Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 172 (2008) 1094–1118
Artificial Intelligence
www.elsevier.com/locate/artint

Domain permutation reduction for constraint satisfaction problems

Martin J. Green *, David A. Cohen

Computer Science Department, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK

Received 24 January 2007; received in revised form 28 November 2007; accepted 7 December 2007

Available online 15 December 2007

Abstract

This paper is concerned with the Constraint Satisfaction Problem (CSP). It is well-known that the general CSP is NP-hard. However, there has been significant success in discovering subproblems which are tractable (polynomial time solvable). One of the most effective ways to obtain a tractable subproblem has been to force all of the constraint relations to lie in some *tractable language*.

In this paper we define a new way of identifying tractable subproblems of the CSP. Let P be an arbitrary CSP instance and Γ be any tractable language. Suppose there exists, for each variable of P, a permutation of the domain such the resultant permuted constraint relations of P all lie in Γ. The domain permuted instance is then an instance of a tractable class and can be solved by the polynomial time algorithm for Γ. Solutions to P can be obtained by inverting the domain permutations.

The question, for a given class of instances and language Γ, whether such a set of domain permutations can be found efficiently is the key to this method's tractability. One of the important contributions made in this paper is the notion of a "lifted constraint instance" which is a powerful tool to study this question.

- We consider the open problem of discovering domain permutations which make instances max-closed. We show that, for bounded arity instances over a Boolean domain this problem is tractable, while for domain size three it is intractable even for binary instances.
- We give a simple proof verifying the tractability of discovering domain permutations which make instances row convex. We refute a published result by giving a simple proof of the intractability of discovering domain permutations which make instances, even with domain size four, connected row convex.
- We demonstrate that triangulated and stable marriage instances are reducible, via domain permutations, to max-closed instances. This provides a simple explanation for arc consistency deciding these instances.
- We verify with a simple direct proof the tractability of identification of renamable Horn instances, and the intractability of finding the largest renamable Horn subset of clauses of an instance of SAT.
- We describe natural tractable classes which properly extend the maximal relational classes arising from tractable constraint languages.

We believe that domain permutation reductions have a significant chance of being useful in practical applications.

Keywords: Complexity; Constraint satisfaction problem; CSP; NP-completeness; Renamable Horn; Stable marriage; Tractability

* Corresponding author.
E-mail address: M.J.Green@cs.rhul.ac.uk (M.J. Green).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2007.12.001

1. Introduction

The constraint satisfaction paradigm involves modelling a real-world problem as a set of variables to which we can assign values from some domain [44]. The values that can be assigned are limited by constraints. A constraint consists of a list of variables (its scope), and a set of tuples which are the allowed assignments to this list of variables (its relation). A solution is the assignment of a value from the domain to every variable so that all of the constraints are satisfied.

Many real-world problems are naturally represented as instances of the general constraint satisfaction problem (CSP), including planning [40], scheduling [48], image processing [44], frequency assignment [21] and natural language understanding [1].

The CSP is NP-hard [43]. This motivates the search for polynomial time solvable (tractable) subproblems.

The structure of a CSP instance is the hypergraph whose hyperedges are the scopes (abstracted to be sets of distinct variables) of the constraints. The set of CSP instances whose structure is acyclic form a tractable subproblem [4]. This structural tractability result has been extended by identifying tractable decompositions of hypergraphs (cycle-cutset [17], hypertree decompositions [30], hinges [22,33], tree-clustering [18], etc.). These decompositions all join the original constraints in sets of size at most k and project each such join to form the constraints of a new, solution equivalent, acyclic CSP instance [15]. The value k is called the width of the decomposition. A decomposition is tractable if, for any k, the set of instances with width at most k can be tractably identified and reduced to the acyclic case.

It is rare that a real-world example has an acyclic structure when modelled as a CSP instance. However the possibility of using these widely applicable decompositions to *reduce* instances to the acyclic case makes structural tractability useful in practice.

A constraint language is a set of allowed (constraint) relations. A relational subproblem of the CSP consists of those instances whose relations lie in a specified language. A language which gives a tractable relational subproblem is called tractable [34,35]. Many tractable languages have been discovered [7,46].

This paper introduces a new study: reductions by domain permutations. A domain permutation reduction of an instance to a given subproblem of the CSP is an independent permutation of the domain of each variable that transforms the instance to lie in that subproblem. Some preliminary results appeared in a conference paper [32].

We will show that determining whether a domain permutation reduction exists for a particular instance to a *relational* subproblem corresponds exactly to the solving of an associated instance of another CSP subproblem. When this "lifted" problem for a tractable language is itself tractable then, under certain natural assumptions, we obtain a new large tractable subproblem.

We consider the well-known max-closed tractable languages [36]. In their original paper Jeavons and Cooper left as an open question whether there exists an efficient algorithm that can decide, for any given instance, if a domain permutation exists that makes the instance max-closed [36]. We answer this question in Section 5.2 for the case where the domain has three or more elements (it is intractable) by analysing the appropriate lifted problem.

However, by more careful analysis of the lifted problem for max-closed languages we are able to properly extend the tractable subproblem of bounded arity Boolean max-closed instances. We can also extend the tractable subproblem of all binary max-closed instances, which has been shown [36] to be maximal as a tractable binary relational subproblem of the CSP.

Our theory also explains why the constraint representation of the Stable Marriage Problem (SMP) [27] is tractably solvable. We have shown that these instances have a natural domain permutation reduction to the max-closed subproblem.

Recently, a new tractable subproblem of binary instances for which arc consistency is a decision procedure, known as triangulated CSP instances [12], has been described. It was left as an open question to discover if some unifying reason exists that explains why arc consistency is a decision procedure for this class. In this paper we provide just such an explanation by showing that these instances also have natural domain permutation reductions to the max-closed subproblem.

A row convex CSP instance [49] can be transformed by a domain permutation to make a particular combinatoric property hold for its constraint relations. A connected row convex CSP instance [20] satisfies a slightly more restrictive combinatoric property (without first applying a domain permutation restriction). This stronger property is preserved by projection and join. It was left as an open question as to whether we can tractably identify the instances for

which domain permutations exist transforming them into connected row convex instances. An incorrect answer to this question has unfortunately appeared in the literature. In this paper we give a simple and correct answer to this question.

We also have a natural explanation for the tractability of renamable Horn theories [3,42], since the lifted problems can easily be shown to be contained in the tractable majority-closed relational subclass [38]. It has been shown using ad-hoc methods that it is NP-hard to find the largest renamable Horn theory which is a subset of a given set of clauses [19]. By analysing the lifted language this result becomes a simple consequence of a well-known result in the theory of the MAX-SAT problem [16].

1.1. Outline of the paper

We give the necessary basic definitions in Section 2 and then continue in Section 3 by discussing tractable constraint languages.

We introduce our new work in Section 4 with the concept of a domain permutation reduction to a tractable relational subproblem and its associated lifted problem. We show the power of the new theory with an analysis of the domain permutation reductions to the max-closed constraint language in Section 5.

We use our new theory in Section 6 to:

- develop some new tractable subproblems of the CSP;
- unify several disparate known tractable subproblems;
- show that the main results from renamable Horn theory may be obtained directly.

We conclude the paper in Section 7 with final remarks and directions for future research.

2. Definitions

In this section we will give a formal definition of a constraint satisfaction problem instance, and the associated notions required for the paper.

2.1. Constraint satisfaction problem instances

Definition 1. An r-ary *relation*, ρ, over D is a subset of D^r.

For any $t \in \rho$ we denote by $t[i]$, $i \in \{1, \ldots, r\}$, the value at the ith coordinate position of t, so that $t = \langle t[1], \ldots, t[r] \rangle$.

Definition 2. A *Constraint Satisfaction Problem instance*, P, is a triple, $\langle V, D, C \rangle$ where:

- V is a set of *variables*;
- D is any set, called the *domain* of the instance;
- C is a set of *constraints*.
  Each constraint $c \in C$ is a pair $\langle \sigma, \rho \rangle$ where σ is a list of variables from V, called the *constraint scope*, and ρ is a $|\sigma|$-ary relation over D, called the *constraint relation*. We call $|\sigma|$ the *arity* of the constraint.

A *solution* to P is a mapping $s : V \to D$ such that, for each $\langle \sigma, \rho \rangle \in C$, $s(\sigma) \in \rho$.

A CSP instance is Boolean if its domain has two elements. It is binary if all its constraints have arity at most two. There is a special representation of binary relations which we will find useful.

Definition 3. The *matrix representation* of a binary relation ρ over domain D is a $(0, 1)$-matrix M_ρ with $|D|$ rows and columns. The entry $M_\rho\langle i, j \rangle$ is equal to 1 exactly when $\langle i, j \rangle \in \rho$.

2.2. Complexity of constraint satisfaction

For completeness we describe the representation of a CSP instance. Firstly the domain size and size of the variable set are given as integers. Then follows a list of constraints, each first giving the scope as a list of variables then the relation. Each constraint relation is represented extensionally as a list of tuples of domain values. In this representation the size of the instance $\langle V, D, C\rangle$ is:

$$\log|V| + \log|D| + \sum_{\langle\sigma,\rho\rangle\in C} |\sigma|\big(\log|V| + |\rho|\log|D|\big).$$

For any set $\mathcal{C}$ of CSP instances the *decision problem* for $\mathcal{C}$ is to determine, for any instance in $\mathcal{C}$, whether it has a solution.

It is clear that graph 3-colouring, or indeed 3-SAT, can be reduced to the decision problem for (an appropriate) set of CSP instances and so the decision problem for the class of *all* CSP instances is NP-complete [43].

If we can solve a subproblem of the CSP then we can at least tell whether any real-world problem which is modelled as an instance of this subproblem is feasible. However, our task is usually to find such a solution if one exists.

For any set $\mathcal{C}$ of CSP instances the *search problem* is to determine, for any instance in $\mathcal{C}$, whether it has a solution, and if so to find such a solution.

Definition 4. We call a problem $\mathcal{P}$ *tractable* if it has an algorithm which runs in time polynomial in the size of the input instance.

We call $\mathcal{P}$ *intractable* if it is NP-hard [24].

It is the complexity of the search problem with which we are concerned in this paper.

We extend the notion of a tractable set of constraint satisfaction problem instances by requiring that membership should also be polynomially determined. Of course, this is always a sensible restriction if we are going to use tractability results in any sense as an *algorithm selector*. The extra requirement of identifiability is usually adopted by researchers on general satisfiability [19], and researchers on *structural decompositions* [28], but not unfortunately by researchers on *relational tractability*.

If we do not require polynomial time identification and instead consider a set of CSP instances to be tractable if they have a tractable search problem then this leads to the absurd notion that the set of all unsolvable CSP instances is tractable! The algorithm for solving instances from this set of CSP instances is easy (just say no), but determining whether a CSP instance is a member of this set is NP-hard. The usefulness of this notion of tractability is highly doubtful. To overcome this limitation, we define, for any set $\mathcal{C}$ of CSP instances the *identification problem* for $\mathcal{C}$ which is to determine, for any CSP instance, whether or not it lies in $\mathcal{C}$.

We will call a class $\mathcal{C}$ of CSP instances *tractable* if *both* its search problem and its identification problem are tractable, otherwise we say that $\mathcal{C}$ is *intractable*.

This notion of tractability is of practical use: When we are given a CSP instance, we have first to determine which (tractable) subclass (if any) it belongs to, and then we can apply the appropriate solution algorithm.

