Tractable Set Constraints

Manuel Bodirsky
École Polytechnique
LIX (CNRS UMR 7161)
Palaiseau, France

Martin Hils
Équipe de Logique Mathématique
IMJ (CNRS UMR 7586)
Université Paris 7, France

Alex Krimkevich
Stanford University
Department of Computer Science
Stanford, CA, USA

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, 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 $\mathrm{CSP}(\Gamma)$ where the set of allowed constraints is formed from a fixed set Γ of relations $R \subseteq D^k$ over a common domain D. This way of parametrizing the constraint satisfaction problem by a *constraint language* Γ has lead to many strong algorithmic results [Bulatov and Dalmau, 2006; Idziak *et al.*, 2007; Barto and Kozik, 2009; Bodirsky and Kutz, 2007], and to many powerful hardness conditions for large classes of constraint satisfaction problems [Schaefer, 1978; Bulatov *et al.*, 2005; Bulatov, 2003; 2006; Bodirsky and 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 $\sqcap$, $\sqcup$, and c, which are function symbols for intersection, union, and complementation, respectively. Details of the formal definition and many examples of set constraint languages can be found in Section 3. The choice of $\mathbb{N}$ is just for notational convenience; as we will see, we could have selected any infinite set for our purposes. In the following, a *set constraint satisfaction problem (set CSP)* is a problem of the form $\mathrm{CSP}(\Gamma)$ for a set constraint language Γ. It has been shown by Marriott and Odersky [Marriott and Odersky, 1996] that all set CSPs are contained in NP; they also showed that the largest set constraint language, which consists of *all* relations that can be defined as described above, has an NP-hard set CSP.

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

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

They also showed that $\mathrm{CSP}(\Gamma)$ 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

$$x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee x_0 \,||\, y_0$$

where $x_0, \ldots, x_k, y_0, \ldots, y_k$ are not necessarily distinct variables. 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, Drakengren and Jonsson [Drakengren and Jonsson, 1998] show that if Γ contains the relations defined by formulas of the form

$$x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \vee u_1 \,||\, v_1 \vee \cdots \vee u_k \,||\, v_k$$

the problem $\mathrm{CSP}(\Gamma)$ is NP-hard.

Contributions and Outline. We present a significant extension of Drakengren and Jonsson's set constraint language (Section 3) whose CSP can still be solved in quadratic time in the input size (Section 6); 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 7), 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 [Koubarakis, 2001; Jonsson and Bäckström, 1998; Marriott and Odersky, 1996] – also see [Broxvall *et al.*, 2002; Cohen *et al.*, 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. Due to space limitations, all proofs have been omitted; they will appear in the long version of this paper.

Application Areas and Related Literature

Set Constraints for Programming Languages. Set constraints find applications in program analysis; here, a set constraint is of the form $X \subset Y$, 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; see [Aiken, 1994] for a survey.

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 and Rinard, 2007]. This logic (called QFBAPA) is interesting for program verification [Kuncak *et al.*, 2006]. The question whether the approach presented in this paper can also be applied to search for polynomial-time tractable fragments of QFBAPA is left as an open problem.

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 and Molitor, 2002; Baader *et al.*, 2005; Krötzsch *et al.*, 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.

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, 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 in [Drakengren and Jonsson, 1998] (which are contained in the class of set constraint languages considered here), and RCC-5 and its fragments.

2 Constraint Satisfaction Problems

To use existing terminology in logic and model theory, it will be convenient to formalize constraint languages as (relational) *structures* (see e.g. [Hodges, 1993]). A *structure* Γ is a tuple $(D; f_1^\Gamma, f_2^\Gamma, \dots, R_1^\Gamma, R_2^\Gamma, \dots)$ 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 $\mathrm{CSP}(\Gamma)$: given a finite set of variables V and a conjunction Φ of atomic formulas of the form $R(x_1, \dots, x_k)$, where $x_1, \dots, x_k \in V$ and $R \in \tau$, does there exists an assignment $s : V \to D$ such that for every constraint $R(x_1, \dots, x_k)$ in the input we have that $(s(x_1), \dots, s(x_k)) \in R^\Gamma$?

The mapping s is also called a *solution* to the *instance* Φ of $\mathrm{CSP}(\Gamma)$, and the conjuncts of Φ are called *constraints*.

Note that we only introduce constraint satisfaction problems $\mathrm{CSP}(\Gamma)$ for *finite constraint languages*, i.e., relational structures Γ with a *finite* relational signature.

