforward to implement all necessary data structures for outer resolution to obtain a running time that is quadratic in the input size. □

Combining Proposition 22 with Theorem 26, we obtain the following.

Theorem 27. *Let Γ be a finite set constraint language from $\mathcal{EI}$. Then* CSP(Γ) *can be solved in quadratic time.*

6. Maximal Tractability

In this section we show that the class $\mathcal{EI}$ is a maximal tractable set constraint language. More specifically, let Γ be a set constraint language that strictly contains all $\mathcal{EI}$ relations. We then show that Γ contains a finite set of relations Γ' such that already the problem CSP(Γ') is NP-hard (Theorem 40).

6.1 The Universal-Algebraic Approach

In our proof we use the so-called *universal-algebraic approach* to the complexity of constraint satisfaction problems, which requires that we re-formulate set CSPs as constraint satisfaction problems for *ω-categorical* structures. For a more detailed introduction to the universal-algebraic approach for ω-categorical structures (see Bodirsky, 2012). A structure Γ with a countable domain is called *ω-categorical* if all countable structures that satisfy the same first-order sentences as Γ are isomorphic to Γ (see, e.g., Hodges, 1993). By the theorem of Ryll-Nardzewski, and for countable signatures, this is equivalent to requiring that every relation that is preserved by the automorphisms[1] of Γ is first-order definable in Γ (see, e.g., Hodges, 1993). A useful consequence of this is that in an ω-categorical structure Γ, whenever two tuples $\overline{c} = (c_1, \ldots, c_n)$ and $\overline{d} = (d_1, \ldots, d_n)$ from Γ satisfy the same first-order formulas, there is an automorphism α of Γ which maps $\overline{c}$ to $\overline{d}$.

An example of an ω-categorical structure is $(\mathbb{Q}; <)$ (by Cantor's theorem), and a non-example is given by $(\mathbb{Z}; <)$. Note that $(\mathbb{Q}; <)$ and $(\mathbb{Z}; <)$ have the same CSP; indeed, any two infinite linear orders share the same CSP, since they even have the same finite substructures. A characterisation of those infinite structures for which there is an ω-categorical structure having the same CSP has been given by Bodirsky, Hils, and Martin (2011). Empirically, it can be observed that constraint satisfaction problems studied in temporal and spatial reasoning are typically called *qualitative* if and only if they can be formulated with an ω-categorical template.

Set constraint languages are in general not ω-categorical (this follows easily by the mentioned theorem of Ryll-Nardzewski). However, every set CSP can be formulated as the CSP of an ω-categorical structure. To see this, we first have to recall some basic facts about Boolean algebras. All countable atomless[2] Boolean algebras are isomorphic (Koppelberg, 1989, Corollary 5.16; see also Hodges, 1993, Example 4 on page 100). Let $\mathfrak{A}$ denote such a countable atomless Boolean algebra, and let $\mathbb{A}$ denote the domain of $\mathfrak{A}$. Again, we use $\sqcap$ and $\sqcup$ to denote join and meet in $\mathfrak{A}$, respectively. Since the axioms of Boolean algebras and the property of not having atoms can all be written as first-order sentences, it follows that $\mathfrak{A}$ is ω-categorical. A structure $\mathfrak{B}$ has *quantifier elimination* if every first-order formula is over $\mathfrak{B}$ equivalent to a quantifier-free formula. It is well-known that $\mathfrak{A}$ has quantifier elimination (see Hodges, 1993, Exercise 17 on page 391). We will also make use of the following.

Theorem 28 (Marriott & Odersky, 1996, Corollary 5.7)**.** *A quantifier-free formula is satisfiable in some infinite Boolean algebra if and only if it is satisfiable in all infinite Boolean algebras.*

1. An isomorphism of a structure Γ with itself is called an *automorphism* of Γ.
2. An *atom* in a Boolean algebra is an element $x \neq \mathbf{0}$ such that for all y with $x \cap y = y$ and $x \neq y$ we have $y = \mathbf{0}$. If a Boolean algebra does not contains atoms, it is called *atomless*.

In particular, when $\mathfrak{B}$ is an infinite Boolean algebra and $\phi_1, \ldots, \phi_n$ are quantifier-free formulas over the signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$, and when Γ is the relational structure with signature $\{R_1, \ldots, R_n\}$ where R_i is for each $i \leqslant n$ defined by ϕ_i over $\mathfrak{B}$, then $\mathrm{CSP}(\Gamma)$ *does not depend on the choice of* $\mathfrak{B}$.

A fundamental concept in the complexity theory of constraint satisfaction problems is the notion of *primitive positive definitions.* A first-order formula is called *primitive positive (pp)* if it is of the form

$$\exists x_1, \ldots, x_n \, (\psi_1 \wedge \cdots \wedge \psi_m)$$

where for each $i \leqslant m$ the formula ψ_i is of the form $R(y_1, \ldots, y_l)$ or of the form $y_1 = y_2$, and where R is a relation symbol and $y_1, y_2, \ldots, y_l$ are either free variables or from $\{x_1, \ldots, x_n\}$. We say that a k-ary relation $R \subseteq D^k$ is *primitive positive definable (pp definable)* over a τ-structure Γ with domain D iff there exists a primitive positive formula $\phi(x_1, \ldots, x_k)$ with the k free variables $x_1, \ldots, x_k$ such that a tuple $(b_1, \ldots, b_k)$ is in R if and only if $\phi(b_1, \ldots, b_k)$ is true in Γ.

