Variable Elimination in Binary CSP via Forbidden Patterns *

David A. Cohen[1], Martin C. Cooper[2], Guillaume Escamocher[2] and Stanislav Živný[3]
[1]Royal Holloway, University of London, UK, d.cohen@rhul.ac.uk
[2]IRIT, University of Toulouse, France, {cooper|escamocher}@irit.fr
[3]University of Warwick, UK, S.Zivny@warwick.ac.uk

Abstract

A variable elimination rule allows the polynomial-time identification of certain variables whose elimination does not affect the satisfiability of an instance. Variable elimination in the constraint satisfaction problem (CSP) can be used in preprocessing or during search to reduce search space size. We show that there are essentially just four variable elimination rules defined by forbidding generic sub-instances, known as irreducible patterns, in arc-consistent CSP instances. One of these rules is the Broken Triangle Property, whereas the other three are novel.

keywords: constraint satisfaction, tractability, arc consistency, forbidden pattern.

1 Introduction

Constraint satisfaction has proved to be a useful modelling tool in a variety of contexts, such as scheduling, timetabling, planning, bio-informatics and computer vision [Dechter, 2003; Rossi *et al.*, 2006].

In this model we have a number of variables, each of which can take values from its particular finite domain. Certain sets of the variables are constrained in that their simultaneous assignments of values is limited. We are required to assign values to all variables so that every constraint is satisfied. Complete solution algorithms for constraint satisfaction are not polynomial time unless P=NP, since the graph colouring problem, which is NP-complete, can be reduced to constraint satisfaction [Dechter, 2003]. Hence we need to find ways to reduce the search space.

Search algorithms for constraint problems usually proceed by transforming the instance into a set of subproblems, for example, by selecting a variable and assigning to it successively each value from its domain. This naive backtracking approach is recursive and explores the search tree of partial assignments in a depth first manner. Even though the algorithm can take exponential time it is often effective in practice. So, we would like to improve its efficiency.

*Martin Cooper and Guillaume Escamocher are supported by ANR Project ANR-10-BLAN-0210.

There are many ways to improve naive backtracking by pruning the search space in ways that cannot remove solutions. This is done by avoiding searching exhaustively in all generated subproblems when certain kinds of discovered obstruction to solution exists. Such techniques include Back marking, Back jumping, CDBJ etc. [Prosser, 1993], As well as these techniques it is also possible to maintain local consistency by propagating the consequences of early decisions or of the discovered structure. Of these techniques the most common is to maintain so called generalised arc-consistency (GAC). This technique identifies values for variables that cannot possibly form part of a solution.

Of course, savings can be made if we are able to eliminate variables from a sub-problem. Since backtracking is of exponential time complexity the elimination of variables to reduce instance size can in the best case reduce search time by an exponential factor. To maintain the soundness of search we require that such eliminations do not change the satisfiability of the instance.

1.1 Simplification by variable elimination

Suppose that x is a variable of an instance I and that, whenever there is some valid assignment to all variables except x, there is a solution to the whole instance; in this case, we can safely remove variable x from I. The question we address in this paper is how to identify such variables?

Variable elimination has been considered before in the literature. It has been observed that the (local) Broken Triangle Property (lBTP) [Cooper *et al.*, 2010], if it holds at some variable, allows us to eliminate that variable. The closure of a binary CSP instance under the elimination of all variables that satisfy the lBTP is unique and can be found in $O(ned^3)$ time, where n is the number of variables, e the number of constraints and d the maximum domain size, which may well prove effective when compared to the exponential cost of backtracking. As a concrete example, in a path consistent binary CSP instance, all Boolean variables can be eliminated since they necessarily satisfy the lBTP. The more general local min-of-max extendable property (lMME) allows us to eliminate more variables than the lBTP, but requires the identification of a particular domain order. Unfortunately, this domain order is NP-hard to discover [Cooper *et al.*, 2010] for unbounded domain size, and so the lMME is less likely to be effective in practice.

An alternative to simple variable elimination is used in Bucket Elimination [Larrosa and Dechter, 2003]. In this algorithm variables are not simply eliminated. Instead they are replaced by constraints on their neighbourhood. These new constraints precisely capture the rules which allow an assignment to the neighbourhood of a variable to be extended. Such an approach may generate high order constraints, which are exponentially hard to process and to store. The arity can be bounded by the induced treewidth of the instance, but this still limits the applicability of Bucket Elimination.

1.2 Our contribution

In this paper we characterise those local conditions under which we can eliminate variables in binary CSPs without the need to add compensating constraints. By local conditions we mean here configurations of variables, values and constraints which *do not* occur. That is, we will identify (local) obstructions to variable elimination. We will call such constructions variable elimination patterns.

