Closure Functions and Width 1 Problems

Víctor Dalmau[1**] and Justin Pearson[2]

[1] Departament LSI, Universitat Politècnica de Catalunya,
Mòdul C5. Jordi Girona Salgado 1-3. Barcelona 08034, Spain,
dalmau@lsi.upc.es

[2] Department of Computer Systems, Uppsala University,
Box 325, S-751 05 Uppsala, Sweden
justin@DoCS.UU.SE

Abstract. Local Consistency has proven to be an important notion in the study of constraint satisfaction problems. We give an algebraic condition that characterizes all the constraint types for which generalized arc-consistency is sufficient to ensure the existence of a solution. We give some examples to illustrate the application of this result.

1 Introduction

The constraint satisfaction problem provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in artificial intelligence and elsewhere. A constraint satisfaction problem is represented by a set of variables, a domain of values for each variable, and a set of constraints between variables. Generally, a constraint C is represented by two components: a *scope* S that expresses the set of variables constrained and a *relation* R which expresses the combination of values allowed for those variables. The aim of a constraint satisfaction problem is then to find an assignment of values to the variables that satisfies the constraints.

Since solving a general constraint satisfaction problem is known to be NP-complete [1, 20], a natural and important question is: what restrictions to the general problem are sufficient to ensure tractability. Such restrictions may either involve the scope in the constraints, i.e., which variables may be constrained with other variables [6, 11, 12, 21, 22], or they may involve the relation in the constraints, in other words, which combinations of values are allowed for values which are mutually constrained [3, 15, 18, 19, 21, 27, 28].

In this paper, we focus in the second approach. More precisely we are interested in characterizing the complexity of solving constraint satisfaction problems in which every constraint belongs to a fixed set called *basis*. Jeavons, Cohen and

** This work has been partly supported by: the Spanish Government through the DGES under project PB95-0787 (KOALA), the U.K. Government through the EPSRC under grant GR/L099636, the Catalan Government through CIRIT under the project 1997SGR-00366 (SGR), and the European Commission through the ESPRIT BRA Program under the following projects: Working Group 27150 (NeuroCOLT2), project 20244 (ALCOM IT) and project TIC97-1475-CE (CICYT).

Gyssens in [15,16] introduced a novel framework for studying this class of constraint satisfaction problems. The approach started in [15,16] relies on the fact that the tractability of constraint satisfaction problems using certain relations depends on some algebraic condition (that of closure) of the set of relations used to build the problem. This approach has lead to the identification of several tractable classes [15–18,14].

Local consistency methods [20,9,10,14,7,2] have been intensively studied as a fundamental tool for solving constraint problems. Briefly, the underlying idea in local consistency methods is the following: when a constraint network is inconsistent, this is probably due to the existence of an inconsistent subnetwork of reduced size. It is sometimes possible to decide the existence of a solution of a constraint satisfaction problem by checking the consistency of all the subproblems up to a certain size. Since there exists problems which can not be solved by local consistency methods, there is a large body of work [11,5,7,14] which derives conditions that guarantee that a problem can be solved by a certain local consistency method.

In this paper we consider a family of local consistency methods called (j,k)-consistency, derived from the concept of bounded width Datalog programs [8]. This notion of consistency is related to the notion of consistency defined in [11]. We refer to the class of problems solvable by enforcing (j,k)-consistency as width (j,k) problems. It is not know, in general, whether a constraint satisfaction problem has width (j,k).

In particular, we will be interested in $(1,k)$-consistency, which can be regarded as the natural generalization of arc-consistency [20] to non-binary problems. The class of problems that can be solved by enforcing $(1,k)$-consistency for some fixed k is called width 1. This class contains some previously known tractable families of problems including Horn, constant and ACI Problems [25,17]. The main result of this paper is a characterization of width 1 problems in terms of closure functions. That is, we present an algebraic condition (in the sense of [17]) on the relations used to build constraint problems which ensure that any problem built using these relations can be solved in polynomial time by enforcing $(1,k)$-consistency for some k. Furthermore, we show that this condition is also necessary.

With this new characterization we revisit some already known tractable families, as constant and ACI problems [17], and we see that they fit into this common scheme, in other words, we prove that they are particular case of width 1 problems. Furthermore, we derive a new tractable class called CSCI (from Constant Semiprojection Commutative Idempotent) using purely algebraic arguments.

The motivation of these results is an alternative characterization of width 1 problems due to Feder and Vardi [8]. In [8] three main families of tractable problems, referred to as *subgroup*, *bounded strict width* and *width* 1 problems are identified using concepts from Database theory (Datalog programs) and group theory. These three classes include all previously known tractable problems. In [14], the family of strict width problems was characterized in terms of closure functions. An important subclass of subgroup problems called *affine* problems was introduced in [17] and was generalized to the whole class of subgroup problems in [4] studying the learnability of quantified formulas. The results in this paper characterize the remaining class, completing the research work started in [14].