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 $\mathbb{N} \setminus S$;
- $\mathbf{0}$ and $\mathbf{1}$ are constants (treated as 0-ary function symbols) denoting the empty set $\emptyset$ and the full set $\mathbb{N}$, respectively.

Sometimes, we simply write $\sqcap$ for the function $\sqcap^{\mathfrak{S}}$ and $\sqcup$ for the relation $\sqcup^{\mathfrak{S}}$, i.e., we do not distinguish between function symbol and respective relation. 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 with a set of relations with a quantifier-free first-order definition in $\mathfrak{S}$; we always allow equality in first-order formulas. For example, the relation $\{(x, y, z) \in \mathcal{P}(\mathbb{N})^3 \mid x \cap y \subseteq z\}$ has the quantifier-free first-order definition $z \sqcap (x \sqcap y) = x \sqcap y$ over $\mathfrak{S}$.

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

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.

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 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 (every term over $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$ can be re-written 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 $c_i := \{l_{ij} \mid 1 \leq j \leq n_i\}$ 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 $t = \mathbf{1}$ has a satisfying assignment).

We assume that all quantifier-free first-order formulas ϕ over the signature $\{\sqcap, \sqcup, c, \mathbf{0}, \mathbf{1}\}$ are written 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 an equivalent formula in this form. We refer to $C_i := \{L_{ij} \mid 1 \leq j \leq m_i\}$ as an *(outer) clause* of ϕ, and to L_{ij} as an *(outer) literal* of ϕ. Whenever convenient, we identify ϕ with its set of clauses. In the following, when we write *formula* we always mean a formula over the signature $\{\sqcup, \sqcap, c, \mathbf{0}, \mathbf{1}\}$.

4 $\mathcal{EI}$ Set Constraints

To define $\mathcal{EI}$ set constraints, we need to introduce a series of important functions defined on subsets of natural numbers.

Definition 2. Let

- $i : (\mathcal{P}(\mathbb{N}))^2 \to \mathcal{P}(\mathbb{N})$ be the function that maps (S_1, S_2) to the set $\{2x \mid x \in S_1\} \cup \{2x + 1 \mid x \in S_2\}$;
- F be the mapping that maps $S \subseteq \mathbb{N}$ to the set of finite non-empty subsets of S;
- $f : \mathbb{N} \to F(\mathbb{N})$ be a bijection between $\mathbb{N}$ and the set of finite non-empty subsets of $\mathbb{N}$ (since both sets are countable, such a bijection exists);
- $e : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ be $f^{-1} \circ F$ (first apply F, then apply the inverse of f);
- ei be the operation defined by $ei(x, y) \mapsto e(i(x, y))$.

Definition 3. Let $f : (\mathcal{P}(\mathbb{N}))^k \to \mathcal{P}(\mathbb{N})$ 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 $t^1, \ldots, t^k \in (\mathcal{P}(\mathbb{N}))^l$ we have that $(f(t^1_1, \ldots, t^k_1), \ldots, f(t^1_l, \ldots, t^k_l)) \in R$ if $t^i \in R$ for all $i \leq k$. We say that *f strongly preserves R* is f preserves both R and $(\mathcal{P}(\mathbb{N}))^l \setminus R$. 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*.

Note that if a relation R is (strongly) preserved by e and by i, then it is also (strongly) preserved by their composition ei.

Definition 4. The set of all relations with a quantifier-free first-order definition over $\mathfrak{S}$ that are preserved by the operation ei is denoted by $\mathcal{EI}$.

Proposition 9 shows that $\mathcal{EI}$ has a large subclass, called Horn-Horn, which has an intuitive syntactic description. In Section 5 we also present many examples of relations that are from $\mathcal{EI}$ and of relations that are not from $\mathcal{EI}$. Before, we have to verify some properties of the functions i and e.

Proposition 5. *The mapping i is an isomorphism between $\mathfrak{S}^2$ and $\mathfrak{S}$.*

We write $x \sqsubseteq y$ as a shortcut for $x \sqcap y = x$.

Proposition 6. *The function e has the following properties.*

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

Note that in particular e preserves $\sqsubseteq$, and that $e(c(x)) \sqsubseteq c(e(x))$.

5 Horn-Horn Set Constraints

A large and important subclass of $\mathcal{EI}$ set constraints is the class of Horn-Horn set constraints.

Definition 7. A quantifier-free first-order formula is called *Horn-Horn* if

- every outer clause is *Horn*, i.e., contains at most one positive outer literal;
- every inner clause is *Horn*, i.e., contains at most one positive inner literal.

Relations defined by a conjunction of outer Horn clauses are also called *Horn*. Relations defined by a Horn-Horn formula over $\mathfrak{S}$ are also called *Horn-Horn*.