Even under this stronger notion of tractability there are many tractable sets of CSP instances.

Definition 5. A *hypergraph* H is a pair $\langle V, E\rangle$ where V is a set of vertices and E is a collection of subsets of V, called the *hyperedges* of H.

The *structure* of a CSP instance, P, is the hypergraph $\sigma(P)$ whose vertices are the variables of P and whose hyperedges are the scopes of the constraints of P (abstracted to be sets of distinct variables).

A class of CSP instances is called *structural* if it is precisely those instances whose structure is limited in some particular way.

A fundamental tractable structural subclass of the general constraint satisfaction problem is those instances whose structure is acyclic [4]. However, this subclass has been extended by considering instances whose structure is "nearly acyclic" in the sense that there is a tractable *reduction* to an acyclic instance [29]. These reductions give rise to sets of CSP instances whose cost of reduction to the acyclic subclass is bounded by a polynomial. Such subclasses have been well studied and have made the use of tractable structural subclasses applicable to a wide variety of real-world examples [41].

Definition 6. A *constraint language* over a domain D is a set of relations over D.

The *language* of a CSP instance, P, is the constraint language defined by the set of relations of the constraints of P. The language of a set of CSP instances is the union of the languages of the instances. For any constraint language Γ we will refer to the set of CSP instances with language contained in Γ as CSP(Γ).

A class of CSP instances is called *relational* if it is precisely CSP(Γ) for some language Γ.

A constraint language is called tractable if the set of instances defined over this language has a tractable decision problem. There are many known tractable constraint languages [7,13]. We will describe some of these in Section 3.

It has been shown that if the decision problem can be answered polynomially for the set of instances defined over a given constraint language then so can the search problem [11]. It follows that a relational subclass which has a tractable identification problem and whose language is tractable is also tractable in the sense in which we use the term in this paper.

Previous work has not considered the possibility of reductions to relational tractability. In this paper we introduce the notion of domain permutation reduction of an instance to a given subproblem of the CSP, which is an independent permutation of the domain of each variable that transforms the instance to lie in that subproblem.

This natural extension to the theory of tractability yields large new tractable subclasses, unifies apparently disparate known tractable classes and simplifies proofs of (in)tractability.

Example 7. In this example we define a new class of Boolean CSP instances whose constraints are generalisations of Horn-clauses.

A propositional *literal* is either a propositional variable (positive), for example p, or a negated propositional variable (negative), for example $\overline{q}$. A *clause* is a disjunction of literals, for example, $p \vee \overline{q}$.

Let $V^+ = \{p_1, p_2, \ldots\}$ and $V^- = \{q_1, q_2, \ldots\}$ and let $V = V^+ \cup V^-$. For any clause c over the variables in V, positive disjuncts from V^+ and negative disjuncts from V^- are said to have appropriate negation. We say that c is *split-Horn* if there is at most one disjunct with inappropriate negation.

Order the set of variables V as $p_1, q_1, p_2, q_2, p_3, q_3, \ldots$. We represent a clause as the constraint whose scope is the ordered list of variables occurring in the clause and whose relation allows the truth assignments allowed by the clause. In this way we can represent conjuncts of split-Horn clauses as CSP instances over the Boolean domain $\{F, T\}$.

We call these Boolean CSP instances *split-Horn*.

We can easily construct split-Horn instances with any given structure. So the class of all split-Horn instances is not structurally tractable.

Example 8. Let H be any hypergraph. Without loss of generality identify the vertex set of H with a subset of V^+. For each edge e of H define the edge clause $c(e) = \bigvee_{p \in e} p$. The CSP instance representing the conjunct of all of the edge clauses has structure H and is split-Horn.

There is a well-known dichotomy result for Boolean constraint languages [47]. The language of the split-Horn instances is NP-hard. However, we easily show the tractability of the split-Horn subclasses in Section 6.5 as part of a more general tractable class using the theory of reductions to tractable constraint languages.

3. Tractable constraint languages

In this section we briefly describe the well-known algebraic approach to determining the tractability of a (finite) constraint language [13,37].

Definition 9. An *n-ary operator* ϕ over a set D is a function from D^n to D. We apply ϕ to n tuples $t_1, \ldots, t_n$ from the r-ary relation ρ over D by applying it pointwise as follows:

$$\phi(t_1, \ldots, t_n) = \big\langle \phi\big(t_1[1], \ldots, t_n[1]\big), \ldots, \phi\big(t_1[r], \ldots, t_n[r]\big)\big\rangle.$$

We call ϕ an *n-ary polymorphism* [37] of ρ if it preserves membership of ρ in the following way:

$$\forall t_1, \ldots, t_n \in \rho, \quad \phi(t_1, \ldots, t_n) \in \rho.$$

When ϕ is a polymorphism of ρ we say that ρ is *ϕ-closed* or *closed under ϕ*. We say that ϕ is a polymorphism of a constraint language Γ if ϕ is a polymorphism of every relation in Γ. If this is the case we also say that Γ is ϕ-closed or closed under ϕ.

Closure under certain types of polymorphism guarantees tractability. That is, there is an efficient algorithm for solving the class of CSP instances whose constraint relations are all closed under a particular polymorphism. In the next section we shall describe some of these special polymorphisms which give tractability and we will then use these examples throughout the remainder of the paper. We will also describe a tool, Polyanna [25,26], which can be used to find such useful polymorphisms for a given finite constraint language, or even to prove that none exist.

3.1. Useful polymorphisms

Definition 10. Given the ordered domain $\{1, \ldots, k\}$ we define the binary *max* operator to be the operator that returns the larger of its two arguments.

The set of all max-closed relations forms a tractable constraint language [36]. In particular, a polynomial solution technique is to establish (generalised) arc consistency [5,43], and then to choose the largest remaining domain value for each variable.

There is also the analogous tractable language of all min-closed relations that are closed under the binary operator min which returns the smaller of its two arguments.

Definition 11. A ternary operator ϕ over domain D is called a *majority* operator if, for every $d, e \in D$, we have that:

$$\phi(d, d, e) = \phi(d, e, d) = \phi(e, d, d) = d.$$

The set of all relations closed under any given majority operator is tractable [38]. A polynomial solution technique here is to transform the instance into an equivalent binary instance for which path consistency is a decision procedure.

3.2. Polyanna

A very difficult problem is that of determining whether a given constraint language is tractable or intractable. We know that the existence of a useful polymorphism can guarantee tractability [36,37]. Even so, it is usually very difficult to determine tractability for quite small constraint languages.

However, Gault has written a program, Polyanna [25,26], which determines the polymorphisms of an input constraint language. It uses many optimisation techniques to reduce the cost of searching for polymorphisms. In addition to listing the polymorphisms, Polyanna can classify a constraint language into one of several tractability classes.

Using the fact that a constraint language with domain D is intractable if it has no non-trivial polymorphisms up to arity $\max(3, |D|)$ [37] Polyanna is also able to demonstrate intractability in many cases.

4. Domain permutation reductions

Suppose that we have a tractable constraint language Γ and that P is any CSP instance with the same domain. If we can find permutations of the domain (independently) for each variable, that transform P into an instance of CSP(Γ), then we can solve the instance P using the algorithm for Γ. We first permute the domains, then apply the algorithm for Γ, then permute the domains back again for any discovered solution.

This reduction technique for (tractable) constraint languages is the focus of this paper.

Definition 12. A *permutation* of a set D is a bijection from D to D.

Let $\vec{\pi} = \langle \pi_1, \ldots, \pi_r \rangle$ be an r-tuple of permutations of D. We apply $\vec{\pi}$ to r-tuples of values from D in the natural way by applying π_i, $i = 1, \ldots, r$, to the ith component. We apply $\vec{\pi}$ to r-ary relations over D by applying it in turn to each of the allowed tuples.

Let $P = \langle V, D, C\rangle$ be a CSP instance, and G be a set of permutations of D. A *domain permutation* for P over G is a mapping Π assigning to each variable $v \in V$ a permutation from G.

For any scope $\sigma = \langle v_1, \ldots, v_r\rangle$ we define $\Pi(\sigma) = \langle \Pi(v_1), \ldots, \Pi(v_r)\rangle$.

For any constraint $c = \langle \sigma, \rho\rangle$ we define $\Pi(c) = \langle \sigma, \Pi(\sigma)(\rho)\rangle$.

For a CSP instance $P = \langle V, D, C\rangle$ we define $\Pi(P) = \langle V, D, \{\Pi(c) \mid c \in C\}\rangle$.

Discovery of appropriate domain permutations will be the identification procedure for these new tractable classes. It can be the case that this identification problem is intractable when we consider every possible permutation of the domain. For this reason we may need to reduce the set G of permutations of the domain. An example (Example 41) is given in Section 6.1.

Example 13. Consider again the split-Horn class defined in Example 7.

Both of the clauses $c_1 = p_1 \vee \overline{p_2} \vee \overline{q_1}$ and $c_2 = p_2 \vee q_1 \vee \overline{q_2}$. have exactly one disjunct with inappropriate negation and so the instance P_{sh} representing their conjunction is split-Horn.

There is exactly one non-trivial permutation τ of the domain $\{F, T\}$ of P_{sh}: it interchanges T and F. The trivial permutation ι fixes both T and F.

Define the domain permutation Π for P_{sh} as:

$$\Pi(v) = \begin{cases} \iota & \text{if } v = p_2, \\ \tau & \text{otherwise.} \end{cases}$$

The scope of c_1 is $\langle p_1, p_2, q_1\rangle$ and $\Pi(\langle p_1, p_2, q_1\rangle) = \langle \tau, \iota, \tau\rangle$. Hence $\Pi(c_1)$ is the constraint representing the clause $\overline{p_1} \vee \overline{p_2} \vee q_1$. Similarly $\Pi(c_2)$ is the constraint representing the clause $p_2 \vee \overline{q_1} \vee q_2$.

Whilst the clause represented by $\Pi(c_2)$ has appropriate negation, that represented by $\Pi(c_1)$ does not and so $\Pi(P_{sh})$ is not split-Horn.

Definition 14. Let Γ be a constraint language, $P = \langle V, D, C\rangle$ a CSP instance and G a set of permutations of D. If there exists a domain permutation Π for P over G such that $\Pi(P) \in \mathrm{CSP}(\Gamma)$ then we say that P is *G-reducible* to Γ.

For a given Γ and G the problem of determining whether an instance is G-reducible to Γ is called the *G-reduction problem* into Γ.

Example 15. Let Γ be any constraint language.

Suppose that G is a set of permutations of D that contains the identity permutation ι.

Let $P = \langle V, D, C\rangle$ be a CSP instance in $\mathrm{CSP}(\Gamma)$. Consider the constant domain permutation Π where $\Pi(v) = \iota$ for every $v \in V$.

It is immediate from the definition that $\Pi(P) = P$.

Hence if G contains the identify permutation then every instance of $\mathrm{CSP}(\Gamma)$ is G-reducible to Γ.

For tractable languages Γ we are looking for large sets of instances $\mathcal{C}$ for which the G-reduction problem into Γ is tractable. In this case the set of G-reducible instances in $\mathcal{C}$ is a tractably solvable set of CSP instances. The results of this paper are all applications of the following theorem whose proof is straightforward.

Theorem 16. *Let Γ be a tractable constraint language over a domain D, G be a set of permutations of D, and $\mathcal{C}$ be a set of instances with a tractable G-reduction problem into Γ. The set of instances of $\mathcal{C}$ which are G-reducible to Γ has a tractable search problem.*

We obtain the following corollary directly from the definition of tractability.