Surprisingly we find that there are precisely four essentially different local patterns whose absence permits variable elimination. Searching for these local patterns takes polynomial time and need only be done during the pre-processing stage, before search. Any discovered obstructions to elimination can be effectively monitored during the subsequent search using techniques analogous to watched literals [Gent *et al.*, 2006]. Whenever a variable no longer participates in any obstruction patterns it can safely be eliminated.

2 Definitions

When certain kinds of local obstructions are not present in a binary CSP instance, variable elimination is possible. Such obstructions are called quantified patterns. A pattern can be seen as a generalisation of the concept of a constraint satisfaction instance that leaves the consistency of some assignments to pairs of variables undefined.

Definition 2.1 *A* pattern *is a four-tuple $\langle V, D, A, \mathrm{cpt}\rangle$ where:*

- *V is a finite set of* variables*;*
- *D is a finite set of* values*;*
- *$A \subseteq V \times D$ is the set of possible* assignments*; The* domain *of $v \in V$ is its set $\mathcal{D}(v)$ of possible values: $\mathcal{D}(v) = \{d \in D \mid \langle v, d\rangle \in A\}$; and*
- *cpt is a partial* compatibility function *from the set of unordered pairs of assignments $\{\{\langle v, a\rangle, \langle w, b\rangle\} \mid v \neq w\}$ to $\{T, F\}$; if $\mathrm{cpt}(\{\langle v, a\rangle, \langle w, b\rangle\}) = T$ (resp., F) we say that $\langle v, a\rangle$ and $\langle w, b\rangle$ are* compatible *(resp.,* incompatible*). We write dom(cpt) to represent the set of pairs of assignments on which cpt is defined.*

A quantified pattern *is a pattern P with a distinguished variable, $\overline{v}(P)$ and a subset of existential values $e(P) \subseteq \mathcal{D}(\overline{v}(P))$.*

A flat quantified pattern *is a quantified pattern for which $e(P)$ is empty. An* existential pattern *is a quantified pattern P for which $e(P)$ is non-empty.*

When the context variable v is clear we use the value d to denote the assignment $\langle v, d\rangle$ to v. We will often simplify notation by writing $\mathrm{cpt}(p, q)$ for $\mathrm{cpt}(\{p, q\})$. We will also use the terminology of graph theory, since a pattern can be viewed as a labelled graph: if $\mathrm{cpt}(p, q) = T$ (resp., F), then we say that there is a *compatibility* (resp., *incompatibility*) *edge* between p and q.

We will use a simple figurative drawing for patterns. Each variable will be drawn as an oval containing dots for each of its possible assignments. Pairs in the domain of the function cpt will be represented by lines between values: solid lines for compatibility and dashed lines for incompatibility. The distinguished variable ($\overline{v}(P)$) and any existential values in $e(P)$ will be indicated by an $\exists$ symbol. Examples of patterns are shown in Figure 1.

We are never interested in the names of variables nor the names of the domain values in patterns. So we define the following equivalence.

Definition 2.2 *Two patterns P and Q are* equivalent *if they are isomorphic, i.e. if they are identical except for possible injective renamings of variables and assignments which preserve $\mathcal{D}$, cpt, $\overline{v}$ and e.*

We can refine patterns to give a definition of a (binary) CSP instance.

Definition 2.3 *A* binary CSP instance *P is a pattern $\langle V, D, A, \mathrm{cpt}\rangle$ where cpt is a total function, i.e. the domain of cpt is precisely $\{\{\langle v, a\rangle, \langle w, b\rangle\} \mid v \neq w\}$.*

- *The* relation *$R_{v,w} \subseteq D^2$ on $\langle v, w\rangle$ is $\{\langle a, b\rangle \mid \mathrm{cpt}(\{\langle v, a\rangle, \langle w, b\rangle\}) = T\}$.*
- *A* partial solution *to P on $X \subseteq V$ is a mapping $s : X \to D$ where, for all $v \neq w \in X$ we have $\langle s(v), s(w)\rangle \in R_{v,w}$.*
- *A* solution *to P is a partial solution on V.*

2.1 Variable elimination

In this paper we are concerned with variable elimination characterised by forbidden patterns. We define what this means in this section.

Definition 2.4 *We say that a variable x can be* eliminated *in the CSP instance $\langle V, D, A, \mathrm{cpt}\rangle$ if, whenever there is a partial solution on $V \setminus \{x\}$ there is a solution.*

In practice, when solving CSP instances we prune the domains of variables in such a way as to maintain all solutions.

Definition 2.5 *Let $P = \langle V, D, A, \mathrm{cpt}\rangle$ be a CSP instance. An assignment $\langle v, a\rangle \in A$ to variable v is called* arc consistent *if, for all variables $w \neq v$ there is some assignment $\langle w, b\rangle$ compatible with $\langle v, a\rangle$.*

The CSP instance $\langle V, D, A, \mathrm{cpt}\rangle$ is called arc consistent *if every assignment in A is arc consistent.*

Assignments that are not arc-consistent cannot be part of a solution so can safely be removed. There are many quadratic-time algorithms for establishing arc consistency which repeatedly remove such values [Bessière *et al.*, 2005]. Hence, for the remainder of this paper we will assume that all CSP instances are arc-consistent.