2 Preliminaries and Definitions

2.1 Constraint Satisfaction Problems

A constraint satisfaction problem is a natural way to express simultaneous requirements for values of variables. More precisely,

Definition 1. An instance of a *constraint satisfaction problem* consists of:

- a finite set of variables, V. For simplicity we assume $V = \{x_1, x_2, \ldots x_n\}$;
- a finite domain of values, D;
- a finite set of constraints $\{C_1, \ldots, C_q\}$; each constraint C_i $(1 \leq i \leq q)$ is a pair (s_i, R_i) where:
  - s_i is a tuple of variables of length k_i, called the *constraint scope*; and
  - R_i is an k_i-ary relation over D, called the *constraint relation.*

For each constraint (s_i, R_i), the tuples in R_i indicate the allowed combinations of simultaneous values for the variables in s_i. The length of s_i, and of the tuples in R_i, is called the *arity* of the constraint.

A *solution* to a constraint satisfaction problem instance is a function from the variables to the domain such that the image of each constraint scope is an element of the corresponding constraint relation. Deciding whether or not a given problem instance has a solution is NP-complete in general, even when the constraints are restricted to binary constraints [20] or the domain of the problem has size 2 [1]. However, by imposing restrictions on the constraint interconnections (see [6, 11, 12, 21, 22]), or the form of the constraints (see [3, 15, 18, 19, 21, 27, 28]), it is possible to obtain restricted versions of the problem that are tractable. As we have said, the aim underlying this research is to determine all the possible restrictions on the form of the constraints that ensure tractability. The central object of study is the set of problems defined from some fixed set (or basis), and the associated complexity of the constraint problems in that set.

Definition 2. For any set of relations Γ, C_Γ is defined to be the class of decision problems with:

- Instance: A constraint satisfaction problem instance $\mathcal{P}$, in which all constraint relations are elements of Γ.
- Question: Does $\mathcal{P}$ have a solution?

2.2 Closure Functions

To our knowledge even though there exists a large collection of individual results about the complexity of C_Γ, there has only been two attempts to study the complexity of C_Γ in a uniform way. The first attempt, due to Feder and Vardi [8], uses an approach based in Datalog Programs and Group Theory and produced, among other important results, a classification of the known tractable classes in three families of basis, called *width* 1, *bounded strict width*, and *subgroup problems*. The second attempt was originally defined by Jeavons et al. [15, 16] and focuses on the algebraic properties of constraints. We follow the second approach in this paper. In the rest of this section we will introduce the basic concepts of this approach (see [15–17, 14] for more information).

Any operation on the elements of a set D can be extended to an operation on tuples over D by applying the operation to the values in each coordinate position separately.

Definition 3. Let $f : D^k \to D$ be a k-ary operation on D. For any collection of n-ary tuples $t_1, t_2, \ldots, t_k \in D^n$, (not necessarily all distinct) define the tuple $f(t_1, t_2, \ldots, t_k)$ as follows:

$$f(t_1, \ldots, t_k) = f(t_1[1], \ldots, t_k[1]), f(t_1[2], \ldots, t_k[2]), \ldots, f(t_1[n], \ldots, t_k[n])\rangle.$$

Using this definition, we now define the following closure property of relations.

Definition 4. Let R be a relation over a domain D, and let $f : D^k \to D$ be a k-ary operation on D. R is said to be *closed* under f (f preserves R, or f is a polymorphism of R) if, for all $t_1, t_2, \ldots, t_k \in R$ (not necessarily all distinct),

$$f(t_1, t_2, \ldots, t_k) \in R.$$

We say that an operation f preserves a basis Γ if f preserves every relation in Γ. The next lemma indicates that the property of being closed under some operation is preserved by some operations on relations.

Lemma 1. *Let R_1 be an n-ary relation over a domain D and let R_2 be a m-ary relation over a domain D. Both R_1 and R_2 are closed under some operation f. The following relations are also closed under f.*

- *The* Cartesian Product, *$R_1 \times R_2$ defined to be the $(n+m)$-ary relation*

$$\begin{aligned} R_1 \times R_2 = \{\langle t[1], t[2], \ldots, t[n+m]\rangle \mid \langle t[1], t[2], \ldots, t[n]\rangle \in R_1 \ \wedge \\ \langle t[n+1], t[n+2], \ldots, t[n+m]\rangle \in R_2\} \end{aligned}$$

- *The* equality selection *$\sigma_{i=j}(R)$ $(1 \le i, j \le n)$ defined to be the n-ary relation*

$$\sigma_{i=j}(R) = \{t \in R | t[i] = t[j]\}$$

- *The* projection $\pi_{i_1,\ldots,i_k}(R)$ *where* $(i_1,\ldots,i_k)$ *is a list of indices chosen from* $\{1,2,\ldots,n\}$*, defined to be the* k*-ary relation*

$$\pi_{i_1,\ldots,i_k}(R) = \{\langle t[i_1],\ldots,t[i_k]\rangle | t \in R\}$$

Proof. Follows directly from the definitions. ∎

The set of all relations that which can be obtained from a given set of relations Γ, using some sequence of Cartesian product, equality selection, and projection operations will be denoted Γ^+. Consequently every operation preserving Γ, preserves Γ^+ as well.

On the other hand, the property of being closed under some operation is preserved by some operations on functions.