Example 14. *The relation $\{(x, y) \in \mathcal{P}(\mathbb{N})^2 \mid x \sqsubset y\}$ is pp definable in $(\mathcal{P}(\mathbb{N}); S, \neq)$ where $S = \{(x, y, z) \mid x \sqcap y \sqsubseteq z\}$. The pp definition is $S(x, x, y) \wedge x \neq y$ (the definition is even quantifier-free).* □

Example 15. *The relation $\{(x_1, x_2, x_3, y) \in \mathcal{P}(\mathbb{N})^4 \mid x_1 \sqcap x_2 \sqcap x_3 \sqsubseteq y\}$ is pp definable in $(\mathcal{P}(\mathbb{N}); S)$ where $S = \{(x, y, z) \mid x \sqcap y \sqsubseteq z\}$. The pp definition is $\exists u \, (S(x_1, x_2, u) \wedge S(u, x_3, y))$.* □

When every relation of a structure Γ is preserved by an operation f, then f is called a *polymorphism* of Γ. Note that polymorphisms of Γ also preserve all relations that have a pp definition in Γ. The following has been shown for finite domain constraint satisfaction by Bulatov et al. (2005); the easy proof also works for infinite domain constraint satisfaction.

Lemma 29. *Let R be a relation with a primitive positive definition in a structure Γ. Then* $\mathrm{CSP}(\Gamma)$ *and the CSP for the expansion of Γ by the relation R are polynomial-time equivalent.*

The following theorem is one of the reasons why it is useful to work with ω-categorical templates (when this is possible).

Theorem 30 (Bodirsky & Nešetřil, 2006)**.** *Let Γ be an ω-categorical structure. Then R is primitive positive definable in Γ if and only if R is preserved by all polymorphisms of Γ.*

The previous and the next result together can be used to translate questions about primitive positive definability into purely operational questions. Let D be a set, let $\mathcal{O}^{(n)}$ be $D^n \to D$, and let $\mathcal{O}$ be $\bigcup_{n=1}^{\infty} \mathcal{O}^{(n)}$ the set of operations on D of finite arity. An operation $\pi \in \mathcal{O}^{(n)}$ is called a *projection* if for some fixed $i \in \{1, \ldots, n\}$ and for all n-tuples $(x_1, \ldots, x_n) \in D^n$ we have the identity $\pi(x_1, \ldots, x_n) = x_i$. The *composition* of a k-ary operation f and k operations $g_1, \ldots, g_k$ of arity n is the n-ary operation defined by

$$\begin{aligned} & (f(g_1, \ldots, g_k))(x_1, \ldots, x_n) \\ = \quad & f\big(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)\big). \end{aligned}$$

Definition 31. We say that $\mathcal{F} \subseteq \mathcal{O}$ *locally generates* $f: D^n \to D$ if for every finite subset A of D there is an operation $g: D^n \to D$ that can be obtained from the operations in $\mathcal{F}$ and projection maps by composition such that $f(a) = g(a)$ for all $a \in A^n$.

Theorem 32 (see Szendrei, 1986, Corollary 1.9; also Bodirsky, 2012, Proposition 5.2.1)**.** *Let $\mathcal{F} \subseteq \mathcal{O}$ be a set of operations with domain D. Then an operation $f: D^k \to D$ preserves all finitary relations that are preserved by all operations in $\mathcal{F}$ if and only if $\mathcal{F}$ locally generates f.*

The set of all automorphisms of a structure Γ is denoted by $\mathrm{Aut}(\Gamma)$. In the following, we always consider sets of operations $\mathcal{F}$ that contain $\mathrm{Aut}(\mathfrak{A})$, and therefore make the following convention. For $\mathcal{F} \subseteq \mathcal{O}$, we say that $\mathcal{F}$ *generates* $f \in \mathcal{O}$ if $\mathcal{F} \cup \mathrm{Aut}(\mathfrak{A})$ locally generates f.

6.2 $\mathcal{EI}$ Set Constraints over the Atomless Boolean Algebra

In the previous subsection we have seen that all set CSPs can be formulated as CSPs for ω-categorical structures. In this section, we describe those ω-categorical templates that correspond to set CSPs for $\mathcal{EI}$ set constraints. In order to do so, we define analogs of the operations e and i, defined on $\mathbb{A}$ instead of $\mathcal{P}(\mathbb{N})$.

Proposition 33. *There is an isomorphism $\tilde{i}$ between $\mathfrak{A}^2$ and $\mathfrak{A}$.*

Proof. It is straightforward to verify that $\mathfrak{A}^2$ is again a countable atomless Boolean algebra. □

Motivated by the properties of e described in Lemma 11, we make the following definition.