In order to use (the absence of) patterns for variable elimination we need to define what we mean when we say that a

quantified pattern occurs at variable x of a CSP instance. We define occurrence in terms of reductions on patterns. The definitions of occurrence and reduction between quantified patterns extend definitions previously given for non-quantified patterns [Cooper and Escamocher, 2012].

Definition 2.6 *Let $P = \langle V, D, A, \mathrm{cpt}\rangle$ be any pattern.*

- *We say that a pattern $P' = \langle V', D', A', \mathrm{cpt}'\rangle$ is a* sub-pattern *of P if $V' \subseteq V, A' \subseteq A$ and $\mathrm{cpt}' = \mathrm{cpt} \restriction_{A'}$.*
  If, furthermore, P' is quantified then we require that P is quantified and that $\overline{v}(P') = \overline{v}(P)$ and $e(P') \subseteq e(P)$.
- *Values $a, b \in \mathcal{D}(v)$ are* mergeable *in a pattern if there is no assignment $p \in A$ for which $\mathrm{cpt}(\{\langle v, a\rangle, p\})$, $\mathrm{cpt}(\{\langle v, b\rangle, p\})$ are both defined and $\mathrm{cpt}(\{\langle v, a\rangle, p\}) \neq \mathrm{cpt}(\{\langle v, b\rangle, p\})$. In a quantified pattern we also require that $a \in e(P)$ if and only if $b \in e(P)$.*
  When $a, b \in \mathcal{D}(v)$ are mergeable we define the merged reduction $\langle V, D, A \setminus \{\langle v, a\rangle\}, \mathrm{cpt}'\rangle$ by reducing the set of assignments. This naturally makes:
  $$\mathrm{cpt}'(p, q) = \begin{cases} \mathrm{cpt}(\langle v, a\rangle, q) & \text{if } p = \langle v, b\rangle \text{ and} \\ & \mathrm{cpt}(\langle v, b\rangle, q) \text{ undefined,} \\ \mathrm{cpt}(p, q) & \text{otherwise.} \end{cases}$$
- *A* dangling assignment *p of P is any assignment not in $\{\langle \overline{v}(P), a\rangle \mid a \in e(P)\}$ for which there is at most one assignment q for which $\mathrm{cpt}(p, q)$ is defined, and furthermore (if defined) $\mathrm{cpt}(p, q) = T$. For any dangling assignment p, we define the dangling reduction $\langle V, D, A', \mathrm{cpt} \restriction_{A'}\rangle$ where $A' = A \setminus \{p\}$.*
- *An* irreducible pattern *is one on which no merged or dangling reduction can be performed.*

Now we want to define when a quantified pattern occurs at a variable in a CSP instance, in order to characterise those patterns whose *non-occurrence* allows this particular variable to be eliminated. We define the slightly more general notion of occurrence of a pattern in another pattern. Recall that a CSP instance corresponds to the special case of a pattern whose compatibility function is total. Essentially we want to say that pattern P occurs in pattern Q if P is homomorphic to a sub-pattern of Q via an injective renaming of variables and a (possibly non-injective) renaming of assignments [Cohen *et al.*, 2012]. However, we find it simpler to define occurrence using the notions of sub-pattern, reduction and equivalence. We first make the observation that dangling assignments in a pattern provide no useful information since we assume that all CSP instances are arc consistent, which explains why dangling assignments can be eliminated from patterns.

We can then define occurrence in terms of reduced patterns.

Definition 2.7 *We say that a quantified pattern P* occurs *in a pattern Q (and that Q* contains *P) if some reduction of P is equivalent to a sub-pattern of Q.*

If Q is a CSP instance, then P occurs at variable x of Q *if some reduction of P is equivalent to a sub-pattern of Q and x is the variable of the sub-pattern of Q corresponding to $\overline{v}(P)$.*

We say that P occurs at variable x of Q with value mapping $m : e(P) \to \mathcal{D}(x)$ *if the values of variable x corresponding to each $d \in e(P)$ are given by the mapping m.*

The definition of a variable elimination pattern is defined in terms of occurrence.

Definition 2.8 *A quantified pattern is a* VE pattern *if, whenever the pattern does not occur at a variable x in an arc-consistent CSP instance I, for at least one injective value mapping, x can be eliminated in I.*

Existential patterns may allow more variables to be eliminated than flat patterns. For example, as we will show later, the patterns snake and ∃snake shown in Figure 1 are both VE patterns, but the latter allows more variables to be eliminated since we only require that it does not occur on a single value in the domain of the variable to be eliminated.