Lemma 2. *Let* $g : D^m \to D$ *be a* m*-ary function and let* $f_1, f_2, \ldots, f_m : D^n \to D$ *be* n*-ary functions. If* $g, f_1, f_2, \ldots, f_m$ *preserve* R*, for some relation* R*, then the following functions preserve also* R*:*

- *The* composition $g(f_1, f_2, \ldots, f_m)$ *defined to be the* n*-ary operation*

$$g(f_1, f_2, \ldots, f_m)(x_1,\ldots,x_n) = g(f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n))$$

- *For every* $j \leq k$*, the* projection $\text{proj}_{,j,k}$ *defined to be the* k*-ary operation*

$$\text{proj}_{j,k}(x_1, x_2, \ldots, x_k) = x_j$$

Proof. Follows from the definitions. ∎

Any set of operations closed under composition and containing all the projection operations is called *Clone* (See [26] for example). In [13], it was established that the complexity of C_Γ is determined by the set of closure functions of Γ. We say that a closure function guarantees tractability if for every basis Γ closed under f, class C_Γ is tractable. Following this approach, some functions that guarantee tractability have been identified. Up to the present moment four families of tractable functions are know. They correspond to the near-unanimity operations [14], coset-generating operations [4], ACI operations [17], and constant operations [17].

2.3 Local Consistency

Local consistency methods are refutation methods, similar in aim to the resolution method for propositional formulas in conjunctive normal form. Different notions of local consistency have been defined [20, 9, 10, 14, 7] but all of them fit into this common scheme: A Local consistency method takes an instance $\mathcal{P}$ and adds all the constraints that appear implicitly in $\mathcal{P}$ (and therefore, eliminating no solution) up to a certain level. A constraint $\langle s, \emptyset\rangle$ with empty constraint relation is called *empty constraint.* If during the process of enforcing consistency, some empty constraint is added, then instance $\mathcal{P}$ has obviously no solution. Unfortunately, the reciprocal is not always true, i.e., the absence of the empty constraint does not imply in general the existence of a solution. So, it is an interesting topic of research to establish under which conditions over the set of constraint relations Γ, establishing a certain level of local consistency guarantees the existence of a solution.

Definition 5. Let $\mathcal{P}$ be a constraint satisfaction instance with set of variables V, domain D, and constraint set C. For any subset W of V the *restriction of* $\mathcal{P}$ *to* W, denoted $\mathcal{P}^*_W$ is the problem instance with set of variables W and domain D, where the constraints are obtained from the constraints of $\mathcal{P}$ eliminating all the constraints with constraint scope not completely contained in W. That is, the constraint (s, R) is in $\mathcal{P}^*_W$ if and only if (s, R) is in $\mathcal{P}$ and every element of the tuple s is in W.

We now define the notion of consistency used in this paper.

Definition 6. A constraint satisfaction problem instance $\mathcal{P}$ with set of variables V is said to be (j, k)-*consistent* $(0 \leq j \leq k)$ if for any sets of variables W, W', such that $W \subseteq W' \subseteq V$, containing at most j and k variables respectively, any solution to $\mathcal{P}^*_W$ can be extended to a solution to $\mathcal{P}^*_{W \cup W'}$.

Informally, a problem is (j, k)-consistent if every partial solution on any set of at most j variables can be extended to a partial solution on any superset containing at most k variables. This notion of consistency generalizes the notion of (i, j)-consistency in [11] to constraint problems with non-binary constraints.

Since the constraint literature contains few similar notions of consistency we believe that it is appropriate to enclose here a brief discussion of the more popular notions of consistency. Informally we can say that the different notions of consistency are characterized by the way in which they define a subproblem. Roughly, consistency notions can be divided in two main types: (1) *variable-based* consistency and (2) *relation-based* consistency.

In variable-based consistency a subproblem is specified as the subproblem containing all the restrictions "concerning" a set of variables. Variable-based consistency has received more attention (see [20, 9, 10] for example) than its relational counterpart. We have different consistency notions depending on how "concerning" is formalized. Very often, the subproblem "concerning" a set of variables W is defined as the subproblem containing all the constraints with scope strictly contained in W, as in the definitions in this paper. This approach is due to [20] and was generalized to sets of variables with arbitrary size in [9]. This notion of consistency in closely related to the notion defined here. More precisely, k-consistency in the sense of [9] coincides exactly with $(k-1, k)$-consistency, as defined in this paper. In some other cases, the problem "concerning" a set of variables W has been defined as the problem containing the projection of all the constraints over W (see [14], for example). In fact, both definitions are equivalent for the purposes of this paper (this will be shown in the forthcoming full version of this paper) We choosed this particular notion of consistency to simplify the proofs. Finally, in relation-based consistency (see [7] for example), a subproblem is specified by a set of restrictions rather than variables. It is important to note that for constraint satisfaction problems with only binary constraints many of the different possible notions of consistency coincide.

Given a CSP instance $\mathcal{P}$, there exists some instance $\mathcal{P}'$ such that (1) Every constraint in $\mathcal{P}$ is in $\mathcal{P}'$, (2) $\mathcal{P}'$ is (j,k)-consistent, and (3) instances $\mathcal{P}$ and $\mathcal{P}'$ have the same set of solutions. Instance $\mathcal{P}'$ is referred to as a (j,k)-consistent instance associated to $\mathcal{P}$. We say that $\mathcal{P}'$ is obtained from $\mathcal{P}$ by enforcing (j,k)-consistency. The CSP literature [21, 20, 9, 2] contains some efficient methods to enforce certain level of consistency. Here we present simple brute-force algorithm, called $\mathrm{Cons}_{(j,k)}$ (j,k fixed), that enforces (j,k)-consistency.