Definition 34. Let $\mathfrak{B}$ and $\mathfrak{B}'$ be two arbitrary Boolean algebras with domains B and B', respectively, and let $g: B \to B'$ be a function that strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$. We say that g *forgets unions* if for all $k \geqslant 1$, $l \geqslant 0$, and $x_1, \ldots, x_k, y_1, \ldots, y_l \in B$ we have

$$e(x_1) \sqcup \cdots \sqcup e(x_k) \sqcup \overline{e(y_1)} \sqcup \cdots \sqcup \overline{e(y_l)} = \mathbf{1}$$

if and only if there exists an $i \leqslant k$ such that $x_i \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$.

Proposition 35. *There exists an injection $\tilde{e}: \mathbb{A} \to \mathbb{A}$ that strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$ in $\mathfrak{A}$, and that forgets unions.*

Proof. The construction of $\tilde{e}$ is by a standard application of König's tree lemma for ω-categorical structures (see, e.g., Bodirsky & Dalmau, 2012, Lemma 2); it suffices to show that there is an injection f from every finite induced substructure $\mathfrak{B}$ of $\mathfrak{A}$ to $\mathfrak{A}$ such that f strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and forgets unions.

So let $\mathfrak{B}$ be such a finite substructure of $\mathfrak{A}$, and let B be the domain of $\mathfrak{B}$. Let $\mathfrak{C} = (\mathcal{P}(B); \sqcap, \sqcup, c, \mathbf{0}, \mathbf{1})$ be the Boolean algebra of the subsets of B. We claim that $g: B \to \mathcal{P}(B)$ given by $g(\mathbf{1}) = \mathbf{1}$ and $g(x) = \{z \mid z \neq \mathbf{0} \wedge z \sqsubseteq^{\mathfrak{B}} x\}$ for $x \neq \mathbf{1}$

- preserves $\mathbf{0}$ and $\mathbf{1}$: this is by definition;

- preserves $\sqcap$: for $x, y \in B$ (including the case that $x = \mathbf{1}$ or $y = \mathbf{1}$) we have

$$\begin{aligned} g(x) \sqcap^{\mathfrak{C}} g(y) &= \{z \mid z \neq \mathbf{0} \wedge z \sqsubseteq^{\mathfrak{B}} x \wedge z \sqsubseteq^{\mathfrak{B}} y\} \\ &= \left\{z \mid z \neq \mathbf{0} \wedge z \sqsubseteq^{\mathfrak{B}} (x \sqcap^{\mathfrak{B}} y)\right\} \\ &= g(x \sqcap^{\mathfrak{B}} y) \; ; \end{aligned}$$

- is injective: if $x, y \in B$ such that $g(x) = g(y)$, then $x \sqsubseteq^{\mathfrak{B}} y$ and $y \sqsubseteq^{\mathfrak{B}} x$, and hence $x = y$;

- strongly preserves $\sqcap$: this follows from the previous two items;

- forgets unions: this can be shown analogously to the proof of Lemma 11.

  Indeed, one has $x_i \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$ iff $x_i \sqsupseteq^{\mathfrak{B}} \bigsqcap_j y_j$ iff $x_i \sqcap \bigsqcap_j y_j = \bigsqcap_j y_j$ iff $g(x_i) \sqcap \bigsqcap_j g(y_j) = \bigsqcap_j g(y_j)$ iff $g(x_i) \sqcup \overline{g(y_1)} \sqcup \cdots \sqcup \overline{g(y_l)} = \mathbf{1}$. Thus, $x_i \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$ for some $1 \leqslant i \leqslant k$ implies $g(x_1) \sqcup \cdots \sqcup g(x_k) \sqcup \overline{g(y_1)} \sqcup \cdots \sqcup \overline{g(y_l)} = \mathbf{1}$.

  To prove the converse, we use that the finite Boolean algebra $\mathfrak{B}$ may be identified with $(\mathcal{P}(A); \sqcap, \sqcup, c, \mathbf{0}, \mathbf{1})$ for some finite set A. If $X_i := x_i \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} \neq \mathbf{1}$ for $i = 1, \ldots, k$, we may choose $a_i \in A \backslash X_i$, i.e. $a_i \in \bigsqcap_j y_j \sqcap \overline{x_i}$, for $i = 1, \ldots, k$. Let $C := \{a_1, \ldots, a_k\} \subseteq A$, so $C \in B$. By construction, for $i \leqslant k$ one has $\{C\} \notin g(x_i) \sqcup \overline{g(y_1)} \sqcup \cdots \sqcup \overline{g(y_l)}$. In particular, it follows that $g(x_1) \sqcup \cdots \sqcup g(x_k) \sqcup \overline{g(y_1)} \sqcup \cdots \sqcup \overline{g(y_l)} \neq \mathbf{1}$.