Suppose that we can assign value 0 to a subset S of the variables of an instance, independently of assignments to any other variables. Furthermore suppose that, within S, 0 is only compatible with 0. The VE pattern ∃invsubBTP, shown in Figure 1, allows us to eliminate all variables in S, without having to explicitly search for S. This is because the pattern does not occur for the mapping $a \mapsto 0$. The flat variant (invsubBTP) would not allow these eliminations.

We conclude this section with the simple observation that VE patterns define tractable classes. It takes polynomial time to establish arc consistency and to detect (by exhaustive search) the non-occurrence of a VE pattern. Hence it takes polynomial time to identify arc-consistent CSP instances for which all variables can be eliminated one by one by a VE pattern P. Such instances are solvable in a greedy fashion.

Hence we are able to significantly extend the list of known tractable classes defined by forbidden patterns since among known tractable patterns, namely BTP [Cooper *et al.*, 2010], 2-constraint patterns [Cooper and Escamocher, 2012], pivots [Cohen *et al.*, 2012] and JWP [Cooper and Živný, 2012], *only* BTP (and its sub-patterns) allow variable elimination.

Indeed, a general hybrid tractable class can be defined: the set of binary CSP instances which fall in some known tractable class after we have performed all variable eliminations by the rules given in this paper.

3 Variable elimination by forbidden patterns

In this paper we characterise irreducible VE patterns. There are essentially just four: the patterns BTP, ∃subBTP, ∃invsubBTP and ∃snake, shown in Figure 1, together with their irreducible sub-patterns. We begin by showing that each of these four patterns allows variable elimination.

Theorem 3.1 *The patterns BTP, ∃subBTP, ∃invsubBTP and ∃snake are VE patterns.*

Proof: Since it is known that BTP is a VE pattern [Cooper *et al.*, 2010], we only need to prove the result for the three existential patterns: ∃subBTP, ∃invsubBTP and ∃snake.

Every two variable arc-consistent CSP instance allows either variable to be eliminated. So we only have to prove that these patterns allow variable elimination in CSP instances with at least three variables.

We first set up some general machinery which will be used in each of the three cases. Consider an arc-consistent CSP instance $I = \langle X, D, A, \mathrm{cpt}\rangle$ and let s be a partial solution on $X \setminus \{x\}$.

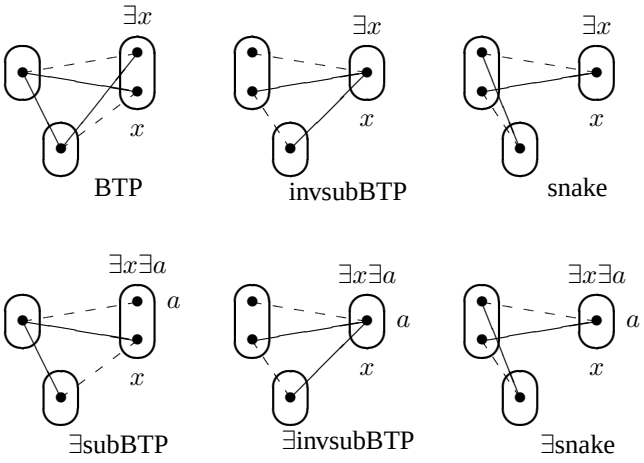

Figure 1: Variable elimination patterns.

Fix some assignment $\langle x, d\rangle$, and let:

$$Y = \{y \in X \setminus \{x\} \mid \mathrm{cpt}(\langle y, s(y)\rangle, \langle x, d\rangle) = T\},$$
$$\overline{Y} = \{z \in X \setminus \{x\} \mid \mathrm{cpt}(\langle z, s(z)\rangle, \langle x, d\rangle) = F\}.$$

For all $y, z \in X$, since s is a partial solution, we know $\mathrm{cpt}(\langle y, s(y)\rangle, \langle z, s(z)\rangle) = T$. Thus, if $X = Y \cup \{x\}$ then we can extend s to a solution to I by choosing value d for variable x. So, in this case x could be eliminated. So we assume from now on that $\overline{Y} \neq \emptyset$.

By arc consistency, for all $z \in \overline{Y}$, there is some $\langle z, t(z)\rangle \in A$ such that $\mathrm{cpt}(\langle z, t(z)\rangle, \langle x, d\rangle) = T$.

We now prove the result for each pattern in turn.

Suppose that ∃subBTP does not occur at x in I for the mapping $a \mapsto d$. Consider any $y \in \overline{Y}$. By arc consistency, $\exists b \in \mathcal{D}(x)$ such that $\mathrm{cpt}(\langle y, s(y)\rangle, \langle x, b\rangle) = T$. Since the pattern ∃subBTP does not occur, and in particular on the set of assignments $\{\langle y, s(y)\rangle, \langle z, s(z)\rangle, \langle x, d\rangle, \langle x, b\rangle\}$, we can deduce that, for every variable $z \in X$ different from both x and y, $\mathrm{cpt}(\langle z, s(z)\rangle, \langle x, b\rangle) = T$. Hence, we can extend s to a solution to I by choosing $s(x) = b$. So, in any case x can be eliminated and ∃subBTP is indeed a VE pattern.