Procedure $\mathrm{Cons}_{(j,k)}$
Input: $\mathcal{P}$
$\mathcal{P}' = \mathcal{P}$
repeat
    for every subset W with $|W| \leq j$
        for every superset W' of W with $|W'| \leq k$
            for every tuple $s = \langle x_1, \ldots, x_i \rangle$ with variables in W
                let R be the projection of ${\mathcal{P}'}^*_{W'}$ over s, i.e.,
                    $R = \{\langle \mu(x_1), \ldots, \mu(x_i) \rangle : \mu \text{ solution of } {\mathcal{P}'}^*_{W'}\}$
                (R contains all the assignment over s that can be extended to
                a solution of ${\mathcal{P}'}^*_{W'}$)
                if $\langle s, R \rangle$ not in $\mathcal{P}'$ then add $\langle s, R \rangle$ to $\mathcal{P}'$.
until no constraint has been added.
return $\mathcal{P}'$.

Basically, the algorithm looks for subsets $W \subseteq W'$ of variables falsifying the (j,k)-consistency condition (Definition 6) and adds the corresponding constraint until the instance $\mathcal{P}'$ satisfies the (j,k)-consistency condition and no constraint has to be added. For fixed values of j,k, procedure $\mathrm{Cons}_{(j,k)}$ runs in time polynomial to the size of the input instance $\mathcal{P}$.

Notice that the constraints added by algorithm $\mathrm{Cons}_{(j,k)}$ are minimal in the sense that for every (j,k)-consistent instance $\mathcal{P}'$ associated to $\mathcal{P}$ and every variable x_i $(1 \leq i \leq n)$, every solution of ${\mathcal{P}'}^*_{\{x_i\}}$ is a solution of $(\mathrm{Cons}_{(j,k)}(\mathcal{P}))^*_{\{x_i\}}$.

Furthermore, for every constraint $\langle s, R \rangle$ added by the algorithm, its associated relation R is the projection of subproblem ${\mathcal{P}'}^*_{W'}$ over s. Since every relation associated to a problem (or equivalently subproblem) with constraint relations in Γ can be obtained by a sequence of cartesian products and equality selections using relations in Γ. We have that if $\mathcal{P} \in C_\Gamma$, then $\mathcal{P}' \in C_{\Gamma^+}$.

Definition 7. A CSP instance $\mathcal{P}$ is said to have *width* (j,k) if $\mathcal{P}$ has a solution if and only if every (j,k)-consistent instance associated to $\mathcal{P}$ does not contain the empty constraint relation $\langle s, \emptyset \rangle$. Similarly, a set of relations Γ over D is said to have width (j,k) if every CSP instance $\mathcal{P}$ in $\mathcal{C}_\Gamma$ has width (j,k). Furthermore Γ is said to have width j if it has width (j,k) for some fixed k.

Hence, a set of relations Γ has width (j,k) if and only if every problem in $\mathcal{C}_\Gamma$ is solvable by enforcing (j,k)-consistency (using the previous algorithm for example). In consequence, any constraint instance in $\mathcal{C}_\Gamma$ is solvable in polynomial-time, since for fixed values j and k, it is possible to enforce (j,k)-consistency in time polynomial in the size of the problem.

Feder and Vardi [8] gave an alternative characterization of these concepts in terms of Datalog Programs. Datalog Programs are far beyond the scope of this paper. Nevertheless, we will only note here, that the notion of width defined above is exactly equivalent to the notion of width, as defined in [8].

3 Width 1 Problems

We are interested in the class of problems that can be solved by enforcing $(1,k)$-consistency for some fixed k, also called width 1 problems. The notion of $(1,k)$ consistency is the natural generalization of arc-consistency to non-binary problems.

In this section we prove that the family of basis of width 1 is absolutely characterized in terms of closure functions. In other words, we will see that for every basis Γ, $\mathcal{C}_\Gamma$ is solvable by enforcing $(1,k)$-consistency for some $k \geq 1$ if and only if some condition over the set of closure functions is satisfied. First we introduce the concept of set function.

A *set function* is any function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ where D is some set and $\mathcal{P}(D)$ is the set of subsets of D.

Associated to every set function f we have a *family of functions associated to* f $\{f_i : i = 1, \ldots, n\}$, where for every i, $f_i : D^i \longrightarrow D$ is given by

$$f_i(x_1, \ldots, x_i) = f(\{x_1, \ldots, x_i\}),$$

Let f be a set function over D and let R be any relation over D, we say that f *preserves* R, (or R is closed under f) if the family of functions associated to f preserves R (Definition 4).

Theorem 1. *Let Γ be a finite set of relations over D. Then the following conditions are equivalent:*

a.- The set Γ is width 1.
b.- Γ is closed under some set function.

