A Dichotomy for 2-Constraint Forbidden CSP Patterns

Martin C. Cooper and Guillaume Escamocher
IRIT, University of Toulouse, France (cooper@irit.fr)

Abstract

Novel tractable classes of the binary CSP (constraint satisfaction problem) have recently been discovered by studying classes of instances defined by excluding subproblems described by patterns. The complete characterisation of all tractable classes defined by forbidden patterns is a challenging problem. We demonstrate a dichotomy in the case of forbidden patterns consisting of two constraints.

Introduction

In a CSP instance the aim is to determine the existence of an assignment of values to variables such that a set of constraints are simultaneously satisfied. A fundamental research question is the identification of tractable subclasses of CSP. Classical approaches consist in identifying restrictions either on the constraint relations or on the (hyper)graph of constraint scopes which imply the existence of a polynomial-time algorithm. In some cases, dichotomies have even been proved (Bulatov, Jeavons, and Krokhin 2005; Bulatov 2006; Grohe 2007; Marx 2010).

Recently, a new avenue of research has been investigated: the identification of tractable classes of CSP instances defined by forbidding a specific (set of) subproblem(s). Novel tractable classes have been discovered by forbidding simple 3-variable subproblems (Cooper, Jeavons, and Salamon 2010; Cooper and Živný 2011b). This paper presents an essential first step towards the identification of all such tractable classes, namely a dichotomy for the special case of forbidden 2-constraint subproblems.

We first define the notion of a CSP pattern. A pattern can represent a set of subproblems by leaving the consistency of some tuples undefined. We use the term point to denote an assignment of a value to a variable. A pattern is a graph in which vertices correspond to points and both vertices and edges are labelled. The label of a vertex corresponding to a variable-value assignment \( \langle v, d \rangle \) is simply the variable \( v \) and the label of an edge between two vertices describes the compatibility of the corresponding pair of assignments.

Definition 1. A pattern is a quintuplet \( \langle V, A, var, E, cpt \rangle \) comprising: a set \( V \) of variables, a set \( A \) of points (assignments), a variable function \( \text{var} : A \rightarrow V \), a set \( E \subseteq \binom{A}{2} \) of edges (unordered pairs of elements of \( A \)) such that \( \{a, b\} \in E \Rightarrow \text{var}(a) \neq \text{var}(b) \), and a Boolean-valued compatibility function \( \text{cpt} : E \rightarrow \{F, T\} \), where for notational simplicity we write \( \text{cpt}(a, b) \) instead of \( \text{cpt}(\{a, b\}) \).

For a pattern \( P = \langle V, A, \text{var}, E, \text{cpt} \rangle \) and a variable \( v \in V \), we use \( A_v \) to denote the set of assignments \( \{a \in A \mid \text{var}(a) = v\} \) to \( v \). If \( \text{cpt}(a, b) = T \) then the two assignments (points) \( a, b \) are compatible and \( \{a, b\} \) is a compatibility edge; if \( \text{cpt}(a, b) = F \) then the two assignments \( a, b \) are incompatible and \( \{a, b\} \) is an incompatibility edge. In a pattern, the compatibility of a pair of points \( a, b \) such that \( \text{var}(a) \neq \text{var}(b) \) and \( \{a, b\} \notin E \) is undefined.

A binary CSP instance is a pattern \( \langle V, A, \text{var}, E, \text{cpt} \rangle \) in which the compatibility of each pair of assignments to distinct variables is specified by the compatibility function. The question corresponding to the instance is: does there exist a solution, that is a pairwise-compatible set of assignments to all variables in \( V \)? The constraint on variables \( v_1, v_2 \in V \) is the 2-variable sub-instance \( \langle \{v_1, v_2\}, A_{v_1}, \text{var}|_{A_{v_1}}, E_{v_2}, \text{cpt}|_{E_{v_2}} \rangle \) where \( A_{v_1} = A_{v_2} \) and \( E_{v_2} = \{\{a, b\} \mid a \in A_{v_1}, b \in A_{v_2}\} \). The constraint between variables \( v_1, v_2 \) in an instance is non-trivial if there is at least one incompatibility pair of assignments, i.e. \( a \in A_{v_1} \) and \( b \in A_{v_2} \) such that \( \text{cpt}(a, b) = F \).

A pattern is a compact way of representing the set of all instances obtained by arbitrarily specifying the compatibility of its undefined pairs. Two patterns \( P \) and \( Q \) are isomorphic if they are identical except for a possible renaming of variables and assignments.