Now instead, suppose ∃invsubBTP does not occur at x in I for the mapping $a \mapsto d$. Since the pattern ∃invsubBTP does not occur, if both y and z belong to $\overline{Y}$ then $\mathrm{cpt}(\langle y, t(y)\rangle, \langle z, t(z)\rangle) = T$ otherwise the pattern would occur on the assignments $\{\langle y, s(y)\rangle, \langle y, t(y)\rangle, \langle z, t(z)\rangle, \langle x, d\rangle\}$. Also, if $y \in Y$, $z \in \overline{Y}$, then $\mathrm{cpt}(\langle y, s(y)\rangle, \langle z, t(z)\rangle) = T$ otherwise the pattern would occur on $\{\langle z, s(z)\rangle, \langle z, t(z)\rangle, \langle y, s(y)\rangle, \langle x, d\rangle\}$.

So, in this case we have a solution s' to I, where

$$s'(v) = \begin{cases} d & \text{if } v = x, \\ s(v) & \text{if } v \in Y, \\ t(v) & \text{otherwise.} \end{cases}$$

So ∃invsubBTP is indeed a VE pattern.

For the final pattern, suppose that ∃snake does not occur at x in I for the mapping $a \mapsto d$. If $y \in Y$, $z \in \overline{Y}$, since the pattern ∃snake does not occur, we can deduce that $\mathrm{cpt}(\langle y, s(y)\rangle, \langle z, t(z)\rangle) = T$ otherwise the pattern would occur on the assignments $\{\langle z, s(z)\rangle, \langle z, t(z)\rangle, \langle y, s(y)\rangle, \langle x, d\rangle\}$. If both y and z both belong to $\overline{Y}$, then we can deduce first that $cpt(\langle y, s(y)\rangle, \langle z, t(z)\rangle) = T$ (as in the previous case) and then, as a consequence, that $cpt(\langle y, t(y)\rangle, \langle z, t(z)\rangle) = T$ (otherwise the pattern would occur on $\{\langle y, s(y)\rangle, \langle y, t(y)\rangle, \langle z, t(z)\rangle, \langle x, d\rangle\}$).

So, again in this case we have a solution s' to I, where s' is defined as above. So ∃snake is also a VE pattern. ∎

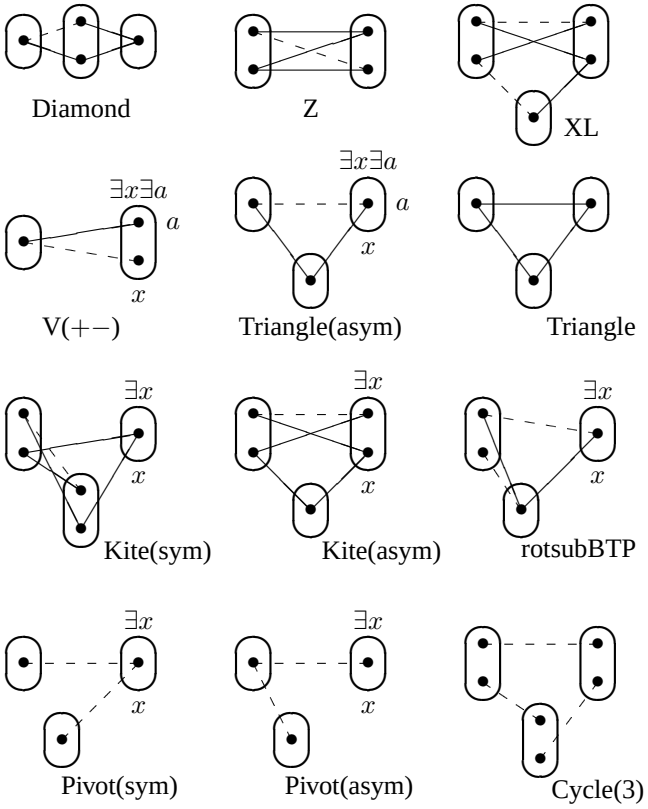

Figure 2: Patterns which do not allow variable elimination.

4 Characterisation of quantified VE patterns

Our aim is to precisely characterise all irreducible patterns which allow variable elimination in an arc-consistent binary CSP instance. We begin by identifying many patterns, including all those shown in Figure 2, which are not variable elimination patterns.

Lemma 4.1 *None of the following patterns allow variable elimination in arc-consistent binary CSP instances: any pattern on more than three variables, any pattern with three distinct values for the same variable, any pattern with two non-mergeable incompatibility edges in the same constraint, Diamond, Z, XL, V(+−), Triangle(asym), Triangle, Kite(sym), Kite(asym), rotsubBTP, Pivot(asym), Pivot(sym), Cycle(3).*

Proof: For each pattern we exhibit a binary arc-consistent CSP instance that:

- does not contain the given pattern P at a variable x;

- has a partial solution on all the variables except x;
- has no solution.