The proof of the theorem uses the following result. Given any set of relations Γ, consider the constraint satisfaction problem $\mathcal{C}(\Gamma)$ defined as follows:

The variables of $\mathcal{C}(\Gamma)$ are the nonempty subsets A of D. For a relation R in Γ of arity k, impose $R(A_1, A_2 \ldots, A_k)$, the A_i not necessarily distinct, if for every $1 \leq i \leq k$ and every a_i in A_i there exist elements a_j in the remaining A_j such that $(a_1, a_2, \ldots, a_k) \in R$.

There is an alternative characterization of the constraints in $\mathcal{C}(\Gamma)$: It is easy to verify that a constraint $R(A_1,\ldots,A_k)$ is in $\mathcal{C}(\Gamma)$ iff there exists some assignments $t_1,\ldots,t_m$ in R such that $\{t_1[l],t_2[l],\ldots,t_m[l]\} = A_l$ for all $1 \leq l \leq k$.

Feder and Vardi proved that $\mathcal{C}(\Gamma)$ has a solution if and only if C_Γ is solvable by a $(1,k)$-Datalog program [8]. In the full version we show that C_Γ is solvable by a $(1,k)$-Datalog Program if and only if Γ is width 1. In consequence,

Theorem 2. *[8] A set of relations Γ is width 1 then $\mathcal{C}(\Gamma)$ has a solution.*

Proof of Theorem 1

[(**a**) $\to$ (**b**)]. Let h be any solution of $\mathcal{C}(\Gamma)$ (the existence of such solution is guaranteed by Theorem 2) and let f_h be the set function defined as:

$$f^h(A_i) = h(A_i).$$

We will prove that f^h preserves Γ: Let m be any integer $m > 1$, let R be any k-ary relation in Γ, and let $t_1,t_2,\ldots,t_m$ (not necessarily different) tuples in R. Let A_i $(1 \leq i \leq k)$ be subsets of D given by:

$$A_i = \{t_l[i] : 1 \leq l \leq m\}, \;\; 1 \leq i \leq k$$

By, construction, $R(A_1,A_2,\ldots,A_k)$ appears in $\mathcal{C}(\Gamma)$ and therefore h satisfies it. Then we have

$$\langle h(A_1),h(A_2),\ldots,h(A_k)\rangle = \langle f^h(A_1),f^h(A_2),\ldots,f^h(A_k)\rangle = f^h_m(t_1,\ldots,t_m) \in R.$$

[(**b**) $\to$ (**a**)]. Let Γ be a set of relations over D closed under a set function f. Let k be the maximum arity of the relations in Γ. We will prove that enforcing $(1,k)$-consistency is enough to decide satisfiability. Let $\mathcal{P}$ be any problem in C_Γ and let $\mathcal{P}'$ be a problem obtained by enforcing $(1,k)$-consistency to $\mathcal{P}$. For every variable x, let D_x be the set of values that can be assigned to the variable x, i.e., the set of solutions of the problem $\mathcal{P}'$ restricted to x: $D_x = \mathcal{P}^*_{\{x\}}$.

Consider the vector t, assigning to every variable x the value of the set function over D_x, that is,

$$t(x) = f(D_x)$$

We will see that t is a solution. Let $\langle (x_1,x_2\ldots,x_m),R\rangle$ be any constraint in $\mathcal{P}$. Clearly, $D_{x_i} \subseteq \pi_i R$ $\;(1 \leq i \leq m)$. Consider now, the subset R' of R, obtained enforcing for every variable x appearing in the scope of the constraint to have a value in D_x. More formally,

$$R' = R \cap (D_{x_1} \times \cdots \times D_{x_m}).$$

Since, D_{x_i} $(1 \leq i \leq m)$ is obtained by enforcing $(1,k)$-consistency, we have $D_{x_i} = \pi_i R'$ $\;(1 \leq i \leq m)$. Let $t_1,t_2,\ldots,t_l$ be the tuples in R'. Since R is closed under f_l, we have:

$$f_l(t_1,t_2,\ldots,t_l) = \langle f(D_{x_1}),f(D_{x_2}),\ldots,f(D_{x_m})\rangle \in R.$$

So t satisfies R. ■

The characterization of width 1 problems in terms of closure functions is interesting theoretically but it is not absolutely satisfactory. First, if we want to use this characterization to check whether a given basis Γ is width 1 then we have to test the closure condition for an infinite family of operations.

Actually it is possible, for a given Γ, fix a bound in the arity of the operations that we have to consider.

Theorem 3. *Let Γ be a finite set of relations over D, let m be the maximum number of tuples of any relation in Γ, and let f be a set function over D. If f_m preserves Γ then f preserves Γ.*

Proof.

Most of the next proofs have a similar structure. We will be interested in proving that if some function h_1 (or set of functions) preserves a relation R then another function h_2 preserves also R. We will prove that by showing that function h_2 can be obtained by a sequence of compositions using h_1 and projection operations only. Then, since the set of closure functions of a relation contains projection operations and is closed under composition (Lemma 2), operation h_2 preserves any relation preserved by h_1. We will say that h_2 is contained in any clone containing h_1.

In the present proof, we will see that every clone containing f_m, contains f_k ($k > 0$), or in other words, f_k can be obtained by a composition using f_m and projections.