Corollary 17. *Let Γ be a tractable constraint language over a domain D, G be a set of permutations of D, and $\mathcal{C}$ be a tractably identifiable set of instances with a tractable G-reduction problem into Γ. The set of instances of $\mathcal{C}$ which are G-reducible to Γ is tractable.*

4.1. Using lifted relations to find domain permutations

A natural application of Theorem 16 is to the case where $\mathcal{C}$ is CSP(Δ) for some language Δ.

Definition 18. Let Δ be any constraint language over domain D.

The subproblem of the *G-reduction problem* into Γ whose instances are precisely the CSP instances of CSP(Δ) is called the G-reduction problem from Δ into Γ.

Example 19. Let Δ be the language of all Boolean ternary relations. Let Γ be the language of all max-closed Boolean ternary relations.

Again let τ swap the Boolean domain elements and let ι leave them fixed. Let $G_1 = \{\iota\}$ and $G_2 = \{\tau, \iota\}$.

The G_1-reduction problem from Δ into Γ just asks which instances of CSP(Δ) happen also to be instances of CSP(Γ). This is just the identification problem for Boolean ternary max-closed instances which is tractable.

On the other hand the G_2-reduction problem for Δ into Γ asks for a given Boolean ternary instance, whether there are independent permutations of the domain at each variable, which make the instance max-closed. The tractability of this question is not obvious. Proposition 36 will show this also to be a tractable problem.

For any r-ary relation ρ over D there are some r-tuples of permutations from G which make ρ into a relation of Γ. The set of such r-tuples over G is simply an r-ary relation over G.

Definition 20. Let Γ be a constraint language, ρ be an r-ary relation over D, and G be a set of permutations of D. We define the *G-lifted relation* of ρ into Γ, ρ^G_Γ, to be the following r-ary relation over G:

$$\rho^G_\Gamma = \left\{ \langle \pi_1, \ldots, \pi_r \rangle \in G^r \;\middle|\; \langle \pi_1, \ldots, \pi_r \rangle(\rho) \in \Gamma \right\}.$$

Given a CSP instance P we can use lifted relations to construct a CSP instance with domain G whose solutions are precisely the domain permutations that transform P into an instance of CSP(Γ).

Definition 21. Let $P = \langle V, D, C \rangle$ be a CSP instance, G be a set of permutations of D and Γ be a constraint language.

We define the *G-lifted instance* for P into Γ, P^G_Γ, to be the CSP instance $\langle V, G, C' \rangle$ where $C' = \{\langle \sigma, \rho^G_\Gamma \rangle \mid \langle \sigma, \rho \rangle \in C\}$.

Example 23, concerning row convex constraints, illustrates the power of this technique.

Definition 22. A binary relation ρ over an ordered domain is said to be *row convex* [49] if in every row and column of its matrix representation the entries that are one are consecutive.

Example 23. Whilst row convex constraints do not form a tractable language, in the sense of this paper, Dechter and van Beek [49] proved that any path consistent CSP instance whose relations are row convex is globally consistent.

They also described the *row convex identification problem* for binary CSP instances, which is precisely the reduction problem into the row convex constraint language.

We will solve the row convex identification problem for a particular CSP instance $P = \langle V, D, C \rangle$ by solving the lifted instance.

Let the domain D of P be $\{1, 2, 3\}$ and the variables V of P be $\{x, y, z\}$. We will use only one constraint relation, ρ, which for convenience we describe in matrix form:

$$\begin{array}{c} \\ 3 \\ 2 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}.$$

Here we have explicitly indexed the rows and columns of the array with the elements of the domain.

There are three constraints in P, one for each pair of variables, each constrained by the relation ρ. That is, $C = \{\langle\langle x, y \rangle, \rho\rangle\langle\langle y, z \rangle, \rho\rangle\langle\langle z, x \rangle, \rho\rangle\}$.

To make ρ row convex we will apply permutations (possibly trivial) to the domain of each variable. There are six possible permutations of the domain $\{1, 2, 3\}$ which we will name as follows.

- ι fixes all three domain elements.
- For each $\{i, j\}$ in D, $\pi_{\{i,j\}}$ swaps i and j and fixes the third domain element.
- $\pi_{\rightarrow}$ moves 1 to 2, 2 to 3 and 3 to 1.
- $\pi_{\leftarrow}$ moves 3 to 2, 2 to 1 and 1 to 3.

Let G be the set of these six permutations and Γ be the set of row convex relations with domain $\{1, 2, 3\}$. The lifted relation ρ_Γ^G consists of pairs of elements of G. The first component of a tuple in ρ_Γ^G permutes the rows of the matrix representation and the second permutes the columns.

Let $\langle \omega_1, \omega_2 \rangle \in \rho_\Gamma^G$. The permutation ω_1 must leave 1 and 3 adjacent. Hence ω_1 must be in the set $\Omega_1 = \{\pi_{\{1,2\}}, \pi_{\rightarrow}, \pi_{\{2,3\}}, \pi_{\leftarrow}\}$. Similarly, ω_2 must leave 1 and 3 adjacent, and 1 and 2 adjacent. Hence ω_2 must be in the set $\Omega_2 = \{\pi_{\{1,2\}}, \pi_{\rightarrow}\}$.

Since permuting the rows (columns) leaves the row (column) contents unchanged the permutation ω_1 which collects together the ones in the columns can be chosen independently of the permutation ω_2 which collects together the ones in the rows. So the lifted relation ρ_Γ^G is the product of unary relations and we have that $\rho_\Gamma^G = \Omega_1 \times \Omega_2$.

The lifted instance $P_\Gamma^G = \langle V, G, C' \rangle$ has the same three variables as P. The domain is the set of permutations, G.

There is a constraint in C' for each constraint in C. In this case $C' = \{\langle \langle x, y \rangle, \rho_\Gamma^G \rangle \langle \langle y, z \rangle, \rho_\Gamma^G \rangle \langle \langle z, x \rangle, \rho_\Gamma^G \rangle\}$.

The solutions to P_Γ^G are precisely the assignments $s : V \mapsto G$ where $\forall v \in V$, $s(v) \in \{\pi_{\{1,2\}}, \pi_{\rightarrow}\}$. By construction these are precisely the set of domain permutations that solve the reduction problem for P: simultaneously permuting the domain of each variable to make each constraint relation row convex.

We can generalise these notions to specify a new relational problem equivalent to the G reduction problem from a language Δ to the language Γ.

Definition 24. For any constraint language Δ we define the *G-lifted language* of Δ into Γ to be $\Delta_\Gamma^G = \{\rho_\Gamma^G \mid \rho \in \Delta\}$.

We call the search problem for the set of instances $\{P_\Gamma^G \mid P \in \mathrm{CSP}(\Delta)\}$ the *G-lifted problem* from Δ into Γ.

Proposition 25. *The G-lifted problem from Δ into Γ is tractable if and only if Δ_Γ^G is a tractable constraint language.*

Proof. The result follows immediately from the fact that the instances of $\mathrm{CSP}(\Delta_\Gamma^G)$ are precisely the lifted instances of $\mathrm{CSP}(\Delta)$. □

Proposition 26. *The G-reduction problem into Γ for P has a solution if and only if P_Γ^G has a solution.*

Proof. For any CSP instance, P, solutions to the instance P_Γ^G are defined to be the domain permutations from G that reduce P to an instance of $\mathrm{CSP}(\Gamma)$. □

We do not get immediately that the G-reduction problem is tractable if and only if the lifted language is tractable. We have to consider the cost of moving between representations. In this paper we will use two results connecting lifted languages and reduction problems. The proof of these propositions is straightforward and omitted.

Proposition 27. *If there exists a polynomial time algorithm that, given any $P \in \mathrm{CSP}(\Delta)$, generates P_Γ^G and Δ_Γ^G is tractable then the G-reduction problem from Δ into Γ is tractable.*

Proposition 28. *If any finite subset of Δ_Γ^G is intractable, so is the G-reduction problem from Δ into Γ.*

It follows immediately that:

Corollary 29. *If Δ is finite then the G-reduction problem from Δ into Γ is tractable if and only if Δ_Γ^G is tractable.*

Proof. If Δ is finite then so is Δ_Γ^G. Since Δ is finite, for any $P \in \mathrm{CSP}(\Delta)$ we can generate P_Γ^G using a lookup table. □

Proposition 27 will be used to establish the tractability of novel subclasses of the general constraint satisfaction problem. Proposition 28 will only be useful in establishing the intractability of certain problems. We will use Corollary 29 repeatedly in the remainder of this paper to prove tractability results.

We finish this section by showing how this new theory can give a simple explanation for an important theorem whose existing proof is quite involved.

Example 30. Recall the definition of the row convex identification problem from Example 23.

Dechter and van Beek [49] proved that the row convex identification problem is tractable using a result from graph theory.

In Example 23 we established the fact that permuting the rows (columns) leaves the row (column) contents unchanged. It follows that any lifted relation will be the product of unary relations so the lifted language is tractable.

The tractability of the reduction problem follows immediately by Proposition 27.

4.2. The intractability of reduction

In general, the lifted problem and reduction problem are intractable, even when lifting into tractable languages.

More specifically, there exist constraint languages with only one relation whose lifted problem and reduction problem into a tractable language are both intractable.

Example 31. Let Γ be the 0-*valid* language over $D = \{0, 1, 2\}$. That is, Γ contains all relations over D which allow the all zeros tuple.We can solve any instance by assigning zero to every variable, so Γ is tractable.

The graph 3-colourability problem can be naturally encoded using CSP instances over domain D with only binary three valued inequality constraints. It follows that the language, Δ, containing only the binary three-valued inequality relation is intractable.

Now consider any CSP instance $P = \langle V, D, C\rangle$ in $\mathrm{CSP}(\Delta)$.

Suppose that s is a solution to P. Let π_v be the permutation of D which swaps $s(v)$ and 0 and fixes all other domain values. The domain permutation Π where $\forall v \in V$, $\Pi(v) = \pi_v$ is a solution to the lifted instance P_Γ.

Conversely, let $\hat{s}$ be a solution to P_Γ. Define the function $s' : V \mapsto D$ by $s'(v) = (\hat{s}(v))^{-1}(0)$. By construction, s' is a solution to P.

Since Δ is finite it follows by Corollary 29 that the reduction problem is intractable.

5. Lifted relations into max-closure

It may not be surprising that the lifted problem of Example 31 is intractable, for it is exactly as hard as solving the instances for which we are trying to find reductions.

Jeavons and Cooper [36] pose a question of the complexity of identifying instances that become max-closed after applying an (independent) re-ordering of the domain at each variable. In this section we will show that this problem is tractable for the class of bounded arity Boolean CSP instances but is intractable for larger domains, even in the case of binary CSP instances. The question remains open for Boolean CSP instances of unbounded arity.

Schaefer [47] classified all maximal Boolean tractable *relational* classes of CSP instances. Our new result for bounded arity Boolean instances properly extends a maximal Boolean relational class and so has interest beyond (partially) solving an open question.

Example 32. A disjunction of propositional literals (clause) with at most one negative literal is called an anti-Horn clause. Any Boolean relation corresponding to an anti-Horn clause is called an anti-Horn relation. Jeavons et al. [37] showed that the anti-Horn relations are all max-closed.