Clearly, there is an embedding h from $\mathfrak{C}$ into $\mathfrak{A}$. Then $f := h(g)$ is a homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$ that forgets unions. □

Proposition 36. *Let ϕ be a quantifier-free formula over the signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$. Then e preserves ϕ over $\mathfrak{S}$ if and only if $\tilde{e}$ preserves ϕ over $\mathfrak{A}$. Moreover, every operation from $\mathbb{A} \to \mathbb{A}$ that strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$ and forgets unions generates $\tilde{e}$, and is generated by $\tilde{e}$.*

Proof. Let $\bar{a}$ be a tuple of elements from $\mathbb{A}$. Clearly, there exists a tuple $\bar{b}$ of elements from $\mathcal{P}(\mathbb{N})$ such that $\bar{a}$ and $\bar{b}$ satisfy the same set ψ of quantifier-free formulas; this follows from the fact that every finite Boolean algebra is the Boolean algebra of subsets of a finite set. Now observe that whether or not the tuple $e(\bar{b})$ satisfies a quantifier-free formula ϕ only depends on ψ, by Lemma 11. Since $\tilde{e}$ strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and forgets unions, the same is true for the quantifier-free formulas that hold on $\tilde{e}(\bar{a})$. Hence, $\tilde{e}$ preserves ϕ over $\mathfrak{A}$ if and only if e preserves ϕ over $\mathfrak{S}$.

To prove the second part of the statement, we use Theorem 32. Suppose that $\bar{c}$ and $\bar{d}$ are tuples (of the same length) of elements from $\mathbb{A}$ that satisfy the same quantifier-free formulas. Since $\mathfrak{A}$ has quantifier-elimination, it follows that $\bar{c}$ and $\bar{d}$ satisfy the same first-order formulas in $\mathfrak{A}$. By the consequence of the theorem of Ryll-Nardzewski mentioned at the beginning of Section 6.1, there exists an automorphism α of $\mathfrak{A}$ that maps $\bar{c}$ to $\bar{d}$. By the above observations and Theorem 32, this implies that all operations that strongly preserve $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and forget unions generate each other. □

Let $\widetilde{ei}$ be the operation $(x, y) \mapsto \tilde{e}(\tilde{i}(x, y))$.

Proposition 37. *Let ϕ be a quantifier-free formula over the signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$. Then ei preserves ϕ over $\mathfrak{S}$ if and only if $\widetilde{ei}$ preserves ϕ over $\mathfrak{A}$. Moreover, every binary operation g that strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and forgets unions generates $\widetilde{ei}$, and is generated by $\widetilde{ei}$.*

Proof. The arguments are similar to the ones given in the proof of Proposition 36. If $\bar{a}_1$ and $\bar{a}_2$ are n-tuples of elements from $\mathbb{A}$, there are n-tuples $\bar{b}_1, \bar{b}_2$ of elements from $\mathcal{P}(\mathbb{N})$ such that $(\bar{a}_1, \bar{a}_2)$ and $(\bar{b}_1, \bar{b}_2)$ satisfy the same set ψ of quantifier-free formulas. Whether or not $ei(\bar{b}_1, \bar{b}_2)$ satisfies a quantifier-free formula ϕ only depends on ψ, as ei strongly preserves $\sqcap$, $\mathbf{0}$, and $\mathbf{1}$, and forgets unions. The same holds for $\widetilde{ei}(\bar{a}_1, \bar{a}_2)$, and so $\widetilde{ei}$ preserves ϕ over $\mathfrak{A}$ if and only if ei preserves ϕ over $\mathfrak{S}$.

The proof of the second part of the statement is identical to the one for Proposition 36. □

6.3 The Central Argument

We now give the central argument for the maximal tractability of $\mathcal{EI}$, stated in universal-algebraic language. We say that an operation from $\mathbb{A}^k \to \mathbb{A}$ *depends* on the argument $i \in \{1, \ldots, k\}$ if there is no $(k-1)$-ary operation f' such that for all $x_1, \ldots, x_k \in \mathbb{A}$

$$f(x_1, \ldots, x_k) = f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k) .$$

We can equivalently characterize k-ary operations that depend on the i-th argument by requiring that there are $x_1, \ldots, x_k \in \mathbb{A}$ and $x_i' \in \mathbb{A}$ such that

$$f(x_1, \ldots, x_k) \neq f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k) .$$

The following is a general fact about injective maps.

Lemma 38. *Let $f \colon \mathbb{A}^k \to \mathbb{A}$ be a function that depends on all arguments, and which is locally generated by a set of injective operations $\mathcal{F}$. Then f is injective.*

Proof. We first prove that every term $T(x_1, \ldots, x_n)$ formed from operations from $\mathcal{F}$ over the variables $x_1, \ldots, x_n$ such that every variable appears at least once defines an injective map. We prove this by induction over the term structure. In case that $n = 1$ and T is just x_1 there is nothing to show. Otherwise, T has the form $f(T_1, \ldots, T_k)$ for a k-ary $f \in \mathcal{F}$ such that $T_j = T_j(x_{i_1}, \ldots, x_{i_{m(j)}})$ is for all $j \leqslant k$ a term over operations from $\mathcal{F}$ with variables $x_{i_1}, \ldots, x_{i_{m(j)}}$ each of which appears at least once in T_j. Now suppose that $a_1, \ldots, a_n \in \mathbb{A}$ and $b_1, \ldots, b_n \in \mathbb{A}$ are such that $T(a_1, \ldots, a_n) = T(b_1, \ldots, b_n)$. We want to show that $a_i = b_i$ for all $i \leqslant n$. Since f is injective we must have that $T_j(a_{i_1}, \ldots, a_{i_{m(j)}}) = T_j(b_{i_1}, \ldots, b_{i_{m(j)}})$ for all $j \leqslant k$. Since every variable from $x_1, \ldots, x_n$ appears in T at least once, the variable x_i must appear in T_j, for some $j \leqslant k$. Since T_j defines an injective operation by inductive assumptions, we must have that $a_i = b_i$. It follows that T defines an injective map.

