Chain Length and CSPs Learnable with Few Queries

Christian Bessiere, Clement Carbonnel, George Katsirelos

Chain Length and CSPs Learnable with Few Queries

Christian Bessiere,[1] **Clément Carbonnel,**[1] **George Katsirelos**[2]
[1]LIRMM, CNRS, University of Montpellier, France
{christian.bessiere, clement.carbonnel}@lirmm.fr
[2]UMR MIA-Paris, INRA, AgroParisTech, Université Paris-Saclay, Paris, France
gkatsi@gmail.com

Abstract

The goal of constraint acquisition is to learn exactly a constraint network given access to an oracle that answers truthfully certain types of queries. In this paper we focus on partial membership queries and initiate a systematic investigation of the learning complexity of constraint languages. First, we use the notion of chain length to show that a wide class of languages can be learned with as few as $O(n \log(n))$ queries. Then, we combine this result with generic lower bounds to derive a dichotomy in the learning complexity of binary languages. Finally, we identify a class of ternary languages that eludes our framework and hints at new research directions.

1 Introduction

Constraint programming is a declarative programming paradigm for solving combinatorial problems, with a wide range of applications in artificial intelligence (van Beek and Chen 1999; Négrevergne and Guns 2015) and operational research (Shaw 1998; Hooker and van Hoeve 2018). The user declares a list of variables, which range over a domain of his choice, and then *constraints*, which specify forbidden assignments to certain subsets of variables. A solver will then either find an assignment to the variables that satisfies every constraint or prove that no such assignment exists.

From the user's perspective, the difficulty lies in the modelling phase. Variables and domain values are often easily derived from the unkowns in the original problem, but specifying rigorously what a solution is using only relations imposed upon subsets of variables may require considerable expertise, especially if the solver only offers a limited catalog of possible relations. Constraint acquisition systems are learning agents designed to assist the user in modelling his problem. Such systems provide the user with simple queries — for instance *membership queries*, in which a given assignment must be labelled as either "solution" or "non-solution" — and use the collected data to construct a constraint network that hopefully corresponds to user's problem. There exist a number of constraint acquisition systems (Bessiere et al. 2005; 2013; Beldiceanu and Simonis 2012), with various learning strategies and theoretical guarantees.

For constraint acquisition systems, there is a tradeoff between the computational difficulty of answering the queries and the number of queries needed to identify the target network. Some queries that could guarantee quick convergence towards a constraint network are too difficult for a human to answer with certainty. For instance, equivalence queries provide the user with a candidate constraint network N and ask if it captures his problem; if it does not, the user must reply with an assignment misclassified by N. Other queries (such as membership queries) are easy to answer but convergence may require exponentially many queries, even on very simple constraint networks (Bessiere et al. 2017).

Partial membership queries are generalisations of membership queries in which the user is required to classify *partial* assignments. In 2013, Bessiere et al. presented a constraint acquisition algorithm `Quacq` based on partial membership queries, and they observed that on certain elementary constraint languages ($\{=, \neq\}$ and $\{>\}$ on the Boolean domain) their algorithm is guaranteed to converge after $O(n \log(n))$ queries (Bessiere et al. 2013). These languages are learnable with few, simple queries, but they are not very useful for modelling. Are there more expressive languages with similar learning properties?

Our first contribution is a proof that `Quacq`-like algorithms can learn a considerably larger class of languages with a quasi-linear number of partial membership queries. Our argument revolves around the notion of *chain length*. When one constraint is removed from a constraint network, the solution set may either remain unchanged (if the constraint is redundant) or increase in size. The chain length of a constraint language corresponds roughly to the maximum number of times the size of the solution set may increase when the constraints of an n-variables network are removed one by one. We find that `Quacq`-like algorithms always learn constraint networks with $O(\mathrm{CL}(n) \cdot \log(n))$ queries, where $\mathrm{CL}(n)$ is the chain length of the language.

As an example, consider a system of homogeneous linear equations over a finite field. The solution set is a vector space. When an equation (a constraint) is removed, either the solution set remains the same, or new linearly independent solutions appear and the dimension increases. Since the dimension of the solution set is bounded by the num-

ber n of variables, the latter case may only occur at most n times if all the constraints are removed one by one: sets of homogeneous linear equations have linear chain length. As a consequence, they can be learned with a quasi-linear number of partial membership queries. The most general property known to imply linear chain length is the presence of a (finite-domain) *Mal'tsev embedding*, which was introduced recently in the context of parameterised complexity (Lagerkvist and Wahlström 2017). These languages can be fairly expressive; in fact, some NP-hard languages have a Mal'tsev embedding (Lagerkvist and Wahlström 2017).

As our second contribution, we show that every binary constraint language that does not have linear chain length (via a notion closely related to Mal'tsev embeddings, which we call *Mal'tsev derivations*) has VC-dimension $\Omega(n^2)$. This implies that a binary language is either learnable with $O(n \log(n))$ partial membership queries, or it is not learnable with $o(n^2)$ yes/no queries of any kind.

Third and last, we show that there exist *ternary* constraint languages with VC-dimension $o(n^2)$ but no Mal'tsev embedding nor Mal'tsev derivation. This suggests that classifying the learning complexity of ternary languages requires tools beyond VC-dimension and Mal'tsev derivations.

2 Preliminaries

CSP. A (constraint) language is a finite set of relations over a finite domain D. Given a language Γ, a *CSP instance over* Γ is a pair (X, C) where X is a set of n variables and C is a set of *constraints*, that is, pairs $c = (R_c, S_c)$ where $R_c \in \Gamma$ and S_c is a tuple of variables whose length is the arity of R_c. An assignment $\phi : X \to D$ *satisfies* a constraint $c = (R_c, S_c)$ if $\phi(S_c) \in R_c$; otherwise it *violates* c. A *solution* to a CSP instance is an assignment $\phi : X \to D$ that satisfies every constraint. We will assume that the variable set X has a total order $x_1 < x_2 < \ldots < x_n$ and define $\mathrm{sol}(I)$ as the n-ary relation $\{(\phi(x_1), \ldots, \phi(x_n)) \mid \phi \text{ is a solution to } I\}$ over D. The *restriction* of a CSP instance $I = (X, C)$ to a subset X' of X, denoted by $I[X']$, is obtained by dropping all variables in $X \backslash X'$ and all constraints c such that S_c contains a variable in $X \backslash X'$.