For any $k > 2$, let Γ_k be the class of max-closed relations of arity at most k. Jeavons et al. [37] showed that the class $\mathrm{CSP}(\Gamma_k)$ is a maximal tractable relational class. That is, for any arity k relation ρ, where $\rho \notin \Gamma_k$, $\mathrm{CSP}(\Gamma_k \cup \{\rho\})$ is intractable.

Consider again the split-Horn class $\mathcal{C}$ of Boolean instances defined in Example 7. Certainly, for every k, $\mathcal{C}$ contains an instance which is not in $\mathrm{CSP}(\Gamma_k)$. Despite this we will show that each class $\mathcal{C} \cup \mathrm{CSP}(\Gamma_k)$, $k > 2$ is tractable. This extension is compatible with Schaefer's original classification [47] since he identified precisely the maximal (by inclusion) tractable *relational* classes.

Given an instance $P = \langle V, D, C \rangle$ first check (in polynomial time) whether it is in $\mathrm{CSP}(\Gamma_k)$. Otherwise define the domain permutation Π for P, where $\Pi(v)$ is the permutation swapping T and F if $v \in V^-$ and $\Pi(v)$ fixes T and F otherwise. Since each clause defining P has at most one disjunct with inappropriate negation it follows that each constraint relation in $\Pi(P)$ is anti-Horn and so max-closed.

We have established that each class of instances, $\mathcal{C} \cup \mathrm{CSP}(\Gamma_k)$, $k > 2$, is a proper tractable extension of a maximal tractable relational class.

These new tractable CSP classes are obtained by domain permutation reduction to the max-closed language. It is therefore of interest to determine whether the class of all arity k instances which are reducible by domain permutations to the max-closed language is tractable.

5.1. Lifting Boolean CSP instances into max-closure

Let Γ be the set of all max-closed relations over the ordered Boolean domain $\{F, T\}$, where $F < T$. There are only two permutations of a Boolean domain. Again we denote by ι the identity permutation, and by τ the permutation that swaps the two domain values. Let G be the set $\{\iota, \tau\}$.

We find a representation for a tuple of permutations from G in terms of a tuple of binary operators. By considering the structure of the set of tuples corresponding to lifting into max-closure for Boolean CSP instances we will show that this reduction problem is tractable when a bound is imposed on the arity of the CSP instances.

Definition 33. Suppose that $l \in G^r$ so that l is an r-tuple of permutations of the Boolean domain $\{F, T\}$. We define $m_l \in \{\min, \max\}^r$ to be the r-tuple of binary operators such that, for $i = 1, \ldots, r$, $m_l[i] = \max$ if and only if $l[i] = \iota$.

Let ρ be an r-ary relation and $l \in G^r$. We define the application of m_l to $t_1, t_2 \in \rho$, denoted $m_l(t_1, t_2)$, to be the tuple $\langle m_l[1](t_1[1], t_2[1]), \ldots, m_l[r](t_1[r], t_2[r]) \rangle$. We say that m_l *fixes* ρ if for every $t_1, t_2 \in \rho$ we have that $m_l(t_1, t_2) \in \rho$.

We now require the following technical lemma.

Lemma 34. *Let ρ be a Boolean relation of arity r. For any $l \in G^r$, m_l fixes ρ if and only if $l(\rho)$ is max-closed (that is, $l \in \rho^G_\Gamma$).*

Proof. Let $l \in G^r$ and consider any $t_1, t_2 \in \rho$. We now consider the application of l and m_l to the pair of tuples t_1 and t_2.

We first show that, for every $t_1, t_2 \in \rho$, $\max(l(t_1), l(t_2)) = l(m_l(t_1, t_2))$.

Let $i \in \{1, \ldots, r\}$.

If $t_1[i] = t_2[i]$ then $\max(l[i](t_1[i]), l[i](t_2[i])) = l[i](t_1[i]) = l[i](m_l[i](t_1[i], t_2[i]))$.

Alternatively, $\max(l[i](t_1[i]), l[i](t_2[i])) = T = l[i](m_l[i](t_1[i], t_2[i]))$ occurs when $t_1[i] \neq t_2[i]$.

We know that $l \in \rho^G_\Gamma$ if and only if $\forall t_1, t_2 \in \rho$, $\max(l(t_1), l(t_2)) \in l(\rho)$. By the above argument this is equivalent to saying that $\forall t_1, t_2 \in \rho$, $l(m_l(t_1, t_2)) \in l(\rho)$. This is true if and only if $\forall t_1, t_2 \in \rho$, $m_l(t_1, t_2) \in \rho$ which exactly means that m_l fixes ρ, and we are done. □

Example 35. Let $t_1 = \langle F, F, T, T \rangle$, $t_2 = \langle T, T, F, F \rangle$ and $\rho = \{t_1, t_2\}$.

Since t_1 and t_2 differ in each coordinate, for any four-tuple Π of permutations of $\{F, T\}$ we have that $\max(\Pi(t_1), \Pi(t_2)) = \langle T, T, T, T \rangle$. It follows that the lifted relation for ρ into the max-closed constraint language is $\{\langle \iota, \iota, \tau, \tau \rangle, \langle \tau, \tau, \iota, \iota \rangle\}$.

Let $l_1 = \langle \iota, \iota, \tau, \tau \rangle$ which is in this lifted relation. By definition $m_{l_1} = \langle \max, \max, \min, \min \rangle$. We obtain the tuple $m_{l_1}(t_1, t_2) = t_2$.

Now let $l_2 = \langle \iota, \iota, \iota, \tau \rangle$ which is not in the lifted relation. By definition $m_{l_2} = \langle \max, \max, \max, \min \rangle$. In this case $m_{l_2}(t_1, t_2) = \langle T, T, T, F \rangle \notin \rho$ and so m_{l_2} does not fix ρ.

Using Lemma 34 we will show that the lifted problem from the set of Boolean CSP instances into the Boolean max-closed constraint language is tractably solvable. We do so by demonstrating that the lifted relations are all majority-closed, and hence form a tractable constraint language.

Proposition 36. *The set of all lifted relations, from the set of all Boolean relations into the Boolean max-closed constraint language, is majority-closed.*

Proof. Consider a Boolean relation ρ and its lifted relation ρ_Γ^G into the max-closed constraint language. We will prove directly that ρ_Γ^G is majority-closed.

Consider any $l_1, l_2, l_3 \in \rho_\Gamma^G$. Recall from Section 3.1 that there is a unique majority operator over the Boolean domain. Let l_{maj} be the result of applying this majority operator to l_1, l_2 and l_3. By definition of l_{maj} we can see that $m_{l_{\text{maj}}}$ is also the result of applying the unique majority operator over $\{\min, \max\}$ to the three tuples m_{l_1}, m_{l_2} and m_{l_3}.

Consider any two tuples $t_1, t_2 \in \rho$ and let $t_3 = m_{l_3}(m_{l_1}(t_1, t_2), m_{l_2}(t_1, t_2))$. By Lemma 34 m_{l_1}, m_{l_2} and m_{l_3} all fix ρ and so $t_3 \in \rho$. We will show that $t_3 = m_{l_{\text{maj}}}(t_1, t_2)$.

If $m_{l_1}[i] = m_{l_2}[i]$ then

$$\begin{aligned} t_3[i] &= m_{l_3}[i]\big(m_{l_1}[i]\big(t_1[i], t_2[i]\big), m_{l_2}[i]\big(t_1[i], t_2[i]\big)\big) \\ &= m_{l_1}[i]\big(t_1[i], t_2[i]\big) \\ &= m_{l_{\text{maj}}}[i]\big(t_1[i], t_2[i]\big). \end{aligned}$$

Alternatively, $m_{l_1}[i] \neq m_{l_2}[i]$ and then

$$\begin{aligned} t_3[i] &= m_{l_3}[i]\big(\min\big(t_1[i], t_2[i]\big), \max\big(t_1[i], t_2[i]\big)\big) \\ &= m_{l_3}[i]\big(t_1[i], t_2[i]\big) \\ &= m_{l_{\text{maj}}}[i]\big(t_1[i], t_2[i]\big). \end{aligned}$$

Hence for every $t_1, t_2 \in \rho$ we have that $m_{l_{\text{maj}}}(t_1, t_2) = t_3 \in \rho$ and so $m_{l_{\text{maj}}}$ fixes ρ. By Lemma 34 this implies that $l_{\text{maj}} \in \rho_\Gamma^G$ and we are done. □

Proposition 37. *The set of bounded arity Boolean CSP instances which are reducible to the max-closed constraint language forms a tractable subclass.*

Proof. The set of Boolean relations with bounded arity is a finite constraint language and so, by Corollary 29, the reduction problem is tractable if and only if the lifted language is tractable. □

The question of whether the max-closed reduction problem for general Boolean instances is tractable is still open.

In the next section we show that, for larger domains, the identification problem is intractable.

5.2. Binary three valued CSP instances

In this section we consider tractable classes of binary instances over a three valued domain. The classes we consider are those reducible to max-closed instances. The results of this section were obtained by using Polyanna [25,26] to discover the complexity of subsets of lifted relations.

Naturally the subset of 230 binary three valued relations which are max-closed is tractable. The reduction problem is trivial. There are five non-trivial permutations of the three valued domain. By symmetry, for each of these five permutations there are again 230 relations which are max-closed after applying this permutation to both domains. These are the six languages that are max-closed for appropriate choices of domain ordering. These languages all have trivial reduction problems whose solution is to apply the same domain permutation to every variable. All are maximal tractable languages.

There are also binary three valued languages for which the lifted language is tractable but non-trivial. The lifted instances can be solved in polynomial time but not by simply choosing the same permutation for each variable of the

instance. Discovering an appropriate domain permutation for an instance may take quadratic time or cubic time depending which lifted language is being considered. The subsets of the binary three valued relations which correspond to these languages are not tractable. However those instances which have solvable lifted instances form a tractable class of binary three valued instances.

There is one anomalous binary three valued language for which the lifted language is tractable which gives us an empty set of reducible instances. That is the language consisting of the six domain permutations of the three valued not equals relation. These are precisely the relations for which no domain permutation makes them max-closed. As such each lifted instance has no solution. Whilst the reduction problem is tractable, since it never has a solution, we find no reducible instances in this class.

However the set of all lifted relations is not tractable.

Proposition 38. *The reduction problem from the binary three valued constraint language into the binary three valued max-closed constraint language is intractable.*

Proof. Consider the two binary relations with domain $\{1, 2, 3\}$: $\rho_1 = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle\}$ and $\rho_2 = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 1\rangle\}$.

The lifted language for $\{\rho_1, \rho_2\}$ is intractable. By Corollary 29, the lifted problem and the reduction problem are both intractable. □

Using Polyanna, we have demonstrated the tractability of some (large) subsets of the lifted relations, and have shown that the entire set of lifted relations is not tractable.

The intractability result has in fact been verified by hand (see Section 5.3.3 of Green [31]) to reduce reliance on computer generated results.

We have largely answered the question of Jeavons and Cooper [36] concerning the complexity of identifying instances that become max-closed after applying an (independent) re-ordering of the domain at each variable. For the language of arity at most r and domain size d, where $r \geqslant 2$, this identification problem is tractable if and only if $d \leqslant 2$.

6. Domain permutation reductions to tractable languages

In this section we will demonstrate that our new theory can be used to show the tractability of certain new subclasses of CSP instances that are not tractable for any relational or structural reason. We will also show that we can explain the tractability of three published disparate tractable subclasses of the general constraint satisfaction problem that have not been amenable to other general explanations for tractability. Finally we will correct a previously published result regarding the tractability of the connected row convex identification problem.