It is immediate to prove that, in general, for every set function f and every $k > 0$, f_k belongs to any clone containing f_{k+1}. Just consider the identity.

$$f_k(x_1, x_2, \ldots, x_k) = f_{k+1}(x_1, x_2, \ldots, x_k, x_k)$$

For the proof in the opposite direction, let R be any relation in Γ, let k be any integer greater than m, and let $t_1, t_2, \ldots, t_k$ tuples in R. Since $k > m$, some tuples are repeated. Consider a sublist with m tuples $t_{i_1}, t_{i_2}, \ldots, t_{i_m}$ in which we have deleted only repeated tuples, i.e., such that $\{t_j : 1 \leq j \leq k\} = \{t_{i_j} : 1 \leq j \leq m\}$. By the structure of the set functions we have:

$$f_k(t_1, t_2, \ldots, t_k) = f_m(t_{i_1}, t_{i_2}, \ldots, t_{i_m}) \in R$$

■

It would be interesting to get rid of this dependence on the number of tuples, that is, to prove that it is possible to verify whether or not a basis is closed under a set function by checking only the closure property up to a fixed point. The previous assertion is true for the boolean case, since it is known that every clone in the boolean domain is finitely generated [24, 23] but it is false for higher domains. We present a counterexample.

Example 1. Let $D = \{0, 1, 2, \ldots, d\}$ be a domain with size $|D| > 2$. Consider the set function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ defined by:

$$f(S) = \begin{cases} 0 \text{ if } \ S = \{0, 1\} \\ 2 \text{ otherwise} \end{cases}$$

For every m, consider the m-ary relation R_m containing all the tuples such that either (1) exactly one of its components is 1 and the remaining $m-1$ are 0, or (2) at least one of its components is 2. Clearly, f_k preserves R_m if and only if $k < m$. As a consequence, for any value of k there exists some basis given by $\Gamma_k = \{R_{k+1}\}$ such that f_k preserves Γ_k but f does not.

4 Applications of the closure conditions

Using the characterization of width 1 problems established in Theorem 1 it is possible reformulate some already known tractable classes and derive new ones. From now on, we will say that a function (or set of functions) *guarantees tractability* if for every basis Γ closed under it, C_Γ is tractable.

4.1 Constant operations

An unary function f_1 mapping every element to a fixed element $d \in D$ is called *constant function.* Every relation closed under f_1 contains the tuple $\langle d, d, \ldots, d\rangle$ and therefore operation f_1 guarantees tractability, i.e., for every Γ closed under f_1, C_Γ is tractable. Constant operations correspond for example with the families of 0-valid and 1-valid basis in Schaefer's dichotomy [25].

Constant operations are a particular case of width 1 problems. Since clones are closed under addition of inessential variables, for all $i > 0$, function $f_i : D^i \longrightarrow D$ given by $f_i(x_1, \ldots, x_i) = f_1(x_1) = d$ belongs to any clone containing f_1. In consequence, any set of relations Γ closed under f_1 is closed under the set function $f : \mathcal{P}(D)/\emptyset \longrightarrow D$ given by $f(S) = d$.

4.2 ACI operations

A binary operation f is called ACI if it is associative, commutative and idempotent. The class of ACI operations was identified in [17]. Some known tractable families of problems containing for example horn basis [25] or the constraints allowed in the constraint language CHIP [28] can be explained in terms of ACI functions.

Let f be an ACI operation. It is not difficult to see that, as a consequence of the properties of ACI functions, the set function g, given by

$$g(\{x_1, x_2, \ldots, x_i\}) = f(x_1, f(x_2, \ldots f(x_{i-1}, x_i) \ldots))$$

is well defined. For every i, g_i can be built by a sequence of compositions using f. Thus, any basis closed under f is also closed under g.

4.3 Class CSCI (Constant Semiprojection and Commutative Idempotent)

In this section, we identify a new tractable family of set functions. This new class is obtained from the class ACI replacing the associativity condition by the closure under a particular kind of semiprojection. First we need the following definition:

Let D be a finite set. A *semiprojection* f is a function of rank $k \geq 3$ such that for some index $1 \leq i \leq k$, $f(x_1, \ldots, x_k) = x_i$ whenever $|\{x_1, \ldots, x_k\}| < k$. Furthermore, f is called *constant semiprojection* if $f(x_1, \ldots, x_k) = d$ for some fixed $d \in D$ when the semiprojection condition is not satisfied, i.e., $|\{x_1, \ldots, x_k\}| = k$.

Theorem 4. *Let Γ be a finite set of relations closed under some CI operation f and under some constant semiprojection g of arity 3. Then C_Γ is tractable.*

Proof. The proof has two main parts. First, we see that semiprojections can be extended to any arbitrary arity. Let g_k be a constant semiprojection of arity k where the variable projected is the first one and the constant is d. Then, for every $j > k$, the j-ary function g_j given by:

$$g_j(x_1, \ldots, x_j) = \begin{cases} x_1 \text{ if } |\{x_1, \ldots, x_j\}| < k \\ d \quad \text{otherwise} \end{cases}$$

belongs to any clone containing g_k. That is, we will see that for every $j > k$, g_j can be obtained as a sequence of compositions using only g_k and projections.