In a CSP instance \( \langle V, A, \text{var}, E, \text{cpt} \rangle \), we call the set \( \{d \mid \langle v, d \rangle \in A\} \) of values that can be assigned to variable \( v \) the domain of \( v \). The constraint graph of an instance \( \langle V, A, \text{var}, E, \text{cpt} \rangle \) is \( (V, H) \), where \( H \) is the set of pairs of variables \( v_1, v_2 \in V \) such that the constraint on \( v_1, v_2 \) is non-trivial.

Definition 2. We say that a pattern \( P \) occurs in a pattern \( P' \) (and \( P' \) contains \( P \)) if \( P' \) is isomorphic to a pattern \( Q \) in the transitive closure of the following two operations (extension and merging) applied to \( P \):

extension \( P = \langle V_P, A_P, \text{var}_P, E_P, \text{cpt}_P \rangle \) is a sub-pattern of \( Q = \langle V_Q, A_Q, \text{var}_Q, E_Q, \text{cpt}_Q \rangle \): \( V_P \subseteq V_Q, A_P \subseteq A_Q, \text{var}_P = \text{var}_Q|_P, E_P \subseteq E_Q, \text{cpt}_P = \text{cpt}_Q|_{E_P} \).
merging Merging two points in \( P = (V_P, A_P, \text{var}_P, E_P, \text{cpt}_P) \) transforms \( P \) into \( Q = (V_Q, A_Q, \text{var}_Q, E_Q, \text{cpt}_Q) \): \( \exists a, b \in A_P \) such that \( \text{var}_P(a) = \text{var}_P(b) \) and \( \forall c \in A_P \) such that \( \{a, c\}, \{b, c\} \in E_P \), \( \text{cpt}_P(a, c) = \text{cpt}_P(b, c) \). Furthermore, \( V_P = V_Q, A_Q = A_P \setminus \{b\}, \text{var}_Q = \text{var}_P|_{A_Q}, E_Q = E_P \cup \{(a, x) \mid \{b, x\} \in E_P\} \) and \( \text{cpt}_Q(a, x) = \text{cpt}_Q(b, x) \) if \( \{b, x\} \in E_P, \text{cpt}_Q(c) = \text{cpt}_P(c) \) otherwise.

Consider the three patterns shown in Figure 1. Assignments (points) are represented by bullets, and assignments to the same variable \( v \) are grouped together within an oval representing \( A_v \). Solid lines represent compatibility edges and dashed lines incompatibility edges. For example, \( P \) consists of 4 points \( a, b, c, d \in A_v \), such that \( \text{cpt}(a, c) = \text{cpt}(b, c) = T \) and \( \text{cpt}(b, d) = F \). \( P \) occurs in \( Z \) since \( Z \) is an extension of \( P \), \( P \) also occurs in \( V \) since \( V \) can be obtained from \( P \) by merging points \( a, b \).

Definition 3. If \( P \) is a pattern, CSP(\( \mathcal{P} \)) denotes the set of binary CSP instances \( Q \) in which \( P \) does not occur. Pattern \( P \) is tractable if there is a polynomial-time algorithm to solve CSP(\( \mathcal{P} \)): \( P \) is intractable if CSP(\( \mathcal{P} \)) is NP-complete.

Preprocessing Operations

This section describes polynomial-time simplification operations that can be applied to a CSP instance \( \langle V, A, \text{var}, E, \text{cpt} \rangle \). If for some variable \( u, A_u \) is a singleton \( \{a\} \), then the elimination of a single-valued variable corresponds to making the assignment \( a \) and consists of eliminating \( v \) from \( V \) and eliminating \( a \) from \( A \) along with all assignments \( b \) which are incompatible with \( a \).

Arc consistency consists in eliminating from \( A \) all assignments \( a \) for which there is some variable \( v \neq \text{var}(a) \) in \( V \) such that \( \forall b \in A_v, \text{cpt}(a, b) = F \).

If \( \text{var}(a) = \text{var}(b) \) and for all variables \( v \neq \text{var}(a), \forall c \in A_v, \text{cpt}(a, c) = T \Rightarrow \text{cpt}(b, c) = T \), then we can eliminate \( a \) from \( A \) by neighbourhood substitution, since in any solution in which \( a \) appears, we can replace \( a \) by \( b \) (Freuder 1991; Cooper 1997). None of single-valued variable elimination, arc consistency or neighbourhood substitution when applied to an instance in CSP(\( \mathcal{P} \)) can introduce the forbidden pattern \( P \). To simplify our proofs, we assume throughout that we have applied these three operations until convergence to all CSP instances.