Proposition 8. *Every Horn relation is preserved by i.*

Proposition 9. *Every Horn-Horn relation is preserved by e and i; in particular, it is from $\mathcal{EI}$.*

Proof. Suppose that R has a Horn-Horn definition ϕ over $\mathfrak{S}$ with variables V. Since R is in particular Horn, it is preserved by i by Proposition 8.

Now we verify that R is preserved by e. Recall that the operation e preserves $\sqcap$ and is injective (Proposition 6). Note that the formula $x \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$ can be equivalently written as $x \sqcap y_1 \sqcap \cdots \sqcap y_l = y_1 \sqcap \cdots \sqcap y_l$. By an easy induction, one shows that e also preserves this clause, and by injectivity it even strongly preserves this clause. It follows that e preserves formulas of the form $t = \mathbf{1}$ when all inner clauses of t are Horn, and similarly that e also preserves formulas of the form $t \neq \mathbf{1}$ when all inner clauses of t are Horn. Now, every tuple $t \in R$ must satisfy at least one literal in each outer clause of ϕ, and hence $e(t)$ satisfies the same outer clause, which is what we had to show. □

Examples.

1. The disjointness relation $||$ is Horn-Horn: it has the definition $\bar{x} \sqcup \bar{y} = \mathbf{1}$.
2. The relation $\{(x, y, u, v) \mid x \neq y \vee u = v\}$ can easily be seen to be Horn-Horn as well.
3. The ternary relation $\{(x, y, z) \mid x \cap y \subseteq z\}$, which we have encountered above, has the Horn-Horn definition $\bar{x} \sqcup \bar{y} \sqcup z = \mathbf{1}$.
4. Examples of relations that are clearly *not* Horn-Horn: $\{(x, y) \mid x \sqcup y = \mathbf{1}\}$ is violated by e, and $\{(x, y, z) \mid (x = y) \vee (y = z)\}$ is violated by i.
5. The formula

$$(x \sqcap y \neq x) \ \wedge \ (x \sqcap y \neq y)$$
$$\wedge \ (v = 1 \ \vee \ u = 1 \ \vee \ x \sqcup y \neq 1)$$

   is not Horn-Horn; more interestingly, it can be shown that it is from $\mathcal{EI}$, but there is no equivalent Horn-Horn formula, since the relation is not preserved by i.

Combining the last example with Proposition 9, we get the following result.

Corollary 10. *The set of relations admitting a Horn-Horn definition is strictly contained in $\mathcal{EI}$.*

We prepare now some results that can be viewed as a partial converse of Proposition 9.

Definition 11. A quantifier-free first-order formula ϕ is called *reduced* if it is in conjunctive normal form (CNF) and if every formula obtained from ϕ by removing an outer literal or an inner literal is not equivalent to ϕ over $\mathfrak{S}$.

Lemma 12. *Every quantifier-free formula is over $\mathfrak{S}$ equivalent to a reduced formula.*

We first prove the converse of Proposition 8.

Proposition 13. *Let ϕ be a reduced formula that is preserved by i. Then each outer clause of ϕ is Horn.*

Definition 14. Let ϕ be a formula with variable set V, and let $s : V \to \mathcal{P}(\mathbb{N})$ be a mapping. Then an assignment of the form $x \mapsto e(s(x))$ is called a *core assignment (for V)*. A reduced formula ϕ with variable set V is called *strongly reduced* if

- all negative outer literals are of the form $x \neq \mathbf{1}$ or $\overline{x} \neq \mathbf{1}$, where x is a variable,
- every outer clause is Horn, and
- whenever ϕ' is obtained from ϕ by removing an inner literal or an outer literal from ϕ, there is a core assignment that satisfies ϕ but not ϕ'.

Lemma 15. *For every quantifier-free formula ϕ there exists a strongly reduced formula ϕ' with the same set of variables V such that ϕ and ϕ' have the same core assignments.*

For every quantifier-free formula ϕ, we call a strongly reduced formula ϕ' as constructed in Lemma 15 from ϕ a *core formula for ϕ*.

Proposition 16. *Let ϕ be a formula preserved by ei, and let ϕ' be a core formula for ϕ. Then ϕ' is Horn-Horn.*

Proposition 17. *Let Γ be a finite set constraint language from $\mathcal{EI}$. Then $\mathrm{CSP}(\Gamma)$ can be reduced in linear time to the problem to find a satisfying assignment for a given set 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 a reduced formula that defines R over $\mathfrak{S}$ (which exists by Lemma 12), and let ψ_R be the core formula for ϕ_R (which exists by Lemma 15). By Proposition 16, the formula ψ_R is Horn-Horn.