By definition, any such instance is enough to prove that a pattern is not a VE pattern.

- For any pattern P which is either Diamond, Z, XL, or Triangle, or has at least four variables, or has three distinct values for the same variable.

  Let I_3^{2COL} be the CSP instance (corresponding to 2-colouring on 3 variables) with three Boolean variables, where the constraint between any two variables forces them to take different values.

  This instance has partial solutions on any two variables, but has no solution, and does not contain P.

- For V($+-$) and Triangle(asym).

  Let $I_4^{\exists}$ be the instance on four variables x_1, x_2, x_3 and x, where the domains of x_1, x_2 and x_3 are all $\{0,1,2\}$ and the domain of x is $\{0,1,2,3\}$. Each pair of variables in $\{x_1, x_2, x_3\}$ must take values in $\{\langle 0,0\rangle, \langle 1,2\rangle, \langle 2,1\rangle\}$. There are three further constraints: for $i = 1,2,3$, we have that $(x_i > 0) \vee (x = i)$.

  $I_4^{\exists}$ contains neither V($+-$) nor Triangle(asym) at variable x for the value mapping $m(a) = 0$.

- For Kite(sym).

  Let I_4 be the CSP instance on four variables x_1, x_2, x_3, x where x_1, x_2 and x_3 are Boolean and $\mathcal{D}(x) = \{1,2,3\}$, with the following constraints: $x_1 \vee x_2$, $x_1 \vee x_3$, $x_2 \vee x_3$, $x_i \Leftrightarrow (x = i)$ $(i = 1,2,3)$.

- For Kite(asym).

  Let I_4^{ZOA} be the CSP instance on the four variables x_1, x_2, x_3, x where x_1, x_2 and x_3 are Boolean and $\mathcal{D}(x) = \{1,2,3\}$, with the following constraints: $x_1, x_2, x_3, x \in \{1,2,3\}$, $x_1 = x_2$, $x_1 = x_3$, $x_2 = x_3$, $(x_1 = 1) \vee (x = 1)$, $(x_2 = 2) \vee (x = 2)$, $(x_3 = 3) \vee (x = 3)$.

- For rotsubBTP.

  Define the three binary relations:

$$\begin{aligned} R &= \{\langle 0,0\rangle, \langle 1,2\rangle, \langle 2,1\rangle\}, \\ R_0 &= \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,1\rangle\}, \\ R_1 &= \{\langle 0,1\rangle, \langle 1,0\rangle, \langle 2,0\rangle\}. \end{aligned}$$

  Let I_7 be the CSP instance on the seven variables $x_1, \ldots, x_6, x$ where $\mathcal{D}(x_i) = \{0,1,2\}$, for $i = 1, \ldots, 6$, and $\mathcal{D}(x) = \{0,1\}$, with the following constraints:

  For $(1 \le i < j \le 3)$ and $4 \le i < j \le 6)$, $\langle x_i, x_j\rangle$ must take values in R.

  For $(1 \le i \le 3)$, $\langle x_i, x\rangle$ must take values in R_0.

  For $(4 \le i \le 6)$, $\langle x_i, x\rangle$ must take values in R_1 .

- For the pattern Pivot(sym).

  Let I_4^{SAT} be the 2SAT instance on four Boolean variables x_1, x_2, x_3, x with the following constraints: $x_1 \equiv x_2$, $x_1 \equiv x_3$, $x_2 \vee x_3$, $\overline{x_2} \vee x$, $\overline{x_3} \vee \overline{x}$.

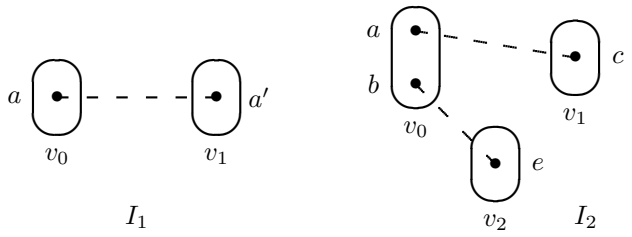

Figure 3: The possible negative skeletons of VE patterns.

- For Cycle(3) or Pivot(asym), or any pattern with two non-mergeable incompatibility edges in the same constraint.

  Let I_6^{SAT} be the 2SAT instance on six Boolean variables $x_1, x_2, x_3, x_4, x_5, x$ with the following constraints: $\overline{x_1} \vee \overline{x_2}$, $\overline{x_1} \vee \overline{x_4}$, $x_1 \vee \overline{x_3}$, $x_1 \vee \overline{x_5}$, $x_2 \vee \overline{x}$, $x_4 \vee x$, $x_3 \vee \overline{x}$, $x_5 \vee x$.