Constraint acquisition. We use Angluin's *exact learning* model (Angluin 1988). Let Γ be a language over a domain D. We associate to each CSP instance $I = (X, C)$ over Γ an oracle which answers truthfully a certain type of queries. A *membership query* asks whether an assignment $\phi : X \to D$ is a solution to I, a *partial membership query* asks whether a partial assignment $\phi : X' \to D$ is a solution to $I[X']$, and an *equivalence query* asks whether a CSP instance I' satisfies $\mathrm{sol}(I) \neq \mathrm{sol}(I')$, in which case the oracle must return a tuple in the symmetric difference of $\mathrm{sol}(I)$ and $\mathrm{sol}(I')$. A query is *Boolean* if the oracle answer can only be `yes` or `no`. Membership queries and partial membership queries are Boolean, but not equivalence queries.

A *constraint acquisition algorithm* for Γ is an algorithm that, given a variable set X and access to an oracle for a target CSP instance $I_T = (X, C_T)$ over Γ, outputs a CSP instance I_L over Γ such that $\mathrm{sol}(I_T) = \mathrm{sol}(I_L)$. Given a type Q of queries, we say that a finite constraint language Γ is learnable with $f(n)$ queries of type Q if there exists a constraint acquisition algorithm that learns any CSP instance over Γ of n variables with at most $f(n)$ queries of type Q. Note that the time complexity of the acquisition algorithm needs not be bounded by a polynomial.

c-definitions. Given a language Γ over a domain D and a relation $R \subseteq D^r$, we say that R is *c-definable* over Γ if there exists a CSP instance I over Γ such that $\mathrm{sol}(I) = R$. (c-definitions are particular cases of the more widely used *qfpp-definitions*, which allow equality relations in addition to the relations of Γ; see e.g. (Jonsson, Lagerkvist, and Roy 2017).) By extension, a language Γ' is c-definable in Γ if every $R \in \Gamma'$ is c-definable in Γ. We define the *chain length* of Γ as the function $\mathrm{CL}_\Gamma : \mathbb{N}^* \to \mathbb{N}^*$ such that $\mathrm{CL}_\Gamma(n)$ is the maximum length of a sequence $R_1, R_2, \ldots, R_q$ of relations of the same arity $k \leq n$ such that

$$R_1 \subsetneq R_2 \subsetneq \ldots \subsetneq R_q$$

and each R_i, $i \in \{1, \ldots, q\}$, is c-definable over Γ.

We say that Γ *shatters* a relation R if for every subrelation $R' \subseteq R$ there exists a relation R^* such that $R^* \cap R = R'$ and R^* is c-definable over Γ. (Note that R itself needs not be c-definable.) The *VC-dimension* of Γ is the function $\mathrm{VC}_\Gamma : \mathbb{N}^* \to \mathbb{N}^*$ such that $\mathrm{VC}_\Gamma(n)$ is the maximum size (i.e. number of tuples) of a relation of arity at most n that is shattered by Γ. We will also discuss a measure of redundancy, which is tightly connected to the VC-dimension of a language. Given a CSP instance (X, C), we say that a constraint $c \in C$ is *non-redundant* if there exists an assignment $\phi : X \to D$ such that ϕ is a solution to $(X, C \backslash \{c\})$ but ϕ violates c. The *non-redundancy* of a language Γ, which we denote by $\mathrm{NRD}_\Gamma(n)$, is the maximum number of non-redundant constraints in a CSP instance over Γ with n variables. Throughout the paper we will use the following observations about $\mathrm{CL}_\Gamma(n)$, $\mathrm{VC}_\Gamma(n)$ and $\mathrm{NRD}_\Gamma(n)$.

Observation 1. *If* $\Gamma = \Gamma_1 \cup \Gamma_2$, *then* $CL_\Gamma(n) \leq CL_{\Gamma_1}(n) + CL_{\Gamma_2}(n)$.

Proof. Let $n > 0$, $q = \mathrm{CL}_\Gamma(n)$ and $R_1, \ldots, R_q$ be a sequence of relations of arity at most n, c-definable over Γ, and such that $R_1 \subsetneq R_2 \subsetneq \ldots \subsetneq R_q$. Let $I_1, \ldots, I_q$ be CSP instances over Γ such that $I_j = (X, C_j)$ and $\mathrm{sol}(I_j) = R_j$ for all $j \leq q$. We assume without loss of generality that $C_{j+1} \subsetneq C_j$ for all $j < q$. (If it is not the case, we add to C_j all constraints in $\cup_{k>j} C_k$; this operation leaves $\mathrm{sol}(I_j)$ unchanged.) We partition each constraint set C_j into $C_j^1 = \{(R, S) \in C_j \mid R \in \Gamma_1\}$, $C_j^2 = \{(R, S) \in C_j \mid R \in \Gamma_2\}$ and define $R_j^1 = \mathrm{sol}(X, C_j^1)$, $R_j^2 = \mathrm{sol}(X, C_j^2)$. Then, we have $R_j = R_j^1 \cap R_j^2$ for all $j \leq q$. Furthermore, since $R_1 \subsetneq \ldots \subsetneq R_q$, $R_1^1 \subseteq \ldots \subseteq R_q^1$ and $R_1^2 \subseteq \ldots \subseteq R_q^2$, it follows that for all $j < q$ we have either $R_j^1 \subsetneq R_{j+1}^1$ or $R_j^2 \subsetneq R_{j+1}^2$. These events may happen at most $\mathrm{CL}_{\Gamma_1}(n)$ times and $\mathrm{CL}_{\Gamma_2}(n)$ times, respectively. This implies $q = \mathrm{CL}_\Gamma(n) \leq \mathrm{CL}_{\Gamma_1}(n) + \mathrm{CL}_{\Gamma_2}(n)$. □

Observation 2. *If Γ is c-definable over Γ', then $CL_\Gamma(n) \leq CL_{\Gamma'}(n)$.*

Proof. The claim follows directly from the fact that every relation c-definable over Γ is also c-definable over Γ'. □

Observation 3. *If Γ has VC-dimension $\Omega(f(n))$, then Γ is not learnable with $o(f(n))$ Boolean queries.*

Proof. Suppose that A is a constraint acquisition algorithm that learns any CSP over Γ with $o(f(n))$ queries. The output of A is completely determined by the $o(f(n))$ bits that it receives as answers to its Boolean queries. There are $2^{o(f(n))}$ possible combinations of inputs, so the algorithm can only output $2^{o(f(n))}$ different CSP instances over n variables. However, since Γ has VC-dimension $\Omega(f(n))$, the number of CSP instances over Γ with distinct solution sets and n variables is $2^{\Omega(f(n)}$. It follows that for sufficiently large n the algorithm has fewer outputs than there are distinct n-variables constraint networks over Γ, a contradiction. □

Observation 4. *For every language Γ, it holds that $NRD_\Gamma(n) = VC_\Gamma(n)$.*

