Variable Elimination in Binary CSP via Forbidden Patterns

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

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 (IBTP) [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 IBTP 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 IBTP. The more general local min-of-max extendable property (IMME) allows us to eliminate more variables than the IBTP, 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 IMME is less likely to be effective in practice.

\textsuperscript{*}Martin Cooper and Guillaume Escamocher are supported by ANR Project ANR-10-BLAN-0210.
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 \( \langle V, D, A, \text{cpt} \rangle \) 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 \( D(v) \) of possible values: \( D(v) = \{ d \in D \mid (v, d) \in A \} \); and
- \( \text{cpt} \) is a partial compatibility function from the set of unordered pairs of assignments \( \{(v, a), (w, b)\} \mid v \neq w\) to \( T, F \): if \( \text{cpt}(\{(v, a), (w, b)\}) = T \) (resp., \( F \)) we say that \( (v, a) \) and \( (w, b) \) are compatible (resp., incompatible). We write \( \text{dom}(\text{cpt}) \) to represent the set of pairs of assignments on which \( \text{cpt} \) is defined.

A quantified pattern is a pattern \( P \) with a distinguished variable, \( \tau(P) \), and a subset of existential values \( e(P) \subseteq D(\tau(P)) \).

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 \( (v, d) \) to \( v \). We will often simplify notation by writing \( \text{cpt}(p, q) \) for \( \text{cpt}(\{p, q\}) \). We will also use the terminology of graph theory, since a pattern can be viewed as a labelled graph: if \( \text{cpt}(p, q) = T \) (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 \( \text{cpt} \) will be represented by lines between values: solid lines for compatibility and dashed lines for incompatibility. The distinguished variable \( \tau(P) \) 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 \( D, \text{cpt}, \tau \) 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 \( \langle V, D, A, \text{cpt} \rangle \) where \( \text{cpt} \) is a total function, i.e. the domain of \( \text{cpt} \) is precisely \( \{(v, a), (w, b)\} \mid v \neq w\). The relation \( R_{v,w} \subseteq D^2 \) on \( \langle v, w \rangle \) is \( \{(a, b) \mid \text{cpt}(\{(v, a), (w, b)\}) = T\} \).

A partial solution to \( P \) on \( X \subseteq V \) is a mapping \( s : X \rightarrow D \) 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 \( \langle V, D, A, \text{cpt} \rangle \) 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 \( P = \langle V, D, A, \text{cpt} \rangle \) be a CSP instance. An assignment \( (v, a) \in A \) to variable \( v \) is called arc consistent if, for all variables \( w \neq v \) there is some assignment \( (w, b) \) compatible with \( (v, a) \).

The CSP instance \( \langle V, D, A, \text{cpt} \rangle \) 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 \( P = (V, D, A, \text{cpt}) \) be any pattern.

- We say that a pattern \( P' = (V', D', A', \text{cpt}') \) is a sub-pattern of \( P \) if \( V' \subseteq V \), \( A' \subseteq A \) and \( \text{cpt}' = \text{cpt} \, |_{A'} \).
- If, furthermore, \( P' \) is quantified then we require that \( P \) is quantified and that \( \overline{\pi}(P') = \overline{\pi}(P) \) and \( e(P') \subseteq e(P) \).
- Values \( a, b \in D(v) \) are mergeable in a pattern if there is no assignment \( p \in A \) for which \( \text{cpt}((\{v, a\}, p)) \) or \( \text{cpt}((\{v, b\}, p)) \) are both defined and \( \text{cpt}((\{v, a\}, p)) \neq \text{cpt}((\{v, b\}, p)) \). In a quantified pattern we also require that \( a \in e(P) \) if and only if \( b \in e(P) \).
- If \( a, b \in D(v) \) are mergeable we define the merged reduction \( (V, D, A \setminus \{v, a\}, \text{cpt}') \) by reducing the set of assignments. This naturally makes:
  \[
  \text{cpt}'(p, q) = \begin{cases} 
  \text{cpt}((v, a), q) & \text{if } p = (v, b) \\
  \text{cpt}((v, b), q) & \text{if } p = (v, q) \\
  \text{cpt}(p, q) & \text{else}
  \end{cases}
  \]

- A dangling assignment \( p \) of \( P \) is any assignment not in \( \{\overline{\pi}(P), a\} \mid a \in e(P)\} \) for which there is at most one assignment \( q \) for which \( \text{cpt}(p, q) \) is defined, and furthermore (if defined) \( \text{cpt}(p, q) = T \). For any dangling assignment \( p \), we define the dangling reduction \( (V, D, A', \text{cpt} \, |_{A'}) \) 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 \( \overline{\pi}(P) \).