Now suppose that f is an operation that is locally generated by $\mathcal{F}$ and depends on all arguments. Thus, for each i there are $c_1^i, \ldots, c_n^i$ and d_i such that $f(c_1^i, \ldots, c_n^i) \neq f(c_1^i, \ldots, c_{i-1}^i, d_i, c_{i+1}^i, \ldots, c_n^i)$. Let $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{A}$ be such that $f(a_1, \ldots, a_n) = f(b_1, \ldots, b_n)$. We have to show that $a_1 = b_1, \ldots, a_n = b_n$. Since f is locally generated by $\mathcal{F}$, there exists a term $T(x_1, \ldots, x_n)$ composed from the variables $x_1, \ldots, x_n$ and operations from $\mathcal{F}$ such that $T(e_1, \ldots, e_n) = f(e_1, \ldots, e_n)$ for all elements $e_1, \ldots, e_n$ from the set $\{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1^1, \ldots, c_n^n, d_1, \ldots, d_n\}$. For all i, the variable x_i must appear in $T(x_1, \ldots, x_n)$ because $T(c_1^i, \ldots, c_n^i) \neq T(c_1^i, \ldots, c_{i_1}^i, d_i, c_{i+1}^i, \ldots, c_n^i)$. Hence, the argument from the beginning of the proof shows that $T(x_1, \ldots, x_n)$ defines an injective map, and therefore that $a_1 = b_1, \ldots, a_n = b_n$. We have shown that f is injective. ☐

Theorem 39. *Let f be an operation generated by $\{\widetilde{ei}\}$. Then either $\{f\}$ generates $\widetilde{ei}$, or f is generated by $\{\tilde{e}\}$.*

Proof. To show the statement of the theorem, let f be a k-ary operation generated by $\{\widetilde{ei}\}$. For the sake of notation, let $x_1, \ldots, x_l$ be the arguments on which f depends, for $l \leqslant k$. Let $f' \colon \mathbb{A}^l \to \mathbb{A}$ be the operation given by $f'(x_1, \ldots, x_l) = f(x_1, \ldots, x_{l-1}, x_l, x_l, \ldots, x_l)$. Observe that f' depends on all arguments, and is locally generated by injective operations; by Lemma 38, f' is injective. Since f' is generated by operations that preserve $\mathbf{0}$, $\mathbf{1}$, and $\sqcap$, also f' preserves them. As f' is injective, it even strongly preserves $\mathbf{0}$, $\mathbf{1}$, and $\sqcap$.

Consider first the case that $l = 1$, i.e., f' is unary. If for all finite subsets of $\mathbb{A}$, the operation f' equals an automorphism of $\mathfrak{A}$, then f is generated by $\mathrm{Aut}(\mathfrak{A})$ and there is nothing to show. So assume otherwise; that is, assume that there is a finite set $S \subseteq \mathbb{A}$ such that there is no $a \in \mathrm{Aut}(\mathfrak{A})$ with $f'(x) = a(x)$ for all $x \in S$. We claim that f' forgets unions. To see this, let $u_1, \ldots, u_m, v_1, \ldots, v_n$ be from $\mathbb{A}$ such that $f'(u_1) \sqcup \cdots \sqcup f'(u_m) \sqcup \overline{f'(v_1)} \sqcup \cdots \sqcup \overline{f'(v_n)} = \mathbf{1}$. Since f' is generated by $\{\widetilde{ei}\}$, there is a term $T(x)$ composed from $\widetilde{ei}$, the automorphisms of $\mathfrak{A}$, and a single variable x such that $f'(x) = T(x)$ for all $x \in S \cup \{u_1, \ldots, u_m, v_1, \ldots, v_n\}$. By the choice of S, this term cannot be composed of automorphisms alone, and hence there must be $a \in \mathrm{Aut}(\mathfrak{A})$ and operational terms T_1, T_2 composed from automorphisms of $\mathfrak{A}$ and $\widetilde{ei}$ such that $f'(x) = a(\widetilde{ei}(T_1(x), T_2(x)))$ for all $x \in S$. As $\widetilde{ei}$ forgets unions, there exists an $i \leqslant k$ such that $T_1(u_i) \sqcup \overline{T_1(v_1)} \sqcup \cdots \sqcup \overline{T_l(v_n)} = \mathbf{1}$. Since T_1 strongly preserves $\sqcap$, this means that $u_i \sqcup \overline{v}_1 \sqcup \cdots \sqcup \overline{v}_n = \mathbf{1}$ (see the proof of Proposition 10), which is what we wanted to show. By Proposition 36 it follows that f' is generated by $\tilde{e}$. But then f is generated by $\tilde{e}$ as well.