6.1. Two permutations and a binary language

Our new theory can be used to show the tractability of certain subclasses that are not tractable for any relational or structural reason. We will show that when the set of allowed permutations, G, has size 2, the G-reduction problem into any Γ is tractable for the entire set of binary relations, Δ, over the finite domain D. We will use this result in Example 41 to construct a natural tractable class which extends two maximal tractable relational classes.

Lemma 39. *Let Γ be a constraint language over finite domain D, G be a set of two permutations of D, and Δ be the set of all binary relations over D. The G-reduction problem from Δ into Γ is tractable.*

Proof. It is enough to note that any G-lifted relation from Δ into Γ is binary two valued. It follows that the G-lifted problem from Δ into Γ may be reduced to 2-SAT, which is tractable. Since D is finite we know that Δ is also finite. By Corollary 29 we deduce that the G-reduction problem from Δ into Γ is tractable. □

Corollary 40. *Let Γ be a tractable constraint language over finite domain D, G be a set of two permutations of D, and Δ be the set of all binary relations over D. The set of instances of* CSP(Δ) *which are G-reducible to Γ is tractable.*

Example 41. Let D be the ordered domain $\{1, \ldots, k\}$. Let $G = \{\iota, \tau\}$ where $\iota(e) = e$ and $\tau(e) = k - e + 1$.

By Lemma 39 we have that the G-reduction problem from the set of all binary relations over D into the binary max-closed constraint language is tractable.

The tractable set of instances defined by this reduction, according to Corollary 40, certainly includes all CSP instances whose language is binary max-closed. In this case the domain permutation reduction applies the permutation ι to each variable. This subclass also naturally includes the analogous tractable set of binary min-closed instances. We can reduce these to the max-closed constraint language by applying τ to each variable. In fact, there are instances which are neither max-closed nor min-closed. We may obtain such instances from a binary max-closed instance by arbitrarily applying τ to some subset of the variables.

Since the max-closed binary relations form a maximal tractable binary constraint language (see Theorem 6.5 of Jeavons and Cooper [36]) we can see that the tractable subclass of this example is not tractable for any relational reason. Furthermore it includes instances with arbitrary (binary) structure so it is not a structurally tractable subclass.

Many novel tractable subclasses can be described using Corollary 40.

6.2. The Stable Marriage Problem

An instance of the *Stable Marriage Problem* (*SMP*) [23] consists of n men and n women who are to be married. Each man has a preference list that ranks the women. He would prefer to marry those higher in the list than those lower in the list. Similarly each woman has a preference list for the men. The problem is to find a *stable* set of marriages. A set of marriages is stable if:

- Every person is in just one marriage.
- There is no man, m, and woman, w, such that m prefers w to his wife, and w prefers m to her husband.

It is known that the Stable Marriage Problem is polynomially solvable. In particular, arc consistency is a decision procedure [27] for a particular representation of this problem. Informally, each variable is a man and the domain is the set of women. Here we are choosing a woman for each man. Arc consistency will remove some women from the set of allowed partners for each man. A solution is then for each man to marry his most preferred remaining partner.

More formally, an instance of the Stable Marriage Problem is a constraint satisfaction problem instance, $P = \langle V, D, C\rangle$, where:

- $V = \{1, \ldots, n\}$;
- $D = \{1, \ldots, n\}$;
- For each $m \in V$, π_m is a permutation of D, and for each $w \in D$, τ_w is a permutation of V.
  $C = \{c_{s,t} \mid s, t \in V, s < t\}$, where $c_{s,t} = \langle\langle s, t\rangle, \rho_{s,t}\rangle$, and

$$\begin{aligned} \langle p, q\rangle \in \rho_{s,t} \quad \Leftrightarrow \quad & p \neq q \wedge \\ & \tau_p(t) > \tau_p(s) \quad \Rightarrow \quad \pi_t(q) > \pi_t(p) \wedge \\ & \tau_q(s) > \tau_q(t) \quad \Rightarrow \quad \pi_s(p) > \pi_s(q). \end{aligned}$$

We interpret V as a set of men, and D as a set of women. We interpret $\pi_m(w)$ to be the level of preference (n is the most preferred and 1 the least preferred) that man m has for woman w. Similarly, $\tau_w(m)$ defines the level of preference given to man m by woman w.

We will show that every SMP instance is reducible to a max-closed instance. What is more, the required domain permutation orders the domain for each man according to his preferences amongst the women. This completely explains the known solution algorithm. Since the preference orderings are known, we will have shown that the set of SMP instances is an example of the relational reduction described in this paper.

Definition 42. Let P be an SMP instance. Consider any pair of marriages and let p be one of the four people involved. We say p is *happy* with respect to this pair of marriages if they prefer their partner to the person (of the same gender) in the other marriage.

A pair of marriages is *stable* if there is no unmarried unhappy man and woman.

Theorem 43. *Let P be an SMP instance and define, for each $m \in V$, $\Pi(m) = \pi_m$. Then $\Pi(P)$ is max-closed.*

Proof. Consider a pair of variables s and t. We require to show that $\Pi(c_{s,t})$ is max-closed. In other words we have to show that $\rho = \{\langle \pi_s(p), \pi_t(q)\rangle \mid \langle p, q\rangle \in \rho_{s,t}\}$ is max-closed.

Choose some $a, b, e, f \in D$, $\langle a, f\rangle, \langle b, e\rangle \in \rho_{s,t}$. If $\pi_s(a) < \pi_s(b)$ and $\pi_t(f) < \pi_t(e)$ then we have that

$$\max\big(\big\langle \pi_s(a), \pi_t(f)\big\rangle, \big\langle \pi_s(b), \pi_t(e)\big\rangle\big) = \big\langle \pi_s(b), \pi_t(e)\big\rangle \in \rho$$

and we are done. Similarly we are done if $\pi_s(b) < \pi_s(a)$ and $\pi_t(e) < \pi_t(f)$. So, without loss of generality we may assume that

$$\pi_s(a) < \pi_s(b) \quad \text{and} \quad \pi_t(e) < \pi_t(f).$$

We need to show that $\langle \pi_s(b), \pi_t(f)\rangle \in \rho$. This is equivalent to proving that $\langle b, f\rangle \in \rho_{s,t}$. To prove this we must show that $b \neq f$ and that there is no unmarried unhappy man and woman in the pair of marriages (s to b and t to f).

Assume for contradiction that $b = f$. Then both men prefer b to their other potential partner. However, woman b must prefer one of the men. It is clear that she cannot marry her less preferred man. This contradicts the fact that both $\langle a, f\rangle$ and $\langle b, e\rangle$ are in $\rho_{s,t}$. Hence, $b \neq f$.

Now assume that s is unhappy in the pair of marriages (s to b and t to f). Since s is unhappy he prefers f to b. We know that $\pi_s(a) < \pi_s(b)$ and so by transitivity s prefers f to a. Since $\langle a, f\rangle \in \rho_{s,t}$ we know that the pair of marriages s to a and t to f is stable, and so if s prefers f to a then f prefers t to s. Hence f is indeed happy. So, by symmetry, the pair of marriages s to b and t to f is stable. □

This result leaves an open question. Whilst we have shown a reduction from SMP to the max-closed class exists we have not shown that it is polynomial to determine. The reduction does serve to explain why arc consistency, and choosing the most preferred remaining woman for each man, is a polynomial solution algorithm. However, in order to show that SMP is a tractable class we also have to identify the SMP instances. That is, given a binary CSP instance, we have to determine whether there is a domain permutation which makes it an SMP instance. We conjecture that this identification problem is tractable.

Conjecture 44. *The Stable Marriage Problem is a tractable subclass of the general constraint satisfaction problem.*

6.3. Triangulated CSP instances

Cohen [12] defined the triangulated CSP instances, for which arc consistency is a decision procedure. It has been shown that the set of triangulated CSP instances is incomparable with any other known subclass for which arc consistency is a decision procedure. It was left as an open question as to whether the triangulated CSP instances can be reduced to a tractable relational subclass. In this section we will show that triangulated CSP instances are simply another subclass of the general constraint satisfaction problem that can be polynomially reduced to the max-closed constraint language.

In order to define the triangulated subclass we need the notion of the complement of the microstructure of a CSP instance [39]. In general this is a hypergraph and is defined in [12,39]. For our purposes it is enough to define the graph which is the complement of the microstructure for binary CSP instances.

Definition 45. The *complement of the microstructure* of the binary CSP instance, $P = \langle V, D, C\rangle$, denoted $\overline{\mathcal{M}}(P)$, is a graph $\langle W, F\rangle$, defined as follows:

$$\begin{aligned} W &= V \times D; \\ F &= \big\{\big\{\langle v, d\rangle, \langle v, e\rangle\big\} \bigm| d \neq e\big\} \cup \\ &\quad \big\{\big\{\langle v_1, d_1\rangle, \langle v_2, d_2\rangle\big\} \bigm| \exists\big\langle\langle v_1, v_2\rangle, \rho\big\rangle \in C, \langle d_1, d_2\rangle \notin \rho\big\}. \end{aligned}$$

Fig. 1. There are two possible configurations of a binary relation that are an impossible cross. These prevent the relation from being max-closed.

Fig. 2. There are two possible configurations of a binary relation that are an impossible cross. This figure shows what they look like in the *complement of the microstructure graph* (assuming this relation has been used in a constraint with scope $\langle v, w\rangle$).

An *independent set* in a graph is a set of vertices, no two of which are connected by an edge. We can see from the definition that solutions to P correspond exactly to independent sets in $\overline{\mathcal{M}}(P)$ containing $|V|$ vertices.

Definition 46. Let $G = \langle V, E\rangle$ be a graph.

A sequence S of vertices of G is a *path* if consecutive vertices in S are adjacent in G. The path S is a *cycle* if there is an edge between the first vertex and the last vertex of the sequence. The cycle is *chordal* if there is an edge between any pair of non-adjacent vertices in S. The graph G is *triangulated* if every cycle is chordal.

A set M of vertices of G is a *clique* if every pair of vertices of M is an edge of G.

A binary CSP instance P is *triangulated* [12] if $\overline{\mathcal{M}}(P)$ is triangulated.

6.3.1. Reduction to the max-closed subclass

Definition 47. Let $<$ be an ordering of the vertices of a graph G. We say that $<$ is an *elimination ordering* if, for any vertex v of G, the set of vertices that are smaller than, and connected to, v form a clique.

It is well-known [45] that every triangulated graph admits an elimination ordering. In the remainder of this section we will show that triangulated CSP instances are in fact reducible to the max-closed constraint language. What is more, for any triangulated CSP instance, P, we will show that the necessary domain permutation can be found directly from an elimination ordering of the vertices of $\overline{\mathcal{M}}(P)$.

Definition 48. A *configuration* is a binary relation on the ordered Boolean domain $\{0, 1\}$. Let ρ be a binary relation on the ordered domain D and let $a_0, a_1, b_0, b_1 \in D$ with $a_0 < a_1$, $b_0 < b_1$.

The configuration of ρ at $\langle\{a_0, a_1\}, \{b_0, b_1\}\rangle$ is the binary relation over the Boolean domain $\{0, 1\}$ which allows the tuple $\langle x, y\rangle$ exactly when $\langle a_x, b_y\rangle \in \rho$.