We now consider two new simplification operations. They are simplification operations that can be applied to certain CSP instances. We can always perform the fusion of two variables \( v_1, v_2 \) in a CSP instance into a single variable \( v \) whose set of assignments is the cartesian product of the sets of assignments to \( v_1 \) and to \( v_2 \). Under certain conditions, we do not need to keep all elements of this cartesian product and, indeed, the total number of assignments actually decreases.

Definition 4. Consider a CSP instance \( \langle V, A, \text{var}, E, \text{cpt} \rangle \) with \( v_1, v_2 \in V \). Suppose that there is a fusion function \( f : A_{v_1} \rightarrow A_{v_2} \), such that \( \forall u \in A_{v_1}, \) whenever \( u \) is in a solution \( S \), there is a solution \( S' \) containing both \( u \) and \( f(u) \). Then we can perform the simple fusion of \( v_2 \) and \( v_1 \) to create a new fused variable \( v \). The resulting instance is \( \langle V', A', \text{var}', E', \text{cpt}' \rangle \) defined by \( V' = (V \setminus \{v_1, v_2\}) \cup \{v\} \), \( A' = A \setminus A_{v_1} \cup A_{v_2} \), \( \text{var}'(v) = \text{var}(u) \) for all \( u \in A' \setminus A_{v_1} \) and \( \text{var}'(u) = v \) for all \( u \in A_{v_1} \), \( E' = \{(p, q) \mid \text{cpt}'(p, q) = \text{cpt}(p, q) \text{ if } p, q \in A' \setminus A_{v_1}, \text{cpt}'(u, q) = \text{cpt}(u, q) \wedge \text{cpt}(f(u), q) \text{ for all } u \in A_{v_1} \text{ and all } q \in A' \setminus (A_{v_1} \cup A_{v_2})\} \).

Definition 5. Consider a CSP instance \( \langle V, A, \text{var}, E, \text{cpt} \rangle \) with \( v_1, v_2 \in V \) and a hinge value \( a \in A_{v_1} \). Suppose that there is a fusion function \( f : A_v \setminus \{a\} \rightarrow A_{v_2} \), such that \( \forall u \in A_v \setminus \{a\}, \) whenever \( u \) is in a solution \( S \), there is a solution \( S' \) containing both \( u \) and \( f(u) \). Then we can perform the complex fusion of \( v_2 \) and \( v_1 \) to create a new fused variable \( v \). The resulting instance is \( \langle V', A', \text{var}', E', \text{cpt}' \rangle \) defined by \( V' = (V \setminus \{v_1, v_2\}) \cup \{v\} \), \( A' = A \setminus \{a\} \), \( \text{var}'(u) = \text{var}(u) \) for all \( u \in A' \setminus (A_{v_1} \cup A_{v_2}) \) and \( \text{var}'(u) = v \) for all \( u \in (A_{v_1} \setminus \{a\}) \cup A_{v_2}, \) \( E' = \{(p, q) \mid v \neq \text{var}'(q), \text{cpt}'(p, q) = \text{cpt}(p, q) \text{ if } p, q \in A' \setminus (A_{v_1} \cup A_{v_2}), \text{cpt}'(u, q) = \text{cpt}(u, q) \wedge \text{cpt}(f(u), q) \text{ for all } u \in A_{v_1} \setminus \{a\} \text{ and all } q \in A' \setminus (A_{v_1} \cup A_{v_2})\} \).

Lemma 1. If \( I \) is a CSP instance and \( I' \) the result of a (simple or complex) fusion of two variables in \( I \), then \( I' \) is solvable iff \( I \) is solvable.

Proof. We give the proof only for the case of a complex fusion, since a simple fusion can be considered as a special case. Among the assignments in the cartesian product of \( A_{v_1} \) and \( A_{v_2} \), it is sufficient, in order to preserve solvability, to keep only those of the form \( (a, q) \) where \( q \in A_{v_2} \) or of the form \( (u, f(u)) \) where \( u \in A_{v_1} \setminus \{a\} \). To complete the proof, it suffices to observe that in \( A' \) we use \( q \in A_{v_2} \) to represent the pair of assignments \( (a, q) \) and \( u \in A_{v_1} \setminus \{a\} \) to represent \( (u, f(u)) \).