Proof. We first prove $\text{NRD}_\Gamma(n) \leq \text{VC}_\Gamma(n)$. Let $n > 0$ and $I = (X, C)$ be a CSP instance over Γ with n variables and such that C contains precisely $\text{NRD}_\Gamma(n)$ constraints, all of which are non-redundant. Then, for each $c \in C$ there exists an assignment ϕ_c such that ϕ_c violates c but satisfies every other constraint. For each $c \in C$, let t_c be a tuple such that $t_c[i] = \phi_c(x_i)$ for all $i \in \{1, \ldots, n\}$. We claim that Γ shatters the relation $R_I = \{t_c \mid c \in C\}$. For each $R_I^* \subseteq R_I$, we define the instance $I^* = (X, C^*)$ with $\{c \in C \mid t_c \notin R_I^*\}$). Then, ϕ_c is a solution to I^* if and only if $c \notin C^*$, and hence $R_I \cap \text{sol}(I^*) = R_I^*$. Therefore, Γ shatters R_I, which contains $\text{NRD}_\Gamma(n)$ tuples, and hence $\text{NRD}_\Gamma(n) \leq \text{VC}_\Gamma(n)$. Now we prove $\text{NRD}_\Gamma(n) \geq \text{VC}_\Gamma(n)$. Let R be a relation of arity $r \leq n$ such that Γ shatters R and $q = |R| = \text{VC}_\Gamma(n)$. Let $t_1, \ldots, t_q$ be an arbitrary ordering of the tuples of R and $I_1, \ldots, I_q$ be a set of CSP instances over Γ with variable set X and constraint sets C_i, $1 \leq i \leq q$, such that $\text{sol}(I_i) \cap R = R \backslash \{t_i\}$. Note that each C_i can be chosen so that it contains a single constraint c_i, since $t_i \notin \text{sol}(I_i)$ if and only if there exists $c_i \in C_i$ such that $t_i \notin \text{sol}(X, \{c_i\})$. This implies that no constraint c may belong to C_i and C_j for any $j \neq i$, since otherwise we would have $\text{sol}(I_i) = \text{sol}(I_j)$. If we define $I^* = (X, C_1 \cup \ldots \cup C_q)$ then this instance has exactly $q = \text{VC}_\Gamma(n)$ constraints $c_1, \ldots, c_q$. Furthermore, removing any constraint c_i adds at least the tuple t_i to $\text{sol}(I^*)$, so no constraint is redundant and $\text{NRD}_\Gamma(n) \geq \text{VC}_\Gamma(n)$. □

Observation 5. *For every language Γ, it holds that $VC_\Gamma(n) \leq \sum_{R \in \Gamma} VC_{\{R\}}(n)$.*

Proof. By Observation 4 it suffices to show that every CSP instance $I = (X, C)$ over Γ with n variables has at most $\sum_{R \in \Gamma} \text{NRD}_{\{R\}}(n)$ non-redundant constraints. For each $R \in \Gamma$ we define $I_R = (X, \{c = (R, S_c) \in C\})$. Note that if $c = (R, S)$ is non-redundant in I, then it is non-redundant in I_R. Each instance I_R has at most $\text{NRD}_{\{R\}}(n)$ non-redundant constraints, so the claim follows. □

Partial polymorphisms. A *partial operation* of arity r on a set D is a mapping $f : Q \to D$ where $Q \subseteq D^r$. The set Q is called the *domain* of f and we write $Q = \text{domain}(f)$. Given a relation R of arity k over a set D and a partial operation f on D of arity r, we say that f is a *partial polymorphism* of R if for every $t_1, \ldots, t_r \in R$, either $f(t_1, \ldots, t_r) \in R$ or there exists $i \in \{1, \ldots, k\}$ such that $(t_1[i], \ldots, t_r[i]) \notin \text{domain}(f)$. If in addition we have $\text{domain}(f) = D^r$, we say that f is a *polymorphism* of R. A partial operation is a (partial) polymorphism of a language Γ if it is a (partial) polymorphism of each of its relations. Polymorphisms and partial polymorphisms are fundamental technical tools in the study of the complexity and expressivity of constraint languages. For a comprehensive exposition of the subject we refer the reader to (Barto, Krokhin, and Willard 2017; Couceiro, Haddad, and Lagerkvist 2019).

3 Constraint acquisition and chain length

In this section we describe a simple constraint acquisition algorithm that learns any CSP instance with n variables over a language Γ using $O(\text{CL}_\Gamma(n) \cdot \log(n))$ partial membership queries. Our contribution does not lie in the algorithm itself, which is essentially a stripped-down version of the constraint acquisition algorithm `Quacq` found in (Bessiere et al. 2013), but rather in its analysis. We borrow the following proposition from their work.

Proposition 6 ((Bessiere et al. 2013), Proposition 2)**.** *There exists an algorithm `FindScope` which given in input*

- *a set X of n variables,*
- *a partial membership oracle for a target CSP instance (X, C_T), and*
- *a negatively answered membership query ϕ*

outputs the set X_c of variables that appear in the scope S_c of a constraint $c \in C_T$ that ϕ violates, using $O(|S_c| \cdot \log(n))$ partial membership queries.

Theorem 7. *Let Γ be a fixed language. There exists a constraint acquisition algorithm that learns any CSP instance over Γ on n variables using $O(CL_\Gamma(n) \cdot \log(n))$ partial membership queries.*

Proof. Let O be an oracle associated with a CSP instance $I_T = (X, C_T)$ over Γ. Consider the following constraint acquisition algorithm. Let L and P be two sets of constraints from the language Γ on the variable set X. These sets can be understood as the sets of *learned* and *possible* constraints, respectively. When the algorithm is invoked, L is initialised to the empty set and P is initialised to the set of all possible constraints from Γ on X. The algorithm then repeats the following steps until $\text{sol}(X, L) \subseteq \text{sol}(X, P)$:

- Find $\phi \in \text{sol}(X, L) \backslash \text{sol}(X, P)$.
- Query ϕ to the oracle.
  - If the answer is `yes`, set $P = \{c \in P : \phi \text{ satisfies } c\}$.
  - If the answer is `no`, invoke `FindScope` on input (X, O, ϕ), and then query all possible partial assignments to the returned subset X' to determine $\text{sol}(I_T[X'])$. Add to L a set C' of constraints such that $\text{sol}(X', C') = \text{sol}(I_T[X'])$.

Once the condition $\text{sol}(X, L) \subseteq \text{sol}(X, P)$ holds, the algorithm returns (X, L) and halts.