Next, consider the case that $l > 1$. Let g be the binary operation defined by $g(x, y) = f'(x, y, \ldots, y)$. This functions depends on both arguments, and so cannot be generated by the automorphisms of $\mathfrak{A}$ alone. Hence, there is a term of the form

$$T(x, y) = a(\widetilde{ei}(T_1(x, y), T_2(x, y)))$$

where

- $a \in \mathrm{Aut}(\mathfrak{A})$,
- T_1 and T_2 are operational terms composed from $\widetilde{ei}$, the automorphisms of $\mathfrak{A}$, and the two variables x and y,
- $g(x, y) = T(x, y)$ for all $(x, y) \in \{u_1, \ldots, u_m, v_1, \ldots, v_n\}$.

We claim that g forgets unions. Assume $g(u_1) \sqcup \cdots \sqcup g(u_m) \sqcup \overline{g(v_1)} \sqcup \cdots \sqcup \overline{g(v_n)} = \mathbf{1}$ for some elements $u_1 = (u_1^1, u_1^2), \ldots, u_m = (u_m^1, u_m^2), v_1 = (v_1^1, v_1^2), \ldots, v_n = (v_n^1, v_n^2)$ from $\mathbb{A}^2$. Since $\widetilde{ei}$ forgets unions, there exists an $i \leqslant k$ such that $T_1(u_i) \sqcup \overline{T_1(v_1)} \sqcup \cdots \sqcup \overline{T_1(v_n)} = \mathbf{1}$ and

$T_2(u_i) \sqcup \overline{T_2(v_1)} \sqcup \cdots \sqcup \overline{T_2(v_n)} = \mathbf{1}$. Suppose first that T_1 depends on both arguments. Then T_1 defines an injective operation and strongly preserves $\sqcap$. It follows that $u_i \sqcup \overline{v}_1 \sqcup \cdots \sqcup \overline{v}_n = \mathbf{1}$ in $\mathfrak{A}^2$ since these equations are inner Horn. We can argue similarly if T_2 depends on both arguments, and in those cases we have established that g forgets unions. Suppose now that each of T_1 and T_2 does *not* depend on both arguments. Consider first the case that T_1 only depends on the first argument. Then the function $x \mapsto T_1(x, x)$ is injective and strongly preserves $\sqcap$, and from $T_1(u_i) \sqcup \overline{T_1(v_1)} \sqcup \cdots \sqcup \overline{T_1(v_n)} = \mathbf{1}$ we derive as above that $u_i^1 \sqcup \overline{v_1^1} \sqcup \cdots \sqcup \overline{v_n^1} = \mathbf{1}$ holds in $\mathfrak{A}$. In this case, T_2 must depend on the second argument, since T depends on both arguments. We therefore also have that $u_i^2 \sqcup \overline{v_1^2} \sqcup \cdots \sqcup \overline{v_n^2} = \mathbf{1}$ holds in $\mathfrak{A}$. The situation that T_1 only depends on the second argument and T_2 only depends on the first argument is analogous. So g forgets unions. By Proposition 37, g generates $\widetilde{ei}$. Consequently, also f generates $\widetilde{ei}$. □

Theorem 40. *Let Γ be a set constraint language. Suppose that Γ contains all relations from $\mathcal{EI}$, and also contains a relation that is not from $\mathcal{EI}$. Then there is a finite sublanguage Γ' of Γ such that* CSP(Γ') *is NP-hard.*

Proof. When $R_1, R_2, \ldots$ are the relations of Γ, let $\phi_1, \phi_2, \ldots$ be quantifier-free formulas that define $R_1^\Gamma, R_2^\Gamma, \ldots$ over $\mathfrak{S} = (\mathcal{P}(\mathbb{N}); \sqcup, \sqcap, c, \mathbf{0}, \mathbf{1})$. Let $R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \ldots$ be the relations defined by $\phi_1, \phi_2, \ldots$ over $\mathfrak{A}$, and let Δ be the relational structure with domain $\mathbb{A}$ and exactly those relations. By Proposition 37, Δ contains a relation that is not preserved by $\widetilde{ei}$, and contains all relations that are preserved by $\widetilde{ei}$. Consider the set $\mathcal{F}$ of all polymorphisms of Δ. By Theorem 32, all operations in $\mathcal{F}$ are locally generated by $\tilde{e}i$.