We proceed by induction. Assume that g_j with $j \geq k$ belongs to the clone. Operation g_{j+1} can be constructed from g_j in the following way:

$$\begin{aligned} g_{j+1}(x_1, \ldots, x_{j+1}) = g_j(g_j(\ldots g_j(g_j(x_1, x_2, \ldots, x_j), x_2, \ldots \\ \ldots, x_{j-1}, x_{j+1}), \ldots), x_3, \ldots, x_{j+1}) \end{aligned}$$

Now consider the set function h given by:

$$h(S) = \begin{cases} f(S) \text{ if } |S| \leq 2 \\ d \qquad \text{otherwise} \end{cases}$$

For every $k > 3$, function h_k can be constructed by a sequence of compositions using f and g_k in the following way.

$$\begin{aligned} h_k(t_1, t_2 \ldots, t_k) = f(\ldots f(f(g_k(t_1, t_2, \ldots, t_k), g_k(t_2, t_3, \ldots, t_1)), \ldots, \\ \ldots, g_k(t_3, t_4, \ldots, t_2)), \ldots, g_k(t_k, t_1, \ldots, t_{k-1})) \end{aligned}$$

■

In consequence, every clone containing f and g contains h_k for all $k \geq 3$. For $k \leq 2$ we have the easy equivalences: $h_2(x_1, x_2) = f(x_1, x_2)$ and $h_1(x_1) = f(x_1, x_1)$. Then set function h preserves any relations closed under f and g.

The class of constant operations and ACI operations have been previously shown to be tractable by other means. The class CSCI has not been previously identified. It is important to know if the tractability of the class CSCI is simply a consequence of a previously known class of tractable relations, or if a new class of tractable problems has been found. We show that the class CSCI does include problems which can not be accounted for in the framework of closure functions as in [17] by producing a relation which gives rise to tractable constraint problems but does not belong to any of the previously identified classes. At present, tractable classes of constraint problems can be classified in for main families: (**1**) coset generating functions, (**2**) near-unanimity operations, and some set functions (as introduced in these pages). More precisely, the set functions already known are (**3**) constant operations and (**4**) ACI operations.

Thus, it is necessary to prove that there exists some CI function φ and some constant semiprojection ϕ of arity 3 and some relation $\mathcal{R}$, all of them over the same domain D, such that φ and ϕ preserve $\mathcal{R}$, but none of the other known tractable classes preserves $\mathcal{R}$.

Let $D = \{0, 1, 2\}$ be the domain. Let $\varphi : D^2 \to D$ the CI operation with the Cayley table

	0	1	2
0	0	0	2
1	0	1	1
2	2	1	2

Operation φ is a tournament; it is known as *the scissors-paper-stone algebra.* Let $\mathcal{R}$ be the 3-ary relation with tuples

$$\begin{array}{l} \{\ \langle 0,0,1\rangle \\ \quad \langle 0,2,1\rangle \\ \quad \langle 0,2,2\rangle \\ \quad \langle 1,0,2\rangle\ \} \end{array}$$

It is immediate to verify that $\mathcal{R}$ is preserved by φ. Furthermore, since every column contains only two different values, $\mathcal{R}$ is preserved by all the semiprojections (and, in particular, $\mathcal{R}$ is preserved by all the constant semiprojections of arity 3).

The hard work is to prove that relation $\mathcal{R}$ is not closed under any other known tractable operation. We do a case analysis.

Coset Generating Operations Coset generating operations [4] are a direct generalization of affine functions [17] to non-abelian groups. For every coset generating operations $f : D^3 \to D$ there exists some group $(D; \cdot, ^{-1})$ such that $f(x, y, z) = x \cdot y^{-1} \cdot z$. In consequence [4], every n-ary relation R closed under f is a right coset of a subgroup of the group $(D; \cdot, ^{-1})^n$. Thus, $\mathcal{R}$ is not closed under any coset generating operation because, in finite groups, the cardinality of every coset should divide the cardinality of the group.

Near-Unanimity operations An operation $f : D^k \to D$ $(k \geq 3)$, is called a *'near-unanimity (NU) operation'* if for all $x, y \in D$,

$$f(x, y, y, \ldots, y) = \varphi(y, x, y, \ldots, y) = \cdots = \varphi(y, y, \ldots, y, x) = y.$$

In [14], it is proved that closure under a near-unanimity operation is condition sufficient to guarantee the tractability of a constraint satisfaction problem. To prove that none of the near-unanimity functions preserves $\mathcal{R}$ requires some detailed study. We study every arity k separately.

For $k = 3$ the analysis is simple. For every 3-ary near-unanimity operation m (also called majority operation) we have

$$m(\langle 0,0,1\rangle, \langle 0,2,2\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}.$$

Thus, operation m does not preserve $\mathcal{R}$.

Now, assume $k \geq 4$. Let m be an k-ary near-unanimity operation over the domain D preserving $\mathcal{R}$. Since $m(0,2,\ldots,2) = 2$ and $m(0,0,\ldots,0,2) = 0$ there exists some integer $1 \leq n \leq k-2$ such that one of the following conditions is satisfied:

– $m(\overbrace{0,\ldots,0}^{n}, 2\ldots,2) = 1$, or
– $m(\overbrace{0,\ldots,0}^{n}, 2,\ldots,2) = 2$ and $m(\overbrace{0,\ldots,0}^{n+1}, 2,\ldots,2) = 0$

In the first case, we get a contradiction with

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,1,2\rangle \in \mathcal{R}$$

(Impossible for any value for x).