We now turn to the analysis. First, observe that at each loop either L grows or P shrinks, so the algorithm will eventually halt. For correctness, it suffices to show that we have the two invariants (i) $\text{sol}(X, C_T) \subseteq \text{sol}(X, L)$ and (ii) $C_T \subseteq P$. (If both (i) and (ii) hold, then $\text{sol}(X, P) \subseteq \text{sol}(X, C_T) \subseteq \text{sol}(X, L)$ is also an invariant, and upon termination we have $\text{sol}(X, C_T) = \text{sol}(X, L)$.) Both (i) and (ii) hold at initialisation. Notice that whenever a constraint c is removed from P, there exists a solution ϕ to (X, C_T) that violates c, so $c \notin C_T$ and (ii) is indeed an invariant of the algorithm. Similarly, whenever a constraint c is added to L we have $\text{sol}(X, C_T \cup \{c\}) = \text{sol}(X, C_T)$; it follows that if $\text{sol}(X, C_T) \subseteq \text{sol}(X, L)$ then $\text{sol}(X, C_T) = \text{sol}(X, C_T \cup \{c\}) \subseteq \text{sol}(X, L \cup \{c\})$, so (i) is also an invariant and the algorithm is correct.

Now, we bound the number of oracle queries performed by the algorithm before it halts. We will use L_i, $0 \leq i \leq l$, and P_j, $0 \leq j \leq p$ to denote the sequences of distinct states of L and P, respectively. Using Proposition 6 to bound the number of oracle queries within each call to `FindScope`, and observing that any subset X' of variables returned by `FindScope` has bounded size since Γ (and hence the maximum constraint arity) is fixed, we find that the algorithm makes a total of $O(p + l \cdot \log(n))$ oracle queries. Thus, it suffices to bound l and p by a linear function of $\text{CL}_\Gamma(n)$.

First, observe that for every $i \in \{0, \ldots, l-1\}$, it holds that $\text{sol}(X, L_{i+1}) \subsetneq \text{sol}(X, L_i)$ since $L_i \subsetneq L_{i+1}$ and $\text{sol}(X, L_{i+1})[X'] = \text{sol}(X, C_T)[X'] \neq \text{sol}(X, L_i)[X']$, where X' is the set of variables returned by `FindScope`. Each relation $\text{sol}(X, L_i)$, $0 \leq i \leq l$, is c-definable over Γ so we deduce $l \leq \text{CL}_\Gamma(n)$.

In order to derive a similar bound on p, let ϕ_j, $0 \leq j \leq p-1$, be the positively answered query that triggered the update of P from state P_j to state P_{j+1}, and let c_j be an arbitrary constraint in $P_j \backslash P_{j+1}$. In particular, observe that the assignment ϕ_j violates c_j, but ϕ_j does not violate c_k for any $k > j$. Thus, we have

$$\text{sol}(X, \{c_p\}) \subsetneq \text{sol}(X, \{c_{p-1}, c_p\}) \subsetneq \ldots \subsetneq \text{sol}(X, \{c_0, \ldots, c_p\})$$

since for each $0 \leq j \leq p-1$ the assignment ϕ_j is a solution to $(X, \{c_{j+1}, \ldots, c_p\})$ but not to $(X, \{c_j, \ldots, c_p\})$. Again, each of these relations is c-definable over Γ so $p \leq \text{CL}_\Gamma(n)$. The total query complexity of the algorithm is $O(\text{CL}_\Gamma(n) + \text{CL}_\Gamma(n) \cdot \log(n)) = O(\text{CL}_\Gamma(n) \cdot \log(n))$, as claimed. □

4 Languages of linear chain length

As a corollary to Theorem 7, every language Γ with linear chain length is learnable with a quasi-linear number of partial membership queries. As noted in the introduction, linear equations over finite fields have linear chain length. They correspond to a special case of a larger class of languages; to describe this class we will need additional terminology.

A *Mal'tsev* operation on a domain D is a ternary operation $f : D^3 \to D$ such that $f(x, x, y) = f(y, x, x) = y$ for all $x, y \in D$. A language Γ is Mal'tsev if it has a Mal'tsev polymorphism. Mal'tsev languages have played an important role in the study of the complexity of constraint languages (Bulatov and Dalmau 2006; Idziak et al. 2010). Their most salient property is that any c-definable relation has a linear-size *generating set* which encodes them efficiently, in the same way that bases describe vector spaces. The fact that Mal'tsev languages have linear chain length is folklore; we include a proof for completeness.

The *signature* of a relation R over D of arity r, denoted by $\text{Sig}(R)$, is the set of all triples $(k, d_1, d_2) \in \{1, \ldots, r\} \times D^2$ such that there exist $t_1, t_2 \in R$ satisfying $t_1[k] = d_1, t_2[k] = d_2$ and $t_1[i] = t_2[i]$ for all $i < k$.

Lemma 8 ((Bulatov and Dalmau 2006), Lemma 3.1)**.** *Let ϕ be a Mal'tsev operation over a domain D and R_1, R_2 be two relations over D such that $R_1 \subseteq R_2$ and ϕ is a polymorphism of both R_1 and R_2. Then, $\text{Sig}(R_1) = \text{Sig}(R_2)$ if and only if $R_1 = R_2$.*

Proposition 9. *If Γ is Mal'tsev, then $\text{CL}_\Gamma(n) = O(n)$.*

Proof. Let ϕ be a Mal'tsev polymorphism of Γ. Let $R_1, \ldots, R_q$ be a sequence of n-ary relations, c-definable over Γ, such that $R_1 \subsetneq \ldots \subsetneq R_q$. Polymorphisms are invariant under c-definitions (Barto, Krokhin, and Willard 2017), so ϕ is a polymorphism of $R_1, \ldots, R_q$. By Lemma 8 we have $\text{Sig}(R_1) \subsetneq \ldots \subsetneq \text{Sig}(R_q)$. Since $|\text{Sig}(R_q)| \leq n|D|^2$, we obtain $q \leq n|D|^2 + 1 = O(n)$ and $\text{CL}_\Gamma(n) = O(n)$. □

We can extend this result using Observation 1 and Observation 2, which provide ways to combine languages and relations while preserving chain length. We say that a constraint language Γ has a *Mal'tsev derivation* if there exists a language $\hat{\Gamma}$ such that (i) every relation in $\hat{\Gamma}$ has a Mal'tsev polymorphism and (ii) every relation in Γ is c-definable in $\hat{\Gamma}$. (Intuitively, languages with Mal'tsev derivations are those that can be obtained from a finite number of Mal'tsev relations through a finite sequence of language unions and c-definitions.) The following proposition follows from Proposition 9, Observation 1 and Observation 2.

Proposition 10. *Let Γ be a language. If Γ has a Mal'tsev derivation, then $\text{CL}_\Gamma(n) = O(n)$.*

Combining this result with Theorem 7, we obtain a general sufficient condition for learnability with a quasi-linear number of partial membership queries.