The set $\mathcal{F}$ does not contain $\tilde{e}i$, since this would contradict by Theorem 32 the fact that Δ contains a relation that is not preserved by $\widetilde{ei}$. Since $\mathcal{F}$ is locally closed, it follows from Theorem 39 that all operations $f \in \mathcal{F}$ are generated by $\tilde{e}$. But then the relation $\{(x, y, z) \mid x = y \neq z \vee x \neq y = z\}$ is preserved by all operations in $\mathcal{F}$ (we have already seen this relation in Example 5), and hence pp definable in Δ by Theorem 30. This relation has an NP-complete CSP (Bodirsky & Kára, 2008). Let Δ' be the reduct of Δ that contains exactly the relations that appear in the pp definition of $\{(x, y, z) \mid x = y \neq z \vee x \neq y = z\}$ in Δ. Clearly, there are finitely many such relations; we denote the corresponding relation symbols by $\tau' \subset \tau$. By Lemma 29, CSP(Δ') is NP-hard.

This establishes also the hardness of CSP(Γ): let Γ' be the τ'-reduct of Γ. We claim that CSP(Γ') and CSP(Δ') are the same computational problem. We have to show that a conjunction of atomic τ'-formulas Φ is satisfiable in Γ' if and only if it is true in Δ'. Replacing each atomic τ'-formula in Φ by its quantifier-free definition, this follows from Theorem 28. □

7. Concluding Remarks

We have introduced the powerful set constraint language of $\mathcal{EI}$ set constraints, which in particular contains all Horn-Horn set constraints and all previously studied tractable set constraint languages. Constraint satisfaction problems over $\mathcal{EI}$ can be solved in polynomial – even quadratic – time. Our tractability result is complemented by a complexity result which shows that tractability of $\mathcal{EI}$ set constraints is best-possible within a large class of set constraint languages.

We would also like to remark that there is an algorithm to test whether a given finite set constraint language (where relations in the language are given by quantifier-free formulas over the signature $\{\sqcup, \sqcap, \mathbf{c}, \mathbf{0}, \mathbf{1}\}$) is contained in $\mathcal{EI}$. This means that the so-called *meta-problem* for $\mathcal{EI}$ set constraints can be decided effectively.

Proposition 41. *There is an algorithm to test whether a given quantifier-free formula over the signature $\{\sqcup, \sqcap, \mathbf{c}, \mathbf{0}, \mathbf{1}\}$ defines a relation from $\mathcal{EI}$.*

Proof. It is clear that ϕ can be effectively transformed into the normal form that is described in Section 3, so we will from now on assume that ϕ is a conjunction of outer clauses, and that each atomic formula is of the form $t = \mathbf{1}$ where t is in inner conjunctive normal form. Let n be the number of variables of ϕ. We have to test that for any two n-tuples u_1, u_2 of elements of $\mathcal{P}(\mathbb{N})$ that satisfy ϕ, the n-tuple $ei(u_1, u_2)$ satisfies ϕ as well. Note that whether or not a tuple satisfies ϕ in $\mathfrak{S}$ only depends on the Boolean algebra generated by the entries of this tuple in $\mathfrak{S}$. Any Boolean algebra generated by n elements is of size at most 2^{2^n}; therefore, there are finitely many cases to check. For each pair of Boolean algebras with generating tuples u_1, u_2, we check whether $ei(u_1, u_2)$ satisfies ϕ as follows. By Lemma 11, $ei(u_1, u_2)$ satisfies an atomic formula $t = \mathbf{1}$ if and only if for every inner clause $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{e(y_1)} \sqcup \cdots \sqcup \overline{e(y_1)}$ of t there exists an $i \leqslant k$ such that $i(u_1, u_2)$ satisfies $x_i \sqcup \bigsqcup_j \overline{y}_j = \mathbf{1}$. This in turn is true if and only if both u_1 and u_2 satisfy $x_i \sqcup \bigsqcup_j \overline{y}_j = \mathbf{1}$. The truth value of non-atomic formulas of the tuple $ei(u_1, u_2)$ can then be computed from the truth value of the atomic formulas in the usual way. $\square$

Finally we would also like to remark that one can analogously obtain tractability for the class of constraints where the inner clauses of the positive outer literals are *dual Horn* (i.e., have at most one *negative* literal). All statements and proofs for the respective result can be obtained by *dualizing* in the following formal sense: the *dual* of a relation R that is definable over a Boolean algebra is the relation $\{\mathbf{c}(t) \mid t \in R\}$. The *dual* of a k-ary operation f on the same domain is the operation $(x_1, \ldots, x_k) \mapsto \mathbf{c}(f(\mathbf{c}(x_1), \ldots, \mathbf{c}(x_k)))$. The proofs then translate literally into proofs for the dualized versions of the statements.

Acknowledgments

An extended abstract of this article appeared in the proceedings of IJCAI'11 (Bodirsky, Hils, & Krimkevitch, 2011)[3]. We want to thank François Bossière who pointed out mistakes in the conference version of the paper. One mistake concerned the reduction from the CSP for languages from $\mathcal{EI}$ to satisfiability of Horn-Horn clauses; the other concerned a problem in a previous proof of Theorem 26.

Manuel Bodirsky has received funding from the ERC under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 257039).

3. The third author of the conference version left the author team for the preparation of the journal version.

References

Aiken, A. (1994). Set constraints: Results, applications, and future directions. In *Proceedings of the Second Workshop on the Principles and Practice of Constraint Programming*, pp. 326–335.