An *impossible cross* for ρ is a configuration, η, of ρ where:

- $\langle 0, 1\rangle, \langle 1, 0\rangle \in \eta$;
- $\langle 1, 1\rangle \notin \eta$.

We need the following property of max-closed relations (proved as Lemma 6.2 of Jeavons and Cooper [36]).

Proposition 49. *A binary relation ρ is max-closed if and only if none of its configurations is an impossible cross.*

We want to show that triangulated CSP instances are reducible to max-closed CSP instances, so that under some domain permutation there exists no impossible cross. There are exactly two impossible cross configurations, shown[1] in Fig. 1, whilst Fig. 2 shows what these same configurations look like in the complement of the microstructure graph. The left hand configuration cannot occur in the constraint relation of any constraint in any triangulated CSP instance since it leads to a non-chordal cycle of four vertices in the complement of the microstructure graph.

[1] We represent binary relations diagrammatically by connecting pairs of domain values when the corresponding tuple is in the relation.

Theorem 50. *Let $P = \langle V, D, C\rangle$ be a triangulated CSP instance. Then P is reducible to a max-closed CSP instance by domain permutation.*

Proof. Let $<$ be an elimination ordering of the vertices of $\overline{\mathcal{M}}(P)$.

For any variable $v \in V$, this elimination ordering induces an ordering, $<_v$, of the domain D at v. This ordering is such that for two domain values $a, b \in D$, $a <_v b$ if and only if $\langle v, a\rangle < \langle v, b\rangle$ in the elimination ordering of $\overline{\mathcal{M}}(P)$.

We will show that for any constraint $\langle \sigma, \rho\rangle \in C$, where $\sigma = \langle v, w\rangle$, ρ is max-closed under the domain orderings $<_v$ at variable v and $<_w$ at variable w. We are required to prove that under these orderings ρ contains no impossible cross.

Consider the configuration κ of ρ at $\langle\{a, b\}, \{e, f\}\rangle$ where $a <_v b$ and $e <_w f$. Assume for contradiction that κ is an impossible cross (and hence for a triangulated CSP instance looks like the right hand configuration given in Fig. 2). Since κ is symmetrical we may assume, without loss of generality, that $\langle v, b\rangle < \langle w, f\rangle$ in the elimination ordering.

Now, we know that $<$ is an elimination ordering of the vertices of $\overline{\mathcal{M}}(P)$. As such, we know that if $\langle w, f\rangle$ is connected to $\langle v, b\rangle$ (as in both of the impossible cross cases) then $\langle v, b\rangle$, $\langle w, e\rangle$ and $\langle w, f\rangle$ must form part of a clique and hence the edge $\{\langle v, b\rangle, \langle w, e\rangle\}$ must exist in $\overline{\mathcal{M}}(P)$. This means that $\langle b, e\rangle \notin \rho$. However, this contradicts the assumption that κ is an impossible cross.

As such, no configuration of ρ is an impossible cross under the domain orders ($<_v$ and $<_w$) imposed by the elimination ordering $<$ and hence ρ is max-closed under these domain orderings. □

6.3.2. Tractability of the reduction

Definition 51. A *maximal cardinality ordering* of the vertices of the graph G is constructed in $|V|$ steps as follows. At step 1, choose any vertex v_1. At step i, for $i < |V|$ we have chosen $v_1, \ldots, v_{i-1}$. Now choose for v_i any vertex that is connected to the largest set of previously numbered vertices. We order $v_i < v_j$ if $i < j$.

The following is a well-known result [45].

Lemma 52. *A maximal cardinality ordering of the vertices of the graph G is an elimination ordering if and only if G is triangulated.*

It follows directly that the triangulated CSP instances are a tractable subclass of the general constraint satisfaction problem. What is more, the following theorem (Theorem 3 of 12) is a direct consequence of the reduction demonstrated in this paper.

Theorem 53. *Arc consistency is a decision procedure for the triangulated CSP instances.*

The open conjecture has been proven. Arc consistency is a decision procedure for triangulated CSP instances for exactly the same reason as it is a decision procedure for max-closed CSP instances.

6.4. Identifying connected row convex instances

Recall the row convex (binary) relations of Definition 22. In this section we consider a subset of these relations: the so-called connected row convex constraints [20]. It is known that path consistency implies global consistency [20, 49], and so we can tractably solve path consistent CSP instances if we know that, under any (independent) domain re-ordering for each variable, each constraint relation is connected row convex.

Definition 54. The *connected row convex identification problem* for binary CSP instances is to find a domain permutation which results in a connected row convex instance.

The tractability of the connected row convex identification problem was left as an open problem [20]. An incorrect answer to this question has unfortunately appeared in the literature [10].

We will determine the complexity of the connected row convex identification problem by establishing the complexity of the associated lifted language.

Definition 55. The *inverse* of a binary relation ρ is the relation

$$\rho^{-1} = \{\langle i, j\rangle \mid \langle j, i\rangle \in \rho\}.$$

The *reduced form* of a relation ρ, denoted ρ^*, is the (0, 1)-matrix obtained by removing from the matrix representation of ρ all rows and columns that contain only 0's.

Let M be any (0, 1)-matrix. For any $i \in D$ such that there exists $j \in D$ with $M\langle i, j\rangle = 1$ define the image of i in M to be the interval

$$\big[\min\{j \mid M\langle i, j\rangle = 1\}, \max\{j \mid M\langle i, j\rangle = 1\}\big].$$

A binary relation ρ is *connected* if the images $[a, b]$ and $[a', b']$ of two consecutive rows in ρ^* are such that $a \leqslant b' - 1$ and $a' \leqslant b + 1$.

A relation ρ is *connected row convex* (CRC) if ρ^* and $(\rho^{-1})^*$ are both row convex and connected.

Deville et al. [20] show that CRC relations are closed under the three necessary operations needed for path consistency algorithms and that path consistency implies global consistency for problem instances whose constraint relations are all CRC. This means that CRC relations form a tractable constraint language.

Definition 56. The *median* operator over the ordered domain $\{1, \ldots, k\}$ is a ternary operator that returns the median, or middle, value of its arguments. The median operator is a majority operator (recall Section 3.1) and so defines a tractable constraint language.

It has been shown [38] that the connected row convex relations are precisely the binary relations closed under the median operator. Cohen et al. [14] present another way of viewing connected row convex constraints (as disjunctions of simpler constraints). However, for our purposes it is sufficient to consider the median-closed relations.

By Corollary 29, the connected row convex identification problem (for any finite domain) is equivalent to solving the lifted problem into the language of median-closed relations.

In this section we will determine the complexity of these lifted languages.

We used Polyanna initially to determine the complexity of the lifted languages but we also verified the results by hand. This means that no proofs in this section rely solely on computer generated results.

6.4.1. Reducing the lifted languages

There are very many possible polymorphisms that Polyanna has to consider. For this reason it is vital first to reduce the size of relations before Polyanna begins. The lifted relations in this case have a natural simplification.

Let M_ρ be the matrix representation of a CRC relation ρ. Reversing the order of either the rows or the columns of M_ρ cannot affect the CRC property because it does not affect which rows/columns are consecutive. It follows that if we are looking for pairs of permutations that make a binary relation ρ CRC, then we can factor out half of the possible permutations by, for instance, assuming that the image of 1 is always less than the image of k.

6.4.2. The three valued domain

We label the six permutations of $\{1, 2, 3\}$ as in Fig. 3. By the argument of Section 6.4.1 we only need to consider the permutations $G = \{\pi_0, \pi_1, \pi_2\}$, which dramatically reduces the cost of searching for polymorphisms of the lifted relations.

We used a program to generate the lifted language for the 512 binary relations over the domain $\{1, 2, 3\}$.

There are 92 relations in this language and Polyanna determined that it is closed under the majority operator which returns the first of three distinct values. It follows [38] that the language is tractable and so the connected row convex identification problem is tractable. So we obtain the following results.

$$\pi_0 : \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3\} \qquad \pi_3 : \{1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 2\}$$
$$\pi_1 : \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 2\} \qquad \pi_4 : \{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1\}$$
$$\pi_2 : \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3\} \qquad \pi_5 : \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1\}$$

Fig. 3. The six possible domain orders (permutations) of a domain of size three.

$$\pi_0 : \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto 4\} \qquad \pi_6 : \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3, 4 \mapsto 4\}$$
$$\pi_1 : \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 4, 4 \mapsto 3\} \qquad \pi_7 : \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 4, 4 \mapsto 3\}$$
$$\pi_2 : \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 2, 4 \mapsto 4\} \qquad \pi_8 : \{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1, 4 \mapsto 4\}$$
$$\pi_3 : \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 2\} \qquad \pi_9 : \{1 \mapsto 2, 2 \mapsto 4, 3 \mapsto 1, 4 \mapsto 3\}$$
$$\pi_4 : \{1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 2, 4 \mapsto 3\} \qquad \pi_{10} : \{1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 2, 4 \mapsto 4\}$$
$$\pi_5 : \{1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 3, 4 \mapsto 2\} \qquad \pi_{11} : \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1, 4 \mapsto 4\}$$

Fig. 4. The twelve possible domain orders (permutations) of the domain $\{1, 2, 3, 4\}$.

Theorem 57. *The connected row convex identification problem for the three valued domain is tractable.*

Proof. The lifted language is closed under a majority operator and is therefore tractable. Since the lifted language is finite we can use Corollary 29 to show that the reduction problem is also tractable. □

Corollary 58. *The set of binary three valued CSP instances which can be made into CRC instances by permutation of the domain independently at each variable is tractable.*

6.4.3. The four valued domain

There are $2^{4^2} = 65536$ binary relations over the domain $\{1, 2, 3, 4\}$. The number of permutations of this domain is 24. After factoring out reflections as in Section 6.4.1 we still have 12 permutations (given in Fig. 4) for the domain of the lifted relations. Using a dedicated program we discovered 14501 lifted relations.

We will show the *intractability* of the reduction problem by showing the *intractability* of a finite subset of this lifted language and appealing to Proposition 28.

Consider the two binary four valued relations, ρ_1 and ρ_2, shown at the top of Fig. 5. The lifted relations for these two relations, λ_1 and λ_2 respectively, are also given in Fig. 5.

A further simplification that we apply is to note that any unary polymorphism of a constraint language can be applied to all of the relations in the language without changing the complexity.

Proposition 59. *(See [13,34,37].) Let Γ be a constraint language over a set D, and let f be a unary polymorphism of Γ.* $\mathrm{CSP}(\Gamma)$ *is polynomial-time equivalent to* $\mathrm{CSP}(f(\Gamma))$*, where* $f(\Gamma) = \{f(\rho) \mid \rho \in \Gamma\}$ *and* $f(\rho) = \{f(t) \mid t \in \rho\}$.

Polyanna discovered a unary polymorphism of the lifted relations for λ_1 and λ_2.

Application of Proposition 59 means that the tractability of this pair of (full domain) lifted relations is equivalent to the tractability of the pair of three valued relations, $\upsilon(\lambda_1)$ and $\upsilon(\lambda_2)$, also shown in Fig. 5.

Polyanna was able to prove this language intractable. A relation over the domain $\{1, 2, 3, 4\}$ can be seen as a relation over the domain $\{1, \ldots, k\}$, $k > 4$ that never allows any tuples with domain values larger than 4. It follows that the connected row convex identification problem for domain size four can be reduced to the connected row convex identification problem for any larger domain, and so these are also intractable problems.