■

The following lemma is then key to proving that we have identified all possible quantified VE patterns.

Lemma 4.2 *The only flat quantified irreducible patterns that do not contain any of the patterns listed in Lemma 4.1 are contained in BTP, invsubBTP or snake (shown in Figure 1).*

Proof: Consider a flat quantified irreducible pattern $P = \langle X, D, A, \mathrm{cpt}\rangle$ that does not contain any of the patterns listed in Lemma 4.1. Thus P has at most three variables each with domain size less than three.

We consider first the case of a 2-variable pattern P. By Lemma 4.1, P does not have two distinct non-mergeable incompatibility edges and does not contain Z. Since P is irreducible and hence does not have any dangling assignment, we can deduce by exhausting over all possibilities that P does not have any compatibility edge and a single incompatibility edge. Hence P is contained in BTP. We can therefore assume that P has exactly three variables.

Now consider the negative sub-pattern $P^- = \langle X, D, A, \mathrm{neg}\rangle$ where the compatibility function neg is cpt with its domain reduced to the incompatible pairs of assignments of P.

Any irreducible pattern on three variables that does not contain an incompatible pair of assignments must contain Triangle. Moreover, if any assignment is incompatible with two other assignments then P must contain either Pivot(sym) or Pivot(asym). Now, since P does not contain Cycle(3), it follows that P^- is I_1 or I_2, as shown in Figure 3.

We first consider the latter case. Without loss of generality, we assume that b is compatible with c, to avoid a and b being mergeable.

Since the domains have at most two elements, we begin by assuming that $\mathcal{D}(v_1) = \{c, d\}$ and $\mathcal{D}(v_2) = \{e, f\}$. In this case a and d must be compatible to avoid d and c being mergeable. Also b and f must be compatible to stop e and f being mergeable. Moreover, d and e cannot be compatible since otherwise XL occurs in P. Furthermore, d and f cannot be compatible since, whichever variable is chosen for $e(P)$,

either Kite(sym) or Kyte(asym) occurs in P. It follows that d can be removed as it is a dangling assignment.

Now we begin again. As before, to avoid merging e and f or Diamond occurring in P, we have that f is compatible with b and not compatible with a. So f must be compatible with c to avoid f being a dangling assignment. In this case P contains Triangle.

Finally, we have $\mathcal{D}(v_1) = \{c\}$ and $\mathcal{D}(v_2) = \{e\}$. Suppose that there is a compatibility edge between c and e. If the distinguished variable $\overline{v}(P)$ is v_0 then, whether or not there is a compatibility edge between a and e, the pattern is contained in BTP. If $\overline{v}(P) = v_1$ and there is no compatibility edge between a and e, then the pattern is contained in invsubBTP. If $\overline{v}(P) = v_1$ and there is a compatibility edge between a and e, then the pattern contains rotsubBTP. If $\overline{v}(P) = v_2$, then the pattern contains rotsubBTP. Since we have covered all cases in which there is a compatibility edge between c and e, we assume that there is no edge between c and e.

Whether or not there is an incompatibility edge between a and e, the pattern is contained in BTP if $\overline{v}(P) = v_0$, and the pattern is contained in snake if $\overline{v}(P)$ is either v_1 or v_2.

The final case to consider is when P is a 3-variable pattern with $P^- = I_1$. Any two assignments for the third variable v_2 could be merged, so we can assume its domain is a singleton which we denote by $\{a''\}$. Since P is irreducible, does not contain Diamond, Triangle, Kite(sym) or Kite(asym), we can deduce that the only compatible pairs of assignments include a''. In fact, both $\{a, a''\}$ and $\{a', a''\}$ must be compatible since P is irreducible. But then P is contained in BTP if $\overline{v}(P)$ is either v_0 or v_1, and is contained in invsubBTP if $\overline{v}(P) = v_2$. ∎

We need the following technical lemma which shortens several proofs.

Lemma 4.3 *If a pattern P occurs in a VE pattern Q, then P is also a VE pattern.*

Proof: Suppose that P occurs in the VE pattern Q. By transitivity of the occurrence relation, if Q occurs in a binary CSP instance I (at variable x), then so does P. It follows that if P does not occur (at variable x) in an arc consistent binary CSP instance I, then neither does Q and variable elimination is possible. ∎

According to Definition 2.7, a flat pattern P is a sub-pattern of any existential version Q of P (and hence P occurs in Q). We state this special case of Lemma 4.3 as a corollary.

Corollary 4.4 *Let Q be an existential VE pattern. If P is the flattened version of pattern Q, corresponding to $e(P) = \emptyset$, then P is also a VE pattern.*

The following theorem is a direct consequence of Theorem 3.1 and Corollary 4.4 together with Lemma 4.1, Lemma 4.2 and Lemma 4.3.