Theorem 11. *If Γ has a Mal'tsev derivation, then Γ is learnable with $O(n \log(n))$ partial membership queries.*

Example 1. A classical example of Mal'tsev relations are those corresponding to the solution space of an equation over a finite field (see e.g. Example 5 in (Bulatov and Dalmau 2006)). Theorem 11 implies that any language in which each individual relation is the solution space of an equation over a finite field is learnable with a quasi-linear number of partial membership queries, even if the field in question differs from one relation to another.

Example 2. Let $k \geq 3$ be a fixed integer and consider the relation R_k that contains all Boolean k-tuples of Hamming weight 1. This relation is not Mal'tsev as mappings satisfying the Mal'tsev identity must map any three tuples of R_k to one with Hamming weight at least 2. However, R_k is precisely the intersection of $\{0,1\}^k$ with the solution set R^* of the equation $(x_1 + \ldots + x_k) \mod k = 1$. Both R^* and $\{0,1\}^k$ are Mal'tsev (we invite the reader to verify that the operation $f(x,y,z) = x - y + z \mod k$ is a polymorphism of R^*), so $\{R_k\}$ has a Mal'tsev derivation.

Mal'tsev derivations are very similar to a notion introduced in (Lagerkvist and Wahlström 2017) called *Mal'tsev embeddings*. A language Γ over D *embeds* in a language $\hat{\Gamma}$ over $\hat{D} \supset D$ if there exists a bijective function $h : \Gamma \to \hat{\Gamma}$ such that for each $R \in \Gamma$ of arity r, we have that $R = h(R) \cap D^r$. If Γ embeds in a Mal'tsev language whose domain is finite, we say that Γ has a (finite) *Mal'tsev embedding*. For instance, in Example 2 the language $\{R_k\}$ embeds in $\{R^*\}$, which is Mal'tsev.

Every language with a Mal'tsev embedding has a straightforward Mal'tsev derivation: if Γ is a language over a domain D that embeds into a Mal'tsev language Γ' of arity at most r, then it has a c-definition over $\hat{\Gamma} = \Gamma' \cup \{D^k : k \leq r\}$, in which each relation is Mal'tsev. Could the two notions coincide? Building on ideas from (Lagerkvist and Wahlström 2017), we can deduce a partial converse.

Observation 12. *If Γ has a Mal'tsev derivation, then it has an infinite-domain Mal'tsev embedding.*

Proof sketch. Theorem 28 in (Lagerkvist and Wahlström 2017) can be generalised to arbitrary domains to show that (i) infinite-domain Mal'tsev embeddings can be characterised in terms of partial polymorphisms only, and (ii) for any two languages Γ_1, Γ_2 with Mal'tsev embeddings, there exists an infinite domain D_∞ and a Mal'tsev operation u over D_∞ such that both Γ_1 and Γ_2 can be embedded in a language over D_∞ with polymorphism u. From (i) and (ii), we obtain that infinite-domain Mal'tsev embeddings are invariant under c-definitions and language unions. □

5 A dichotomy for binary languages

The results of Sections 3 and 4 raise interesting questions. Are there languages learnable with a quasi-linear number of partial membership queries but whose chain length is not linear? Is there a *gap*, in the sense that a language not learnable with $O(n \log(n))$ Boolean queries is not learnable with $o(n^c)$ Boolean queries of any kind, for some $c > 1$?

In this section we prove the following theorem, which answers both questions for *binary* languages. A binary relation R is *rectangular* if there does not exist four (not necessarily distinct) values a, b, c, d such that $(a,b), (c,d), (a,d) \in R$ but $(c,b) \notin R$. A language is rectangular if each of its relations is rectangular.

Theorem 13. *Let Γ be a binary language. If Γ is rectangular, then it has linear chain length and is learnable with $O(n \log(n))$ partial membership queries. Otherwise, Γ is not learnable with $o(n^2)$ Boolean queries.*

We first prove that every rectangular relation has a Mal'tsev derivation. To this purpose, we will need two known facts about rectangular relations.

Lemma 14 ((Kazda 2011), Lemma 10)**.** *Let R be a rectangular relation. If we define the relations $\sim_1$, $\sim_2$ over D as*

$$d \sim_1 d' \iff \exists d^* \in D \mid (d, d^*) \in R \text{ and } (d', d^*) \in R$$
$$d \sim_2 d' \iff \exists d^* \in D \mid (d^*, d) \in R \text{ and } (d^*, d') \in R$$

then $\sim_1$ and $\sim_2$ are equivalence relations on the sets $\{d \in D \mid \exists d^* \in D, (d, d^*) \in R\}$ and $\{d \in D \mid \exists d^* \in D, (d^*, d) \in R\}$, respectively.

Furthermore, if we denote by E_1, E_2 the equivalence classes of $\sim_1$, $\sim_2$ then the mapping $h : E_1 \to E_2$ given by

$$h(e) = f \iff \exists d_e \in e,\, d_f \in f \,:\, (d_e, d_f) \in R$$

is a bijection from E_1 to E_2.

Lemma 15 ((Kazda 2011), Lemma 15)**.** *Let R be a rectangular relation and $\sim_1$ be as in Lemma 14. Let R^+ be the relation whose domain is the set E_1 of equivalence classes of $\sim_1$, and such that*

$$(e, f) \in R^+ \iff \exists d_1 \in e, d_2 \in f \,:\, (d_1, d_2) \in R.$$

Then, R is Mal'tsev if and only if R^+ is Mal'tsev.

Proposition 16. *Let Γ be a binary language. If Γ is rectangular, then it has a Mal'tsev derivation.*

Proof. Let D be the domain of Γ. We will prove that the language $\{R\}$ embeds in a Mal'tsev language $\{\hat{R}\}$ for every $R \in \Gamma$. The claim will then follow from the fact that Γ has a c-definition over $\hat{\Gamma} = \{\hat{R} : R \in \Gamma\} \cup \{D^2\}$, in which every relation is Mal'tsev. Let R be a (rectangular) relation in Γ and $\sim_1$, $\sim_2$, E_1, E_2, h be as in Lemma 14. For each ordered pair $(e, f) \in (E_1)^2$ we introduce a fresh domain value α_{ef}. Then, we define

$$S_1 = \{(\alpha_{ef}, d^*) \mid (e,f) \in (E_1)^2,\, d^* \in h(e)\}$$
$$S_2 = \{(d^*, \alpha_{ef}) \mid (e,f) \in (E_1)^2,\, d^* \in f\}$$
$$S_3 = \{(\alpha_{ef}, \alpha_{ge}) \mid (e,f,g) \in (E_1)^3\}$$

and a relation $\hat{R} = R \cup S_1 \cup S_2 \cup S_3$, whose domain is denoted by $\hat{D}$. By construction, $\hat{R} \cap D^2 = R$.