Baader, F. (2003). Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 319–324.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the EL envelope. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 364–369.

Barto, L., & Kozik, M. (2009). Constraint satisfaction problems of bounded width. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 595–603.

Bodirsky, M. (2012). Complexity classification in infinite-domain constraint satisfaction. Memoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856.

Bodirsky, M., Chen, H., & Pinsker, M. (2010). The reducts of equality up to primitive positive interdefinability. *Journal of Symbolic Logic*, *75*(4), 1249–1292.

Bodirsky, M., & Dalmau, V. (2012). Datalog and constraint satisfaction with infinite templates. *To appear in the Journal on Computer and System Sciences.* A preliminary version appeared in the proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS'05).

Bodirsky, M., Hils, M., & Krimkevitch, A. (2011). Tractable set constraints. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 510–515.

Bodirsky, M., Hils, M., & Martin, B. (2011). On the scope of the universal-algebraic approach to constraint satisfaction. *To appear in Logical Methods in Computer Science (LMCS)*, 90–99. Available at arXiv:0909.5097v3. An extended abstract that announced some of the results appeared in the proceedings of Logic in Computer Science (LICS'10).

Bodirsky, M., & Kára, J. (2008). The complexity of equality constraint languages. *Theory of Computing Systems*, *3*(2), 136–158. A conference version appeared in the proceedings of Computer Science Russia (CSR'06).

Bodirsky, M., & Kára, J. (2009). The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, *57*(2), 1–41. An extended abstract appeared in the Proceedings of the Symposium on Theory of Computing (STOC'08).

Bodirsky, M., & Kutz, M. (2007). Determining the consistency of partial tree descriptions. *Artificial Intelligence*, *171*, 185–196.

Bodirsky, M., & Nešetřil, J. (2006). Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, *16*(3), 359–373.

Boole, G. (1847). *An Investigation of the Laws of Thought.* Walton, London. Reprinted by Philisophical Library, New York, 1954.

Broxvall, M., Jonsson, P., & Renz, J. (2002). Disjunctions, independence, refinements. *Artificial Intelligence*, *140*(1/2), 153–173.

Bulatov, A. A. (2003). Tractable conservative constraint satisfaction problems. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pp. 321–330, Ottawa, Canada.

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

Bulatov, A. A., & Dalmau, V. (2006). A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, *36*(1), 16–27.

Bulatov, A. A., Krokhin, A. A., & Jeavons, P. G. (2005). Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, *34*, 720–742.

Cohen, D., Jeavons, P., Jonsson, P., & Koubarakis, M. (2000). Building tractable disjunctive constraints. *Journal of the ACM*, *47*(5), 826–853.

Dowling, W. F., & Gallier, J. H. (1984). Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *The Journal of Logic Programming*, *1*(3), 267–284.

Drakengren, T., & Jonsson, P. (1998). Reasoning about set constraints applied to tractable inference in intuitionistic logic. *Journal of Logic and Computation*, *8*(6), 855–875.

Garey, M., & Johnson, D. (1978). *A guide to NP-completeness*. CSLI Press, Stanford.

Hodges, W. (1993). *Model theory*. Cambridge University Press.

Idziak, P. M., Markovic, P., McKenzie, R., Valeriote, M., & Willard, R. (2010). Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, *39*(7), 3023–3037.

Jonsson, P., & Bäckström, C. (1998). A unifying approach to temporal constraint reasoning. *Artificial Intelligence*, *102*(1), 143–155.

Jonsson, P., & Drakengren, T. (1997). A complete classification of tractability in RCC-5. *Journal of Artificial Intelligence Research*, *6*, 211–221.

Koppelberg, S. (1989). Projective boolean algebras. In *Handbook of Boolean Algebras*, Vol. 3, pp. 741–773. North Holland, Amsterdam-New York-Oxford- Tokyo.

Koubarakis, M. (2001). Tractable disjunctions of linear constraints: Basic results and applications to temporal reasoning. *Theoretical Computer Science*, *266*, 311–339.

Krötzsch, M., Rudolph, S., & Hitzler, P. (2006). On the complexity of Horn description logics. In *OWL: Experiences and Directions Workshop*.

Kuncak, V., Nguyen, H. H., & Rinard, M. C. (2006). Deciding boolean algebra with presburger arithmetic. *Journal of Automatic Reasoning*, *36*(3), 213–239.

Kuncak, V., & Rinard, M. C. (2007). Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In *Proceedings of the International Conference on automated deduction (CADE)*, pp. 215–230.

Küsters, R., & Molitor, R. (2002). Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, *15*(1), 47–59.

Lassez, J.-L., & McAloon, K. (1989). Independence of negative constraints. In *International Joint Conference on Theory and Practice of Software Development (TAPSOFT), Volume 1*, pp. 19–27.

Marriott, K., & Odersky, M. (1996). Negative Boolean constraints. *Theoretical Computer Science*, *160*(1&2), 365–380.

Schaefer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pp. 216–226.

Szendrei, A. (1986). *Clones in universal algebra*. Séminaire de Mathématiques Supérieures. Les Presses de l'Université de Montréal.