We claim that Φ is a satisfiable instance of $\mathrm{CSP}(\Gamma)$ if and only if the set Ψ of all Horn-Horn clauses of formulas ψ_R obtained in the described manner is satisfiable.

If Φ has a solution $s : V \to \mathcal{P}(\mathbb{N})$, then the core assignment $x \mapsto ei(s(x), s(x))$ is also a solution because Γ is preserved by ei; but then for each R the core-formula ψ_R is satisfied by this assignment. We have thus found an assignment that satisfies Ψ. Conversely, if Ψ has an assignment, then it also has a core assignment, since Ψ is Horn-Horn and therefore preserved by e. Then the same assignment will also satisfy all the formulas ϕ_R, and therefore is a solution to Φ. □

6 Algorithm for Horn-Horn Constraints

This section concludes the proof that $\mathrm{CSP}(\Gamma)$ is tractable when all relations in Γ are from $\mathcal{EI}$. By Proposition 17, it suffices to give an efficient algorithm that takes as input a set Φ of Horn-Horn clauses and decides satisfiability of Φ over $\mathfrak{S} = (\mathcal{P}(\mathbb{N}); \sqcup, \sqcap, c, \mathbf{0}, \mathbf{1})$. We first discuss an important subroutine of our algorithm, which we call the *inner resolution algorithm*.

```
Inner-Res(Φ)
// Input: A finite set Φ of inner Horn clauses
// Accepts iff Φ is satisfiable
During the entire algorithm:
    if Φ contains an empty clause, then reject.
Repeat
    If Φ contains a positive unit clause {x} then
        Remove all clauses where the literal x occurs.
        Remove the literal x̄ from all clauses.
    End if
Until no clause changes.
Accept
```

Figure 1: Inner Resolution Algorithm.

Lemma 18. *Let Ψ be a set of inner Horn clauses. Then Inner-Res($\Psi \cup \{\overline{x_1} = \mathbf{1}, \ldots, \overline{x_k} = \mathbf{1}, y_0 = \mathbf{1}, \ldots, y_l = \mathbf{1}\}$) rejects if and only if Ψ implies that $x_1 \sqcup \cdots \sqcup x_k \sqcup \overline{y_1} \sqcup \cdots \sqcup \overline{y_l} = \mathbf{1}$.*

```
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 outer clause, then reject.
Repeat
    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
    End for
Until no outer literal is removed
Accept
```

Figure 2: Outer Resolution Algorithm.

Theorem 19. *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. The algorithm only rejects when it detects an empty clause. 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 $\Psi \cup \{\overline{x_1} = \mathbf{1}, \ldots, \overline{x_k} = \mathbf{1}, y_0 = \mathbf{1}, \ldots, y_l = \mathbf{1}\}$ for each inner clause $\{x_1, \ldots, x_k, \overline{y_1}, \ldots, \overline{y_l}\}$ of t. By Lemma 18, if Inner-Res rejects $\Psi \cup \{\overline{x_1} = \mathbf{1}, \ldots, \overline{x_k} = \mathbf{1}, y_0 = \mathbf{1}, \ldots, y_l = \mathbf{1}\}$ 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 affecting satisfiability.

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 $D = \{x_1, \ldots, x_k, \overline{y_1}, \ldots, \overline{y_l}\}$ 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 18, the inner resolution algorithm would have rejected $\Psi \cup \{\overline{x_1} = \mathbf{1}, \ldots, \overline{x_k} = \mathbf{1}, y_0 = \mathbf{1}, \ldots, y_l = \mathbf{1}\}$, and would have removed $\{t \neq \mathbf{1}\}$ from Φ. Let $D_1, \ldots, D_s$ be an enumeration of all remaining inner clauses D that appear in all remaining negative outer literals.

We claim that the core assignment $s : V \to \mathcal{P}(\mathbb{N})$ given by

$$x \mapsto e(i(\alpha_{D_1}(x), i(\alpha_{D_2}(x), \ldots \\ i(\alpha_{D_{s-1}}(x), \alpha_{D_s}(x)) \ldots)))$$

satisfies all clauses in Φ (where i is as in Proposition 5 and e is as in Proposition 6). 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 (Proposition 13). 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 must be a remaining inner clause $D = \{x_1, \ldots, x_k, \overline{y_1}, \ldots, \overline{y_l}\}$ of t. Observe that $s(x_0) \sqcup \cdots \sqcup s(x_k) \sqcup \overline{s(y_1)} \sqcup \cdots \sqcup \overline{s(y_l)} = \mathbf{1}$ if and only if $\alpha_{D_j}(x_0) \sqcup \cdots \sqcup \alpha_{D_j}(x_k) \sqcup \overline{\alpha_{D_j}(y_1)} \sqcup \cdots \sqcup \overline{\alpha_{D_j}(y_l)} = \mathbf{1}$ for all $1 \leq j \leq s$. Hence, and since $\alpha_D(x_0) \sqcup \cdots \sqcup \alpha_D(x_k) \sqcup \overline{\alpha_D(y_1)} \sqcup \cdots \sqcup \overline{\alpha_D(y_l)} \neq \mathbf{1}$, 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.