Theorem 60. *The connected row convex identification problem is intractable for domains of size four or more.*

6.4.4. Verifying the complexity results

Since Theorem 60 contradicts a result published in a refereed conference we need to avoid any reliance on computer generated results.

We generated the lifted relations for λ_1 and λ_2 by hand. We also checked the unary polymorphism υ.

Lastly, we needed to verify that $\{\upsilon(\lambda_1), \upsilon(\lambda_2)\}$ is indeed an intractable constraint language. One way of doing so is to use combinations of $\upsilon(\lambda_1)$ and $\upsilon(\lambda_2)$ to generate a known intractable relation, such as the inequality relation over a three valued domain. This would make graph 3-colouring reduce to the connected row convex identification problem for domain size four.

Consider the instance of Fig. 6. It is tedious but straightforward to check that the implied constraint between v_1 and v_2 is indeed the three valued inequality relation and hence the three valued inequality relation can be expressed by $\upsilon(\lambda_1)$ and $\upsilon(\lambda_2)$.

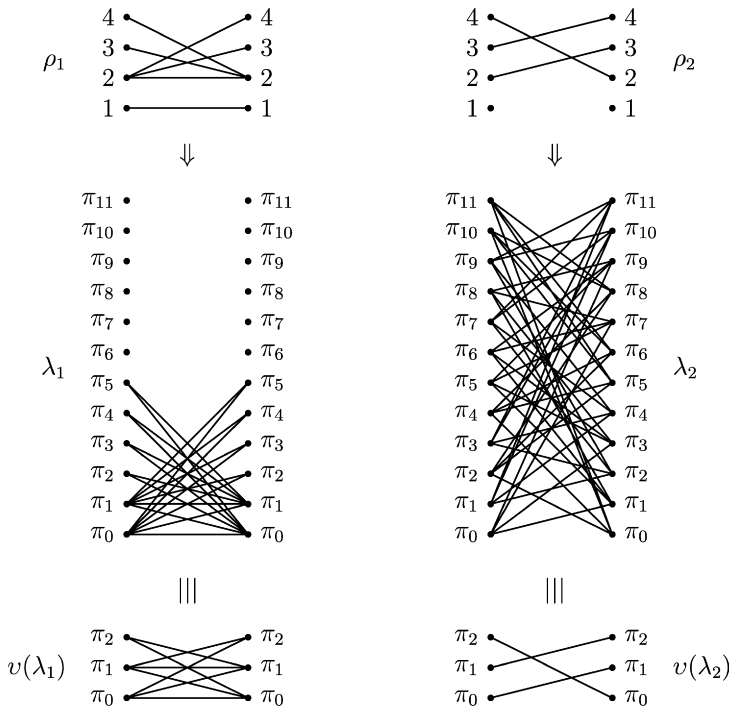

Fig. 5. Relations ρ_1 and ρ_2 with their lifted relations, λ_1 and λ_2 (after factoring out reflections). Also the equivalent intractable language $\upsilon(\lambda_1)$ and $\upsilon(\lambda_2)$, formed by applying the unary polymorphism υ.

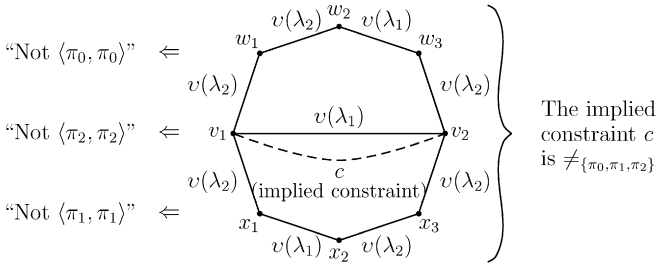

Fig. 6. We use the pair of relations, $\upsilon(\lambda_1)$ and $\upsilon(\lambda_2)$, to generate a known intractable relation, the inequality relation over a three valued domain.

6.4.5. A weakness in the standardised form of CRC relations

We have not shown that P = NP! Instead we have exposed a flaw in the previously published result [10]. Here we make explicit this flaw.

Chen et al. [10] propose a polynomial time algorithm for identifying connected row convex constraint satisfaction problem instances under an independent domain ordering at each variable. Their method relies on a standardised form of CRC relations that they claim can be recognised in polynomial time. Our results are straightforward and hand verified.

Definition 61. (See Definition 9 of Chen et al. [10].) Given some connected row convex relation ρ over domain size k let M_ρ be its (0, 1)-matrix representation with k rows and columns. Let $\min_\rho(i)$ denote the position of the leftmost

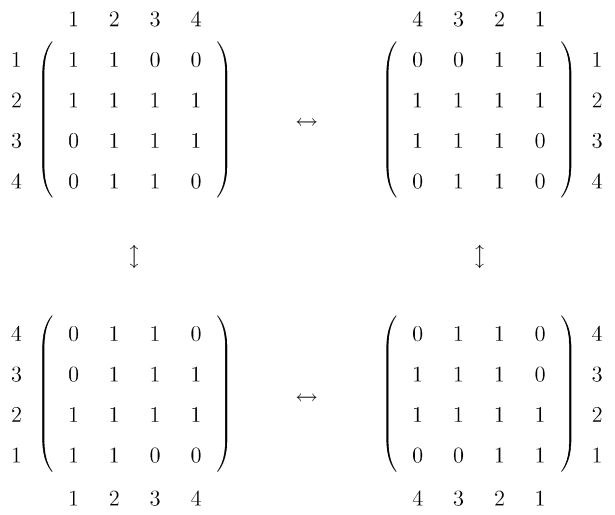

Fig. 7. This table shows the only combinations of permutations that keep the relation in the upper left CRC. It is straightforward to check that none of these are in standardised form.

one in row i of M_ρ and $\max_\rho(i)$ the position of the rightmost one in row i of M_ρ. ρ is called a *standard CRC relation* if:

- $\min_\rho(1) \leqslant \min_\rho(2) \leqslant \cdots \leqslant \min_\rho(k)$;
- $\min_\rho(i) = \min_\rho(i+1) = \cdots = \min_\rho(i+j) \Rightarrow \max_\rho(i) \leqslant \max_\rho(i+1) \leqslant \cdots \leqslant \max_\rho(i+j)$;

where $1 \leqslant i, i+j \leqslant k$.

We will now show that there exist relations that are CRC but which cannot be standardised. Consider the relation of Fig. 7. This binary relation is connected row convex and it can only remain as such under domain reflections.

It can be seen from Fig. 7 that none of the four combinations of domain reflections standardise this CRC relation. It follows that the algorithm of Chen et al. [10] does not in fact solve the more general reduction problem from the set of all binary relations into the CRC constraint language. Instead, it solves a subset of this problem for those instances that can be made standardised CRC. Any solution found by this algorithm is a solution to the lifted problem instance. However, many solutions to the lifted problem instance cannot be found by this algorithm.

6.5. The renamable Horn subclass

Recall the definition of clauses and literals from Example 7. Propositional satisfiability (SAT) is the problem of determining, for a given set of clauses, whether there is a truth assignment to all of the propositional variables that satisfies all of the clauses. There have been many papers published describing the so-called renamable Horn class of SAT instances [2,3,6,8,9,42].

It has been shown that the renamable Horn subclass is tractable, in that recognising, and hence solving, SAT instances that are renamable Horn can be performed in polynomial time. Indeed del Val [19] gives an algorithm which solves both 2-SAT and renamable Horn instances which he calls "the paradigmic examples of tractable problems in propositional satisfiability".

In this section we define the class of renamable Horn SAT instances and the analogous class of renamable Horn CSP instances. We show that the tractability of both classes is due to a tractable reduction problem into the tractable language of propositional Horn clauses [47].

It has also been shown [6] that the problem of identifying a maximal subset of the clauses of an instance that are renamable Horn is NP-hard. A well-known properties of the lifted language for the language of Horn clauses shows directly that this is an NP-hard problem.

Definition 62. A *Horn clause* is a clause with at most one positive literal. A set of clauses is *renamable Horn* if there is a replacement of some literals uniformly in all clauses with their negated versions which transforms all clauses into Horn clauses.

A Boolean CSP instance is *renamable Horn* when its constraint relations all represent clauses and there exists a domain permutation which transforms all the constraint relations into Horn clauses.

6.5.1. The tractability of renamable Horn explained

Solving a renamable Horn instance, given an appropriate domain permutation, is easy. We apply the domain permutation and then solve the resulting instance using unit resolution, the solution algorithm for Horn [47]. It is therefore the discovery of an appropriate domain permutation that we have to show to be tractable.

Proposition 63. *Let c be a clause. Let S_c be the set of variables which must be set to F (all others being set to T) which makes the clause evaluate to F. Then the lifted relation for c into the Horn constraint language consists of all those tuples which invert the domain of a set of variables ψ where the symmetric difference between ψ and S_c has size at most one.*

Proof. Follows directly from the definition of a Horn clause. □

Example 64. Consider the clause $c = p \vee q \vee \bar{r} \vee \bar{s}$. The only assignment that does not satisfy c is $p \mapsto F$, $q \mapsto F$, $r \mapsto T$, $s \mapsto T$. That is, $S_c = \{p, q\}$.

There is exactly one non-trivial permutation τ of the domain $\{F, T\}$: it interchanges T and F. The trivial permutation ι fixes both T and F.

With respect to the order $\langle p, q, r, s\rangle$ the tuple of permutations which negate precisely those in S_c is $\Pi_0 = \langle\tau, \tau, \iota, \iota\rangle$. The clause $\Pi_0(c)$ is $\overline{p} \vee \overline{q} \vee \bar{r} \vee \bar{s}$ which is a Horn clause.

Alternatively, negate p, q and r with the tuple of permutations $\Pi_1 = \langle\tau, \tau, \tau, \iota\rangle$. In this case $\Pi_1(c)$ is $\overline{p} \vee \overline{q} \vee r \vee \bar{s}$, again Horn. The symmetric difference between S_c and $\{p, q, r\}$ has size one.

Conversely, the symmetric difference between S_c and the set $\{p, r\}$ has size two. Let $\Pi_2 = \langle\tau, \iota, \tau, \iota\rangle$ and $\Pi_2(c)$ is $\overline{p} \vee q \vee r \vee \bar{s}$ which is not Horn.

Recall from Section 3.1 that a set of relations closed under majority is tractable.

For the Boolean domain there is precisely one majority operator which we will denote by δ.

Proposition 65. *The lifted language from the set of all clauses into the Horn constraint language is majority-closed.*

Proof. Consider a clause c. Let λ_c be its lifted relation into the Horn constraint language and let l_1, l_2, l_3 be three tuples from λ_c. We have to prove that $\delta(l_1, l_2, l_3) \in \lambda_c$. The result is trivial if two of the tuples are identical so we may assume that they are distinct.

By Proposition 63 we see that any variable of S_c must correspond to the permutation τ in at least two of these tuples. Hence $\delta(l_1, l_2, l_3)$ is equal to τ precisely for those variables in S_c and we are done. □

Clauses in SAT instances are represented implicitly as a disjunct of literals. By Proposition 63 it is clear that any lifted r-ary relation has at most $r + 1$ tuples. So there is a polynomial algorithm for generating lifted instances for the general reduction problem from SAT into Horn-SAT.

Hence Propositions 65 and 27 provide a simple explanation of the tractability of the renamable Horn class of SAT instances.