We claim that $\hat{R}$ is rectangular. It follows from the definitions of S_1, S_2 and S_3 that for any $\alpha_{ef} \in \hat{D} \backslash D$, $\hat{d} \in \hat{D}$ and $d_e \in e$, $(\alpha_{ef}, \hat{d})$ is a tuple of $\hat{R}$ if and only if $(d_e, \hat{d})$ is a tuple of $\hat{R}$. Symmetrically, for any $\alpha_{ef} \in \hat{D} \backslash D$, $\hat{d} \in \hat{D}$ and $d_{h(f)} \in h(f)$, $(\hat{d}, \alpha_{ef})$ is a tuple of $\hat{R}$ if and only if $(\hat{d}, d_{h(f)})$ is a tuple of $\hat{R}$. Towards a contradiction, let $(a,b,c,d) \in \hat{D}$ be such that $(a,b), (c,d), (a,d) \in \hat{R}$ but $(c,b) \notin \hat{R}$. By the observation above, if any of a, b, c, d belongs to $\hat{D} \backslash D$ then it can be replaced by a value in D while still having $(a,b), (c,d), (a,d) \in \hat{R}$ and $(c,b) \notin \hat{R}$. If we replace them all, then we obtain values $(a,b,c,d) \in D$ such

that $(a,b),(c,d),(a,d) \in \hat{R}$ but $(c,b) \notin \hat{R}$, which is impossible since $R = \hat{R} \cap D^2$ is rectangular.

Since $\hat{R}$ is rectangular, we can define the relation $\hat{\sim}_1$ as in Lemma 14 and $\hat{R}^+$ as in Lemma 15, whose domain is the set $\hat{E}$ of equivalence classes of $\hat{\sim}_1$. By construction we have

$$\hat{E} = \{e \cup \{\alpha_{ef} \mid f \in E\} \mid e \in E\}$$

and $(\alpha_{ef}, \alpha_{fe}) \in \hat{R}$ for all $f, e \in E$. Then, for all $\hat{e}, \hat{f} \in \hat{E}$ we have $(\hat{e}, \hat{f}) \in \hat{R}^+$ since $(\alpha_{ef}, \alpha_{fe}) \in \hat{R}$, $\alpha_{ef} \in \hat{e}$ and $\alpha_{fe} \in \hat{f}$. Therefore $\hat{R}^+ = \hat{E}^2$, which is Mal'tsev as it admits all operations on $\hat{E}$ as polymorphisms. By Lemma 15, $\hat{R}$ is Mal'tsev as well, and hence $\{R\}$ has a Mal'tsev embedding. This is true for all $R \in \Gamma$, so Γ has a Mal'tsev derivation over $\hat{\Gamma} = \{\hat{R} : R \in \Gamma\} \cup \{D^2\}$. □

Proposition 17. *Let Γ be a binary language. If Γ is not rectangular, then $VC_\Gamma(n) = \Omega(n^2)$.*

Proof. Let $R \in \Gamma$ be a non-rectangular relation. By definition, there exist three tuples (a,b), (c,d), $(a,d) \in R$ such that $(c,b) \notin R$. Now, let n be an even positive integer. For each pair (i,j) with $1 \leq i \leq n/2$ and $n/2 < j \leq n$, let $t_{(i,j)}$ denote the n-tuple given by $t[i] = c$, $t[j] = b$, and for all $k \in \{1,\dots,n\}$ different from i,j we have $t[k] = a$ if $k \leq n/2$ and $t[k] = d$ otherwise. We claim that Γ shatters the relation $R_n = \{t_{(i,j)} \mid (i,j) \in \{1,\dots,n/2\} \times \{n/2+1,\dots,n\}\}$.

Let R_n^* be an arbitrary subrelation of R_n. Let $I^* = (X, C^*)$ be a CSP instance such that $C^* = \{(R,(x_i,x_j)) \mid t_{(i,j)} \notin R_n^*\}$. By construction, an assignment ϕ to X such that $\phi(x_k) \in \{a,c\}$ for $k \leq n/2$ and $\phi(x_k) \in \{b,d\}$ otherwise is a solution to I^* if and only if there is no constraint $(R,(x_i,x_j)) \in C^*$ such that $\phi(x_i) = c$ and $\phi(x_j) = b$. It follows that $t_{(i,j)} \in \mathrm{sol}(I^*)$ if and only if $(R,(x_i,x_j)) \notin C^*$, and by definition of I^* we have $(R,(x_i,x_j)) \notin C^*$ if and only if $t_{(i,j)} \in R_n^*$. Putting everything together we get that $\mathrm{sol}(I^*) \cap R_n = R_n^*$. Such an instance I^* exists for all choices of subrelation R_n^*, so Γ shatters R_n for every even positive integer n. The relation R_n contains $(n/2)^2 = \Omega(n^2)$ tuples, so the claim follows. □

Proof (of Theorem 13). If Γ is rectangular, then by Proposition 16 it has a Mal'tsev derivation. By Proposition 10 we have $\mathrm{CL}_\Gamma = O(n)$ and Γ is learnable in $O(n\log(n))$ partial membership queries by Theorem 7. Otherwise, by Proposition 17 $\mathrm{VC}_\Gamma = \Omega(n^2)$ and by Observation 3 any acquisition algorithm for Γ must make $\Omega(n^2)$ Boolean queries. □

6 Beyond binary languages

Theorem 13 shows an interesting dichotomy for binary languages: either Γ has a Mal'tsev derivation, or its VC-dimension is quadratic. In this section we show that the situation is more complex for ternary languages already as this dichotomy no longer exists (Theorem 18 and Example 3).

Given a domain D, we define p^* as the ternary partial operation on D with $\mathrm{domain}(p^*) = \{(d_1,d_2,d_3) \in D^3 : |\{d_1,d_2,d_3\}| \leq 2\}$ and satisfying $p^*(x,x,y) = p^*(y,x,x) = y$ and $p^*(x,y,x) = x$ for all $x,y \in D$. These three identities are well known and called the *Pixley identities*; in the terminology of (Lagerkvist and Wahlström 2017) p^* would be called the *first partial Pixley operation* on D. The main result of this section is the following.

Theorem 18. *Let Γ be a ternary language. If p^* is a partial polymorphism of Γ, then $VC_\Gamma(n)$ is $o(n^2)$.*

We will prove Theorem 18 indirectly, using combinatorial arguments to show that every CSP instance over Γ contains at most $o(n^2)$ non-redundant constraints. Observation 4 will then bridge the gap between non-redundancy and VC-dimension. By Observation 5, it suffices to prove Theorem 18 in the case where Γ contains a single relation. For the rest of this section we fix a ternary language $\Gamma = \{R\}$ over D which admits p^* as a partial polymorphism.