We conclude that s is a solution to Φ. □

Combining Proposition 17 with Theorem 19, we obtain:

Theorem 20. *Let Γ be a finite set constraint language from $\mathcal{EI}$. Then $\mathrm{CSP}(\Gamma)$ can be solved in quadratic time.*

7 Maximality

The tractability result Theorem 20 may be complemented by a hardness result. Indeed, the class $\mathcal{EI}$ is a *maximal tractable set constraint language* in the sense of the following theorem.

Theorem 21. *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 $\mathrm{CSP}(\Gamma')$ is NP-hard.

The proof of Theorem 21 uses the so-called *universal-algebraic approach* to the complexity of CSPs, as presented in [Bulatov *et al.*, 2005] for finite domain constraint satisfaction. This requires that we re-formulate set CSPs as constraint satisfaction problems for *ω-categorical* structures (see e.g. [Bodirsky and Kára, 2009]).

Because of the space limits, we cannot present the proof of Theorem 21 in this extended abstract. (We refer to the long version of this paper [Bodirsky *et al.*, 2011] for the proof.)

8 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.

Acknowledgement

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

References

[Aiken, 1994] Alexander Aiken. Set constraints: Results, applications, and future directions. In *PPCP*, pages 326–335, 1994.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI*, pages 364–369, 2005.

[Barto and Kozik, 2009] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of FOCS'09*, 2009.

[Bodirsky and Kára, 2009] Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2), 2009.

[Bodirsky and Kutz, 2007] Manuel Bodirsky and Martin Kutz. Determining the consistency of partial tree descriptions. *Artificial Intelligence*, 171:185–196, 2007.

[Bodirsky *et al.*, 2011] M. Bodirsky, M. Hils, and A. Krimkevich. Tractable set constraints. Preprint, arXiv:1104.1045v1 [cs.AI], 2011.

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

[Broxvall *et al.*, 2002] M. Broxvall, P. Jonsson, and J. Renz. Disjunctions, independence, refinements. *Artificial Intelligence*, 140(1/2):153–173, 2002.

[Bulatov and Dalmau, 2006] Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

[Bulatov *et al.*, 2005] Andrei Bulatov, Andrei Krokhin, and Peter G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.

[Bulatov, 2003] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of LICS'03*, pages 321–330, 2003.

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

[Cohen *et al.*, 2000] David Cohen, Peter Jeavons, Peter Jonsson, and Manolis Koubarakis. Building tractable disjunctive constraints. *Journal of the ACM*, 47(5):826–853, 2000.

[Drakengren and Jonsson, 1998] Thomas Drakengren and Peter Jonsson. Reasoning about set constraints applied to tractable inference in intuitionistic logic. *J. Log. Comput.*, 8(6):855–875, 1998.

[Hodges, 1993] Wilfrid Hodges. *Model theory*. Cambridge University Press, 1993.

[Idziak *et al.*, 2007] Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. In *Proceedings of LICS'07*, pages 213–224, 2007.

[Jonsson and Bäckström, 1998] Peter Jonsson and Christer Bäckström. A unifying approach to temporal constraint reasoning. *Artif. Intell.*, 102(1):143–155, 1998.

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

[Krötzsch *et al.*, 2006] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. On the complexity of Horn description logics. In *OWLED*, 2006.

[Kuncak and Rinard, 2007] Viktor Kuncak and Martin C. Rinard. Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In *CADE*, pages 215–230, 2007.

[Kuncak *et al.*, 2006] Viktor Kuncak, Huu Hai Nguyen, and Martin C. Rinard. Deciding boolean algebra with presburger arithmetic. *J. Autom. Reasoning*, 36(3):213–239, 2006.

[Küsters and Molitor, 2002] Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Commun.*, 15(1):47–59, 2002.

[Marriott and Odersky, 1996] Kim Marriott and Martin Odersky. Negative boolean constraints. *Theor. Comput. Sci.*, 160(1&2):365–380, 1996.

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