In the second case we have,

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,0,1\rangle, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,0,2\rangle \in \mathcal{R}$$

$$m(\overbrace{\langle 1,0,2\rangle,\ldots,\langle 1,0,2\rangle}^{n}, \langle 0,2,2\rangle, \langle 0,2,2\rangle,\ldots,\langle 0,2,2\rangle) = \langle x,2,2\rangle \in \mathcal{R}$$

Then, we get a contradiction since any value for x can satisfy both conditions.

Constant functions Immediate from the fact that $\mathcal{R}$ does not contain any tuple of the form (d,d,d).

ACI functions Let f be an affine function preserving $\mathcal{R}$. Since f is associative and commutative we have

$$f(\langle 0,2,1\rangle, \langle 1,0,2\rangle) = \langle 0,0,1\rangle \text{ or } \langle 0,2,2\rangle.$$

In the first case we get a contradiction considering $f(\langle 0,2,2\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}$. For the second case, take $f(\langle 0,0,1\rangle, \langle 1,0,2\rangle) = \langle 0,0,2\rangle \notin \mathcal{R}$

References

1. S.A. Cook. The Complexity of Theorem-Proving Procedures. In *3rd Annual ACM Symposium on Theory of Computing STOC'71*, pages 151–158, 1971.
2. M.C. Cooper. An Optimal k-consistency Algorithm. *Artificial Intelligence*, 41:89–95, 1989.
3. M.C. Cooper, D.A. Cohen, and P.G. Jeavons. Characterizing Tractable Constraints. *Artificial Intelligence*, 65:347–361, 1994.
4. V. Dalmau and P. Jeavons. Learnability of Quantified Formulas. In *4th European Conference on Computational Learning Theory Eurocolt'99*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 63–78, Berlin/New York, 1999. Springer-Verlag.
5. R. Dechter. From Local to Global Consistency. *Artificial Intelligence*, 55:87–107, 1992.
6. R. Dechter and J. Pearl. Network-based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.
7. R. Dechter and P. van Beek. Local and Global Relational Consistency. *Theoretical Computer Science*, 173:283–308, 1997.
8. T. Feder and M.Y. Vardi. The Computational Structure of Monotone Monadic SNP and Contraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. Computing*, 28(1):57–104, 1998.
9. E.C. Freuder. Synthesizing Constraint Expressions. *Comm. ACM*, 21:958–966, 1978.
10. E.C. Freuder. A Sufficient Condition for Backtrack-free Search. *Journal of the ACM*, 29:24–32, 1982.
11. E.C. Freuder. A Sufficient Condition for Backtrack-bounded Search. *Journal of the ACM*, 32:755–761, 1985.
12. M. Gyssens, P. Jeavons, and D. Cohen. Decomposing Constraint Satisfaction Problems using Database Techniques. *Artificial Intelligence*, 66(1):57–89, 1994.
13. P. Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theoretical Computer Science*, 200:185–204, 1998.
14. P. Jeavons, D. Cohen, and M.C. Cooper. Constraints, Consistency and Closure. *Artificial Intelligence*, 101:251–265, 1998.
15. P. Jeavons, D. Cohen, and M. Gyssens. A Unifying Framework for Tractable Constraints. In *1st International Conference on Principles and Practice of Constraint Programming, CP'95, Cassis (France), September 1995*, volume 976 of *Lecture Notes in Computer Science*, pages 276–291, Berlin/New York, 1995. Springer-Verlag.
16. P. Jeavons, D. Cohen, and M. Gyssens. A Test for Tractability. In *2nd International Conference on Principles and Practice of Constraint Programming CP'96*, volume 1118 of *Lecture Notes in Computer Science*, pages 267–281, Berlin/New York, August 1996. Springer-Verlag.
17. P. Jeavons, D. Cohen, and M. Gyssens. Closure Properties of Constraints. *Journal of the ACM*, 44(4):527–548, July 1997.
18. P. Jeavons and M. Cooper. Tractable Constraints on Ordered Domains. *Artificial Intelligence*, 79:327–339, 1996.
19. L. Kirousis. Fast Parallel Constraint Satisfaction. *Artificial Intelligence*, 64:147–160, 1993.
20. A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

21. U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.
22. U. Montanari and F. Rossi. Constraint Relaxation may be Perfect. *Artificial Intelligence*, 48:143–170, 1991.
23. N. Pippenger. *Theories of Computability.* Cambridge University Press, 1997.
24. E.L. Post. *The Two-Valued Iterative Systems of Mathematical Logic*, volume 5 of *Annals of Mathematics Studies.* Princeton, N.J, 1941.
25. T.J. Schaefer. The Complexity of Satisfiability Problems. In *10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.
26. A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathématiques Supéreiores.* University of Montreal, 1986.
27. P. van Beek and R. Dechter. On the Minimality and Decomposability of Row-convex Constraint Networks. *Journal of the ACM*, 42:543–561, 1995.
28. P. van Hentenryck, Y. Deville, and C-M. Teng. A Generic Arc-consistency Algorithm and its Specializations. *Artificial Intelligence*, 1992.