The next lemma establishes a simple but important property that R shares with rectangular relations. As in Section 5, we let $\sim_1$, $\sim_2$ and $\sim_3$ be three binary relations over D given by:

$$d \sim_1 d' \iff \exists d_2,d_3 \in D \mid (d,d_2,d_3),(d',d_2,d_3) \in R$$
$$d \sim_2 d' \iff \exists d_1,d_3 \in D \mid (d_1,d,d_3),(d_1,d',d_3) \in R$$
$$d \sim_3 d' \iff \exists d_1,d_2 \in D \mid (d_1,d_2,d),(d_1,d_2,d') \in R$$

Lemma 19. *$\sim_1$, $\sim_2$ and $\sim_3$ are equivalence relations over $\{d \in D \mid \exists (d,d_2,d_3) \in R\}$, $\{d \in D \mid \exists (d_1,d,d_3) \in R\}$ and $\{d \in D \mid \exists (d_1,d_2,d) \in R\}$, respectively.*

Proof. We prove the lemma for $\sim_1$. The cases of $\sim_2$ and $\sim_3$ are identical. By definition $\sim_1$ is reflexive and symmetric, so we need only prove transivity. Let d,d',d'' be such that $d \sim_1 d'$ and $d' \sim_1 d''$. By definition, there exist d_1,d_2 such that $t_1 = (d,d_1,d_2) \in R$, $t_2 = (d',d_1,d_2) \in R$, and d_3,d_4 such that $t_3 = (d',d_3,d_4) \in R$, $t_4 = (d'',d_3,d_4) \in R$. From the identities of p^* we get $p^*(t_2,t_3,t_4) = (d'',d_1,d_2)$ so $d \sim_1 d''$ and $\sim_1$ is transitive. □

Let $I = (X,C)$ be a CSP instance over Γ. For $i \in \{1,2,3\}$ we let $X_i = \{x_j^i \mid x_j \in X\}$. We let J_I denote the CSP instance (X_J, C_J) over $\Gamma \cup \{=\}$, where $X_J = X_1 \cup X_2 \cup X_3$ and C_J contains a constraint $c_J = (R,(x_i^1,x_j^2,x_k^3))$ for each $c = (R,(x_i,x_j,x_k)) \in C$ as well as two constraints $(=,(x_j^1,x_j^2))$, $(=,(x_j^2,x_j^3))$ for every $x_j \in X$.

Lemma 20. *Let c be a constraint in C. If c_J is redundant in J_I, then c is redundant in I.*

Proof. We prove the contrapositive. Suppose that c is not redundant in I. Then, there exists an assignment $\phi : X \to D$ such that ϕ violates c but satisfies every $c' \in C$, $c' \neq c$. If we define $\phi_J : X_J \to D$ such that $\phi_J(x_j^k) = \phi(x_j)$ for all $x_j^k \in X_J$, then for any $c' \in C$ we have $\phi_J(x_i^1,x_j^2,x_k^3) = \phi(x_i,x_j,x_k)$, so ϕ_J satisfies c'_J if and only if $c' \neq c$. Since ϕ_J also satisfies the equalities, c_J is not redundant. □

By Lemma 20 we can reduce the proof of Theorem 18 to finding a bound on the number of non-redundant constraints in J_I. To this purpose, we will turn J_I into a reduced instance J_I^*, which has the same solution set but fewer constraints; any constraint that does not appear in the reduced instance will be redundant.

We propose the following transformation T. If there exists three constraints $c_1 = (R, (x_1, y_1, z_1))$, $c_2 = (R, (x_1, y_2, z_2))$ and $c_3 = (R, (x_2, y_2, z_1))$ (of which at least 2 are distinct) then we remove c_2, c_3 from the instance and instead we add three constraints $(\sim_1, (x_1, x_2))$, $(\sim_2, (y_1, y_2))$ and $(\sim_3, (z_1, z_2))$.

Lemma 21. *Applying the transformation T leaves the solution set unchanged.*

Proof. Let $c_1 = (R, (x_1, y_1, z_1))$, $c_2 = (R, (x_1, y_2, z_2))$ and $c_3 = (R, (x_2, y_2, z_1))$ be the three constraints on which T will be applied. We first prove that the constraints $(\sim_1, (x_1, x_2))$, $(\sim_2, (y_1, y_2))$ and $(\sim_3, (z_1, z_2))$ are implied by c_1, c_2 and c_3. Suppose that ϕ is a solution to the instance before T is applied. Then, from the constraints c_1, c_2, c_3 we deduce that R contains the tuples $t_1 = (\phi(x_1), \phi(y_1), \phi(z_1))$, $t_2 = (\phi(x_1), \phi(y_2), \phi(z_2))$ and $t_3 = (\phi(x_2), \phi(y_2), \phi(z_1))$. From the identities of p^* we get $p^*(t_1, t_2, t_3) = (\phi(x_2), \phi(y_1), \phi(z_1)) \in R$, which imply $x_1 \sim_1 x_2$ from t_1 and $y_1 \sim_2 y_2$ from t_3. Since $y_1 \sim_2 y_2$ we deduce from t_1 that $(\phi(x_1), \phi(y_2), \phi(z_1)) \in R$, and together with t_2 we obtain $z_1 \sim_3 z_2$. Hence, adding the three equivalence constraints does not remove any solution.

Next, we prove that the four constraints c_1, $(\sim_1, (x_1, x_2))$, $(\sim_2, (y_1, y_2))$ and $(\sim_3, (z_1, z_2))$ together imply c_2 and c_3. If ϕ is a solution to the instance after T has been applied, then we have $(\phi(x_1), \phi(y_1), \phi(z_1)) \in R$. Since $\phi(y_1) \sim_2 \phi(y_2)$ and $\phi(z_1) \sim_3 \phi(z_2)$, we have that $(\phi(x_1), \phi(y_2), \phi(z_2)) \in R$ and c_2 is satisfied. The case of c_3 is symmetric. Putting everything together we get that replacing the constraints c_1, c_2, c_3 with c_1, $(\sim_1, (x_1, x_2))$, $(\sim_2, (y_1, y_2))$ and $(\sim_3, (z_1, z_2))$ leaves the solution set unchanged. □

The transformation T may seem odd, because it removes two constraints but adds *three* new constraints each time. The reason why we are making progress is that the introduced constraints are equivalence relations, among which at most a linear number are non-redundant.