Theorem 4.5 *The irreducible flat quantified patterns allowing variable elimination in arc-consistent binary CSP instances are BTP, invsubBTP or snake (and their irreducible sub-patterns).*

We are now able to provide the characterisation for existential patterns after a little extra work.

Theorem 4.6 *The only irreducible existential patterns which allow variable elimination in arc-consistent binary CSP instances are ∃subBTP, ∃invsubBTP, ∃snake (and their irreducible sub-patterns).*

Proof: We know from Theorem 3.1 that ∃subBTP, ∃invsubBTP, ∃snake are VE patterns.

Theorem 4.5 and Corollary 4.4 show that when we flatten an existential VE pattern then the resulting flat quantified pattern is contained in BTP, invsubBTP or snake.

In the case of invsubBTP and snake, the existential versions of these patterns are VE patterns and so there is nothing left to prove. So we only need to consider quantified patterns which flatten into sub-patterns of BTP.

We firstly consider the case in which P is an existential version of a sub-pattern of BTP with $|e(P)| = 2$. In this case, the two assignments for the distinguished variable $\overline{v}(P)$ are mergeable unless P contains V(+−). Hence, either P is reducible or, by Lemma 4.1 and Lemma 4.3, P is not a VE pattern.

Let ∃BTP denote the existential version Q of BTP such that $|e(Q)| = 1$. By symmetry, ∃BTP is unique. The only remaining case is when P is an irreducible subpattern of ∃BTP with $|e(P)| = 1$. By a straightforward exhaustive case analysis, we find that, in this case, either P contains V(+−) or Triangle(asym) or P is a sub-pattern of ∃subBTP. The result then follows by Lemma 4.1 and Lemma 4.3. ∎

Combining Theorem 4.5 and Theorem 4.6, we obtain the characterisation of irreducible quantified VE patterns.

Theorem 4.7 *The only irreducible quantified patterns which allow variable elimination in arc-consistent binary CSP instances are BTP, ∃subBTP, ∃invsubBTP, ∃snake (and their irreducible sub-patterns).*

5 Conclusion

This paper has introduced the notion of quantified pattern which has allowed us to uncover sound variable elimination rules in binary CSPs. We have identified all irreducible quantified patterns whose absence allows variable elimination. As a consequence, we have also identified novel tractable classes of binary CSPs.

There are several interesting directions for further research. Are there interesting patterns allowing variable elimination after (1,r) consistency for some $r > 2$? Can we find a characterisation result for all possibly-reducible VE patterns or for the even more general case of all *sets* of patterns allowing variable elimination? Can we generalise any of the VE patterns described in this paper to general-arity CSP instances? Are there quantified patterns which are not VE patterns but which nonetheless define tractable classes? Do the VE patterns introduced in this paper generalise to other versions of constraint satisfaction, such as the QCSP (as is the case for the tractable class defined by BTP [Gao *et al.*, 2011]) or the WCSP (as is the case for tractable class defined by JWP [Cooper and Živný, 2012])?

References

[Bessière *et al.*, 2005] Christian Bessière, Jean-Charles Régin, Roland H. C. Yap, and Yuanlin Zhang. An optimal coarse-grained arc consistency algorithm. *Artificial Intelligence*, 165(2):165–185, 2005.

[Cohen *et al.*, 2012] David A. Cohen, Martin C. Cooper, Páidí Creed, Dániel Marx, and András Z. Salamon. The tractability of CSP classes defined by forbidden patterns. *J. Artif. Intell. Res. (JAIR)*, 45:47–78, 2012.

[Cooper and Escamocher, 2012] Martin C. Cooper and Guillaume Escamocher. A dichotomy for 2-constraint forbidden csp patterns. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

[Cooper and Živný, 2012] Martin C. Cooper and Stanislav Živný. Tractable triangles and cross-free convexity in discrete optimisation. *J. Artif. Intell. Res. (JAIR)*, 44:455–490, 2012.

[Cooper *et al.*, 2010] Martin C. Cooper, Peter G. Jeavons, and András Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artif. Intell.*, 174(9-10):570–584, 2010.

[Dechter, 2003] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205, 2003.

[Gao *et al.*, 2011] Jian Gao, Minghao Yin, and Junping Zhou. Hybrid tractable classes of binary quantified constraint satisfaction problems. In *AAAI*, 2011.

[Gent *et al.*, 2006] Ian P. Gent, Christopher Jefferson, and Ian Miguel. Watched literals for constraint propagation in minion. In Frédéric Benhamou, editor, *CP*, volume 4204 of *Lecture Notes in Computer Science*, pages 182–197. Springer, 2006.

[Larrosa and Dechter, 2003] Javier Larrosa and Rina Dechter. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, 8(3):303–326, 2003.

[Prosser, 1993] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, November 1993.

[Rossi *et al.*, 2006] F. Rossi, P. van Beek, and T. Walsh, editors. *The Handbook of Constraint Programming*. Elsevier, 2006.