We say that \( P \) occurs at variable \( x \) of \( Q \) with value mapping \( m : e(P) \to D(x) \) 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 \( \exists \text{snake} \) 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 \( \exists \text{invsubBTP} \) 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 (in-subBTP) 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 Živalj, 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, \( \exists \text{subBTP} \), \( \exists \text{invsubBTP} \) and \( \exists \text{snake} \), 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, \( \exists \text{subBTP} \), \( \exists \text{invsubBTP} \) and \( \exists \text{snake} \) 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: \( \exists \text{subBTP} \), \( \exists \text{invsubBTP} \) and \( \exists \text{snake} \).

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 \( I = (X, D, A, \text{cpt}) \) and let \( s \) be a partial solution on \( X \setminus \{x\} \).
\[ \exists y \text{ and } \text{cpt}(z) \]

Also, if \(y, z \in X\), since \(s\) is a partial solution, we know \(\text{cpt}(y, s(y)), (z, s(z)) = T\). 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 \(Y \neq \emptyset\).

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

We now prove the result for each pattern in turn.

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

Now instead, suppose \(\exists\text{invsubBTP}\) does not occur at \(x\) in \(I\) for the mapping \(a \rightarrow d\). Since the pattern \(\exists\text{invsubBTP}\) does not occur, if both \(y\) and \(z\) belong to \(Y\) then \(\text{cpt}(y, t(y)), (z, t(z)) = T\) otherwise the pattern would occur on the assignments \((y, s(y)), (y, t(y)), (z, t(z)), (x, d)\).

Also, if \(y \in Y\), \(z \in Y\), then \(\text{cpt}(y, s(y)), (z, t(z)) = T\) otherwise the pattern would occur on \((z, s(z)), (z, t(z)), (y, s(y)), (x, d)\).

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

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

So \(\exists\text{invsubBTP}\) is indeed a VE pattern.

For the final pattern, suppose that \(\exists\text{snake}\) does not occur at \(x\) in \(I\) for the mapping \(a \rightarrow d\). If \(y \in Y\), \(z \in Y\), since the pattern \(\exists\text{snake}\) does not occur, we can deduce that \(\text{cpt}(y, s(y)), (z, t(z)) = T\) otherwise the pattern would occur on the assignments \((z, s(z)), (z, t(z))\).

\[ \{y, s(y), (x, d)\} \] If both \(y\) and \(z\) both belong to \(Y\), then we can deduce first that \(\text{cpt}((y, s(y)), (z, t(z))) = T\) (as in the previous case) and then, as a consequence, that \(\text{cpt}((y, t(y)), (z, t(z))) = T\) (otherwise the pattern would occur on \((y, s(y)), (y, t(y)), (z, t(z)), (x, d)\)).

So, again in this case we have a solution \(s'\) to \(I\), where \(s'\) is defined as above. So \(\exists\text{snake}\) is also a VE pattern.

\[ \]
\[\text{Figure 3: The possible negative skeletons of VE patterns.}\]

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

- For Kite(\text{sym}).

- For \text{rotsubBTP}.

- For the pattern Pivot(\text{sym}).

\[
\begin{align*}
\text{Let } I_4^{\text{COL}} & \text{ be the CSP instance (corresponding to 2-colouring on 3 variables) with three Boolean variables, where the constraint between any two variables forces them to take different values. This instance has partial solutions on any two variables, but has no solution, and does not contain } P. \\
\text{For any pattern } P \text{ which is either Diamond, Z, XL, or Triangle, or has at least four variables, or has three distinct values for the same variable. Let } I_4^{\text{CO}} \text{ be the CSP instance (corresponding to 2-colouring on 3 variables) with three Boolean variables, where the constraint between any two variables forces them to take different values. This instance has partial solutions on any two variables, but has no solution, and does not contain } P. \\
\text{For } V(\rightarrow -) \text{ and Triangle(\text{sym}). For } Kite(\text{sym}). \\
\text{For } Kite(\text{asym}). \\
\text{For } \text{rotsubBTP}. \\
\text{For Cycle(3) or Pivot(\text{asym}), or any pattern with two non-mergeable incompatibility edges in the same constraint. Let } I_4^{\text{SAT}} \text{ be the 2SAT instance on six Boolean variables } x_1, x_2, x_3, x_4, x_5, x_6 \text{ with the following constraints: } x_1 \lor x_2, x_1 \lor x_3, x_2 \lor x_3, x_1 \lor x_3, x_2 \lor x_3, x_4 \lor x_5, x_3 \lor x_5, x_5 \lor x_6. \]