Proof of Theorem 18. We establish the theorem for the case where Γ contains a single relation R. The result will then follow from Observation 5. Let I be a CSP instance over Γ and $J_I = (X_J, C_J)$ be the associated instance over $\Gamma \cup \{=\}$. We apply the transformation T to J_I until fixed point, and then remove all redundant equalities and equivalence constraints. (Note that T removes at least one ternary constraint at each step, so a fixed point will eventually be reached.) The resulting instance J_I^* contains $O(n)$ equalities and equivalence constraints, and a yet unknown number k of ternary constraints with relation R. If we define a graph $G = (V, E)$ such that $V = X_J$ and $\{x, y\} \in E$ if and only if there exists a constraint $c = (R, S_c)$ in J_I^* such that both x and y appear in S_c, then we have $|E| = 3k$, since no two ternary constraints can share two variables unless T is applicable. Furthermore, every adjacent pair of vertices are contained in a unique triangle (given by the unique constraint c that contains the two variables) unless T is applicable. (If another triangle exists, then its two extra edges belong to distinct constraints c', c''. Then, T is applicable on $\{c, c', c''\}$ because each variable involved in two constraints must be at the same position in their respective scopes, by definition of J_I.) Therefore G is locally linear, and by (Ruzsa and Szemerédi 1978) G has $o(n^2)$ edges. It follows that $k = o(n^2)$ and J_I^* has $o(n^2)$ constraints.

In order to carry this bound to the number of non-redundant constraints in J_I, observe that every equivalence relation in J_I^* is implied by at most three ternary constraints of J_I. Therefore, if we replace each equivalence constraint in J_I with these ternary constraints, we obtain an instance with $o(n^2)$ constraints, all of which belong to C_J, and whose solution set is by Lemma 21 identical to that of J_I. It follows that J_I has $o(n^2)$ non-redundant constraints, and then by Lemma 20 I has $o(n^2)$ non-redundant constraints. □

Example 3. Consider the ternary relation

$$R = \begin{bmatrix} A & A & B \\ A & C & A \\ B & C & C \\ C & A & C \\ C & B & A \end{bmatrix}$$

over the domain $D = \{A, B, C\}$. For any three distinct tuples $t_1, t_2, t_3 \in R$ there exists $i \in \{1, 2, 3\}$ such that $t_1[i], t_2[i]$ and $t_3[i]$ are three different values, and hence $(t_1[i], t_2[i], t_3[i]) \notin \text{domain}(p^*)$. Furthermore, for any two (not necessarily distinct) tuples $t_1, t_2 \in R$ we have $p^*(t_1, t_1, t_2) = t_2$, $p^*(t_2, t_1, t_1) = t_2$ and $p^*(t_1, t_2, t_1) = t_1$. Therefore, p^* is a partial polymorphism of R and $\text{VC}_{\{R\}}(n) = o(n^2)$ by Theorem 18.

However, $\{R\}$ does not have a Mal'tsev embedding, even on an infinite domain; it follows from Observation 12 that it does not have a Mal'tsev derivation either. To see this, suppose that R embeds in a relation $\hat{R}$ over a (possibly infinite) domain $\hat{D}$ with a Mal'tsev polymorphism f. Then, the operation g given by $g(x_1, \ldots, x_5) = f(x_1, f(x_2, x_3, f(x_1, x_2, x_3)), f(x_5, x_4, f(x_1, x_2, x_3)))$ over $\hat{D}$ is also a polymorphism of $\hat{R}$. Applying g to the five tuples of R (from top to bottom) we obtain (B, B, B), which belongs to $\hat{R} \cap D^3$ but not to R, a contradiction.

As a final remark, we note (without proof, due to space constraints) that the relation R of Example 3 is NP-hard. The proof of Theorem 18 gives a polynomial-time algorithm that transforms any CSP instance over $\{R\}$ into an equivalent one with $o(n^2)$ constraints, or, in other words, a *kernel* with $o(n^2)$ constraints.

7 Conclusion

We have shown that the existence of a Mal'tsev derivation implies linear chain length, and that linear chain length implies efficient learnability with partial membership queries. For binary languages, by Theorem 13 these properties are equivalent. Could this also be true in the general case? Does there exist languages with linear chain length but no Mal'tsev derivation, or languages that are efficiently learnable despite having superlinear chain length?

References

Angluin, D. 1988. Queries and concept learning. *Machine Learning* 2(4):319–342.

Barto, L.; Krokhin, A. A.; and Willard, R. 2017. Polymorphisms, and how to use them. In *The Constraint Satisfaction Problem: Complexity and Approximability*. 1–44.

Beldiceanu, N., and Simonis, H. 2012. A model seeker: Extracting global constraint models from positive examples. In *Proceedings of the Eighteenth International Conference on Principles and Practice of Constraint Programming (CP'12)*, 141–157.

Bessiere, C.; Coletta, R.; Koriche, F.; and O'Sullivan, B. 2005. A SAT-based version space algorithm for acquiring constraint satisfaction problems. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML'05)*, 23–34.

Bessiere, C.; Coletta, R.; Hebrard, E.; Katsirelos, G.; Lazaar, N.; Narodytska, N.; Quimper, C.; and Walsh, T. 2013. Constraint acquisition via partial queries. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 475–481.

Bessiere, C.; Koriche, F.; Lazaar, N.; and O'Sullivan, B. 2017. Constraint acquisition. *Artif. Intell.* 244:315–342.

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

Couceiro, M.; Haddad, L.; and Lagerkvist, V. 2019. Fine-grained complexity of constraint satisfaction problems through partial polymorphisms: A survey. In *Proceedings of the Forty-Ninth International Symposium on Multiple-Valued Logic (ISMVL'19)*, 170–175.

Hooker, J. N., and van Hoeve, W. J. 2018. Constraint programming and operations research. *Constraints* 23(2):172–195.

Idziak, P. M.; Markovic, P.; McKenzie, R.; Valeriote, M.; and Willard, R. 2010. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.* 39(7):3023–3037.

Jonsson, P.; Lagerkvist, V.; and Roy, B. 2017. Time complexity of constraint satisfaction via universal algebra. In *Proceedings of the Forty-Second International Symposium on Mathematical Foundations of Computer Science (MFCS'17)*, 17:1–17:15.

Kazda, A. 2011. Maltsev digraphs have a majority polymorphism. *European Journal of Combinatorics* 32(3):390–397.

Lagerkvist, V., and Wahlström, M. 2017. Kernelization of constraint satisfaction problems: A study through universal algebra. In *Proceedings of the Twenty-Third International Conference on Principles and Practice of Constraint Programming (CP'17)*, 157–171.

Lagerkvist, V., and Wahlström, M. 2017. Kernelization of constraint satisfaction problems: A study through universal algebra. *arXiv:1706.05941, rev. 1*.

Négrevergne, B., and Guns, T. 2015. Constraint-based sequence mining using constraint programming. In *Proceedings of the Twelfth International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR'15)*, 288–305.

Ruzsa, I. Z., and Szemerédi, E. 1978. Triple systems with no six points carrying three triangles. *Combinatorics* 18:939–945.

Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP'98)*, 417–431.

van Beek, P., and Chen, X. 1999. Cplan: A constraint programming approach to planning. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI-IAAI'99)*, 585–590.