6.5.2. A new tractable class

Having explained the tractability of the (implicitly expressed) renamable Horn class of SAT instances we can go further and show that they are tractable as a class of Boolean CSP instances. We have only to show that it is tractable to determine if each relation in an instance represents a clause. Since there is precisely one assignment that does not satisfy a clause, counting the allowed assignments is sufficient to determine whether a constraint represents a clause.

Proposition 66. *The renamable Horn class of CSP instances is tractably identifiable.*

Example 67. Recall the split-Horn subclass defined in Example 7. We can now easily show the tractability of this class. The instances are all renamable Horn.

6.5.3. Max-renamable Horn

Consider the problem of identifying, for any SAT instance, a maximum number of those clauses that are renamable Horn. This is the *max-renamable Horn problem*.

The MAX-CSP problem is to determine, for a given CSP instance, the maximum number of its constraints that can be simultaneously satisfied.

Identifying maximum subsets of the clauses which are renamable Horn corresponds precisely to solving instances of the MAX-CSP problem for the lifted language. Creignou [16] characterised the tractability of all Boolean constraint languages for the MAX-CSP problem. There are three tractable cases.

We can readily determine that the lifted language is not contained in any of the three tractable languages. Explicitly, the lifted relation for the clause c defined in Example 64 is in none of these languages. The intractability of the max-renamable Horn problem then follows immediately by Proposition 28.

7. Conclusion

In this paper we have identified genuinely novel tractable classes of CSP instances that are neither relational nor structural classes. We have even constructed a tractable class which properly extends two maximal tractable relational classes.

We have considered whether it is feasible to identify constraint satisfaction problem instances that become max-closed after an independent domain re-ordering at each variable. We have shown that in the Boolean case it is tractable to identify such instances when the arity of the instances is bounded. We have also shown that for larger domain sizes it is (in general) intractable to identify such instances.

Furthermore we have demonstrated that this theory serves to explain why the constraint approach to the Stable Marriage Problem is tractably solvable, since all instances reduce to max-closed instances. We have also shown that the triangulated instances are reducible to the max-closed constraint language. In this sense we have unified three tractable classes of CSP instances (max-closed, SMP and triangulated) for which arc consistency is a decision procedure.

It might be hoped that this theory would also explain why arc consistency is a decision procedure for tree structured CSP instances. Perhaps all such instances are actually reducible to max-closed instances. Unfortunately, this is not the case since, for instance, there are six binary three valued relations with empty lifted relations for the max-closed constraint language (see Section 5.2). So, we still need to find a unifying theory for all CSP instances for which arc consistency is a decision procedure.

For the class of CSP instances with a given domain we considered the complexity of finding domain permutations that transform a given instance into a connected row convex instance. An incorrect answer, that this is tractable for every domain, has been published at a refereed conference. In this paper we have shown, using easily verifiable results, that this problem is tractable precisely when the domain has at most three elements.

Lastly, this theory has provided a simple explanation of the tractability of recognising instances that are renamable Horn. It even serves to explain why finding maximum subsets of clauses which are renamable Horn is NP-hard.

This new method of reduction to relational tractability has already had surprising and extensive applications in simplifying our theoretical understanding. It may well also be a practical reduction procedure for some real instances.

Acknowledgements

The authors are very grateful to Marc van Dongen [50] who originally derived the SMP example of a reduction to the max-closed tractable subclass, though in a different context. His example as an application of reductions to tractable language theory greatly strengthens our work. We are also grateful to Peter Jeavons who alerted us to the connections between this work and renamable Horn theory. Finally, we would like to thank an anonymous reviewer

of an earlier draft of part of this work for alerting us to the need for polynomial conversion between the reduction problem and the lifted problem, resulting in Propositions 27, 28 and Corollary 29.

References

[1] J. Allen, Natural Language Understanding, second ed., The Benjamin/Cummings Publishing Company, 1995.
[2] B. Aspvall, M.F. Plass, R.E. Tarjan, A linear time algorithm for testing the truth of certain quantified Boolean formulas, Information Processing Letters 8 (1979) 121–123.
[3] B. Aspvall, Recognizing disguised NR(1) instances of the satisfiability problem, Journal of Algorithms 1 (1) (1980) 97–103.
[4] C. Beeri, R. Fagin, D. Maier, M. Yannakakis, On the desirability of acyclic database schemes, Journal of the ACM 30 (1983) 479–513.
[5] C. Bessière, J.-C. Régin, Arc consistency for general constraint networks: preliminary results, in: Proceedings of IJCAI'97, Nagoya, Japan, 1997, pp. 398–404.
[6] E. Boros, Maximum renamable Horn sub-CNFs, Discrete Applied Mathematics 96–97 (1999) 29–40.
[7] A. Bulatov, P. Jeavons, A. Krokhin, Classifying the complexity of constraints using finite algebras, SIAM Journal on Computing 34 (3) (2005) 720–742.
[8] V. Chandru, J.N. Hooker, Extended Horn sets in propositional logic, Journal of the ACM 38 (1991) 205–221.
[9] V. Chandru, C. Coullard, P. Hammer, M. Montanez, X. Sun, On Renamable Horn and Generalized Horn functions, Annals of Mathematics and Artificial Intelligence 1 (1990) 33–48.
[10] E. Chen, Z. Zhang, X. Wang, K. Aihara, An algorithm for fast recognition of connected row-convex constraint networks, in: JFPLC 2001, The Tenth International French Speaking Conference on Logic and Constraint Programming, April 2001, pp. 43–58.
[11] D. Cohen, Tractable decision for a constraint language implies tractable search, Constraints 9 (3) (2004) 219–229.
[12] D.A. Cohen, A new class of binary CSPs for which arc-consistency is a decision procedure, in: Principles and Practice of Constraint Programming—CP 2003, in: Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 807–811.
[13] D. Cohen, P. Jeavons, The complexity of constraint languages, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006 (Chapter 8).
[14] D. Cohen, P. Jeavons, P. Jonsson, M. Koubarakis, Building tractable disjunctive constraints, Journal of the ACM 47 (5) (2000) 826–853.
[15] D.A. Cohen, P. Jeavons, M. Gyssens, A unified theory of structural tractability for constraint satisfaction and spread cut decomposition, in: IJCAI, 2005, pp. 72–77.
[16] N. Creignou, A dichotomy theorem for maximum generalized satisfiability problems, Journal of Computer and System Sciences 51 (1995) 511–522.
[17] R. Dechter, J. Pearl, Network-based heuristics for constraint satisfaction problems, Artificial Intelligence 34 (1) (1988) 1–38.
[18] R. Dechter, J. Pearl, Tree clustering for constraint networks, Artificial Intelligence 38 (1989) 353–366.
[19] A. del Val, On 2-SAT and renamable Horn, in: AAAI: 17th National Conference on Artificial Intelligence, AAAI/MIT Press, 2000, pp. 279–284.
[20] Y. Deville, O. Barette, P. van Hentenryck, Constraint satisfaction over connected row convex constraints, Artificial Intelligence 109 (1999) 243–271.
[21] N.W. Dunkin, J.E. Bater, P.G. Jeavons, D.A. Cohen, Toward high order constraint representations for the frequency assignment problem, Technical Report CSD-TR-98-05, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey, UK, 1998.
[22] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, Journal of the ACM 30 (1983) 514–550.
[23] D. Gale, L.S. Shapley, College admissions and stability of marriage, The American Mathematical Monthly 69 (1) (1962) 9–15.
[24] M. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[25] R.L. Gault, Polyanna technical manual (version 1.00), Technical Report RR-01-20, Computing Laboratory, Oxford University, 2001.
[26] R.L. Gault, P.G. Jeavons, Implementing a test for tractability, Constraints 9 (2) (2004) 139–160.
[27] I.P. Gent, R. W Irving, M. F Manlove, P. Prosser, B.M. Smith, A constraint programming approach to the stable marriage problem, in: Proceedings of CP 2001: the 7th International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, vol. 2239, Springer-Verlag, 2001, pp. 225–239.
[28] G. Gottlob, L. Leone, F. Scarcello, A comparison of structural CSP decomposition methods, Artificial Intelligence 124 (2000) 243–282.
[29] G. Gottlob, L. Leone, F. Scarcello, Hypertree decompositions: A survey, in: Proceedings 26th International Symposium on Mathematical Foundations of Computer Science, MFCS'01, in: Lecture Notes in Computer Science, vol. 2136, Springer-Verlag, 2001, pp. 37–57.
[30] G. Gottlob, L. Leone, F. Scarcello, Hypertree decomposition and tractable queries, Journal of Computer and System Sciences 64 (3) (2002) 579–627.
[31] M.J. Green, New methods for the tractability of constraint satisfaction problems, PhD thesis, University of London, Department of Computer Science, Royal Holloway, Egham, Surrey, UK, June 2005.
[32] M.J. Green, D.A. Cohen, Tractability by approximating constraint languages, in: Principles and Practice of Constraint Programming—CP 2003, in: Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 392–406.
[33] M. Gyssens, P.G. Jeavons, D.A. Cohen, Decomposing constraint satisfaction problems using database techniques, Artificial Intelligence 66 (1) (1994) 57–89.
[34] P.G. Jeavons, On the algebraic structure of combinatorial problems, Theoretical Computer Science 200 (1998) 185–204.
[35] P.G. Jeavons, D.A. Cohen, An algebraic characterization of tractable constraints, in: Computing and Combinatorics. First International Conference COCOON'95, Xi'an, China, August 1995, in: Lecture Notes in Computer Science, vol. 959, Springer-Verlag, 1995, pp. 633–642.
[36] P.G. Jeavons, M.C. Cooper, Tractable constraints on ordered domains, Artificial Intelligence 79 (2) (1995) 327–339.

[37] P.G. Jeavons, D.A. Cohen, M. Gyssens, Closure properties of constraints, Journal of the ACM 44 (1997) 527–548.
[38] P.G. Jeavons, D.A. Cohen, M.C. Cooper, Constraints, consistency and closure, Artificial Intelligence 101 (1–2) (1998) 251–265.
[39] P. Jégou, Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems, in: AAAI, 1993, pp. 731–736.
[40] H.A. Kautz, B. Selman, Planning as satisfiability, in: Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92), 1992, pp. 359–363.
[41] J. Larrosa, E. Morancho, D. Niso, On the practical use of variable elimination in constraint optimization problems: ?still-life? as a case study, Journal of Artificial Intelligence Research 23 (2005) 421–440.
[42] H.R. Lewis, Renaming a set of clauses as a Horn set, JACM 25 (1) (1978) 134–135.
[43] A.K. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1977) 99–118.
[44] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132.
[45] D.J. Rose, Triangulated graphs and the elimination process, Journal of Mathematical Analysis and Applications 32 (1970) 597–609.
[46] F. Rossi, P. van Beek, T. Walsh (Eds.), The Handbook of Constraint Programming, Elsevier, 2006 (Chapter 8. Foundations of Artificial Intelligence).
[47] T.J. Schaefer, The complexity of satisfiability problems, in: Proceedings 10th ACM Symposium on Theory of Computing, STOC'78, 1978, pp. 216–226.
[48] P. van Beek, Reasoning about qualitative temporal information, Artificial Intelligence 58 (1992) 297–326.
[49] P. van Beek, R. Dechter, On the minimality and decomposability of row-convex constraint networks, Journal of the ACM 42 (1995) 543–561.
[50] M.R.C. van Dongen, From local closure properties to a global closure property, Personal communication, 2002. The source of the SMP example.