\[
\begin{align*}
\text{The following lemma is then key to proving that we have identified all possible quantified VE patterns.} \\
\text{Lemma 4.2} \quad \text{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).} \\
\text{Proof:} \quad \text{Consider a flat quantified irreducible pattern } P = \langle X, D, A, cpt \rangle \text{ that does not contain any of the patterns listed in Lemma 4.1. Thus } P \text{ has at most three variables each with domain size less than three.} \\
\text{We consider first the case of a 2-variable pattern } P. \text{ By Lemma 4.1, } P \text{ does not have two distinct non-mergeable incompatibility edges and does not contain Z. Since } P \text{ is ir-} \\
\text{reducible and hence does not have any dangling assignment, we can deduce by exhausting over all possibilities that } P \text{ does not have any compatibility edge and a single incompatibility edge. Hence } P \text{ is contained in BTP. We can therefore assume that } P \text{ has exactly three variables.} \\
\text{Now consider the negative sub-pattern } P^\prime = \langle X, D, A, neg \rangle \text{ where the compatibility function } \text{cpt} \text{ is } \text{cpt} \text{ with its domain reduced to the incompatible pairs of assignments of } P. \\
\text{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 \text{ must contain either Pivot(\text{sym}) or Pivot(\text{asym}). Now, since } P \text{ does not contain Cycle(3), it follows that } P^\prime \text{ is } I_1 \text{ or } I_2, \text{ as shown in Figure 3.} \\
\text{We first consider the latter case. Without loss of generality, we assume that } b \text{ is compatible with } c, \text{ to avoid } a \text{ and } b \text{ being mergeable.} \\
\text{Since the domains have at most two elements, we begin by assuming that } D(v_1) = \{c, d\} \text{ and } D(v_2) = \{e, f\}. \text{ In this case } a \text{ and } d \text{ must be compatible to avoid } d \text{ and } c \text{ being mergeable. Also } b \text{ and } f \text{ must be compatible to stop } c \text{ and } f \text{ being mergeable. Moreover, } d \text{ and } e \text{ cannot be compatible since otherwise XL occurs in } P. \text{ Furthermore, } d \text{ and } f \text{ cannot be compatible since, whichever variable is chosen for } e(P),
\end{align*}
\]
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
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 $e$ to avoid $f$ being a dangling assignment. In this case $P$
contains Triangle.

Finally, we have $D(v_3) = \{e\}$ and $D(v_2) = \{e\}$. Suppose
that there is a compatibility edge between $e$ and $c$. If the dis-
tinguished variable $\tau(P) = v_0$ then, whether or not there is a
compatibility edge between $a$ and $e$, the pattern is contained
in BTP. If $\tau(P) = v_1$ and there is no compatibility edge be-
tween $a$ and $e$, then the pattern is contained in invsubBTP. If
$\tau(P) = v_1$ and there is a compatibility edge between $a$ and $e$,
then the pattern contains rotsubBTP. If $\tau(P) = v_2$, then the
pattern contains rotsubBTP. Since we have covered all cases in
which there is a compatibility edge between $e$ and $c$, we
assume that there is no edge between $e$ and $c$.

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

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 tran-
sitivity 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 The-
orem 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 allow-
ing 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 exist-
ential patterns after a little extra work.

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

**Proof:** We know from Theorem 3.1 that $\exists subBTP$,
$\exists invsubBTP$, $\exists snake$ are VE patterns.

Theorem 4.5 and Corollary 4.4 show that when we flatten
an existential VE pattern then the resulting flat quantified pat-
tern 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
flattens 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 $\tau(P)$
are mergeable unless $P$ contains $V(\pm\pm)$. Hence, either $P$ is
reducible or, by Lemma 4.1 and Lemma 4.3, $P$ is not a VE
pattern.

Let $\exists BTP$ denote the existential version $Q$ of BTP such
that $|e(Q)| = 1$. By symmetry, $\exists BTP$ is unique. The only re-
maining case is when $P$ is an irreducible subpattern of $\exists BTP
with $|e(P)| = 1$. By a straightforward exhaustive case anal-
ysis, we find that, in this case, either $P$ contains $V(\pm\pm)$ or
Triangle(asym) or $P$ is a sub-pattern of $\exists subBTP$. 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 $BTP$, $\exists subBTP$, $\exists invsubBTP$, $\exists snake$ (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 quanti-
fied 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 char-
acterisation 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 pat-
terns 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 pat-
terns 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