Fusion preserves solvability and the total number of assignments decreases by at least 1 (in fact, by \( |A_{v_2}| \) in the case of a simple fusion). However, when solving instances \( I \in \text{CSP}(\mathcal{P}) \), for some pattern \( P \), a fusion operation will only be useful if it does not introduce the forbidden pattern \( P \).

Reduction and Intractable Patterns

In a pattern \( P \), a point \( a \) which is linked by a single compatibility edge to the rest of \( P \) is a dangling point. If an arc consistent instance \( I \) does not contain the pattern \( P \) then it does not contain the pattern \( P' \) which is equivalent to \( P \) in
which the dangling point \(a\) and the corresponding compatibility edge have been deleted. Thus, to decide tractability we only need consider patterns without dangling points.

**Definition 6.** We say that a pattern \(P\) can be reduced to a pattern \(Q\), and that \(Q\) is a reduction of \(P\), if \(Q\) is in the transitive closure of the three operations extension, merging and dp-elimination applied to \(P\), where dp-elimination is the following operation:

**dp-elimination** Eliminating a dangling point and its corresponding compatibility edge from \(P\) transforms \(P\) into \(Q\).

The following lemma follows immediately from the definitions.

**Lemma 2.** Let \(P\) and \(Q\) be two patterns, such that \(P\) can be reduced to \(Q\). Let \(I\) be a CSP instance satisfying arc consistency. If \(Q\) occurs in \(I\), then \(P\) also occurs in \(I\). If \(Q\) is tractable, then \(P\) is tractable. If \(P\) is intractable, then \(Q\) is intractable.

It follows that we only need to study those patterns that cannot be reduced to a known tractable pattern and that are not the reduction of a known intractable pattern. In the remainder of this section we prove some results that are essential for the proof of the 2-constraint dichotomy given in the following section.

**Lemma 3.** Let \(P\) be a pattern such that a constraint in \(P\) contains two distinct incompatibility edges that cannot be merged. Then \(P\) is intractable.

**Proof.** Let \(P\) be a pattern such that a constraint in \(P\) contains two non-mergeable incompatibility edges. Let SAT1 be the set of SAT instances with at most one occurrence of each variable in each clause. SAT1 is trivially equivalent to SAT which is well known to be NP-complete (Cook 1971). It suffices to give a polynomial reduction from SAT1 to CSP(\(\mathcal{P}\)). We suppose that we have a SAT1 instance \(I = \{V, S\}\) with \(V\) a set of variables \(\{v_1, v_2, \ldots, v_n\}\) and \(S\) a set of clauses \(\{C_1, C_2, \ldots, C_k\}\) such that each clause \(C_i\) is a disjunction of \(c_i\) literals \(l_1^i \lor \cdots \lor l_{c_i}^i\). We create the following CSP instance \(I'\):

- \(n + k\) variables \(v_1', \ldots, v_{n+k}\).
- \(\forall v_i'\) with \(1 \leq i \leq n\), two points "\(v_i\)" and "\(v_i\)" in \(A_{v_i'}\).
- \(\forall v_i'\) with \(n + 1 \leq i \leq n + k\), \(c_i - n\) points \(l_1^i \ldots, l_{c_i-n}^i\) in \(A_{v_i'}\).
- \(\forall 1 \leq i \leq k, \forall 1 \leq j \leq c,\) an incompatibility edge between the point \(l_1^i \ldots, l_{c_i}^i\) in \(A_{v_i'}\), and the occurrence in \(A_{v_i'}\) of the literal \(l_1^i\).

By construction, \(I'\) has a solution if and only if \(I\) has a solution. Furthermore, each time an incompatibility edge occurs in a constraint \(C_i\), this constraint \(C_i\) is between a CSP variable \(v_i'\) representing the SAT1 variable \(v_i\) and another CSP variable \(v_{n+j}'\) representing the SAT1 clause \(C_j\). Since \(v_i\) occurs at most once in \(C_j\), then there is only one incompatibility edge in \(C_i\). So \(I'\) does not contain the pattern \(P\). So we have reduced SAT1 to CSP(\(\mathcal{P}\)).

**Definition 7.** Given a pattern \(P = (V, A, \text{var}, E, \text{cpt})\), a variable \(v \in V\), and a point \(a \in A_v\), we say that \(a\) is explicitly compatible (respectively explicitly incompatible) if there is a point \(b \in A\) such that \(a\) is compatible with \(b\) (respectively such that \(a\) is incompatible with \(b\)).

The following lemma follows from the definition of merging.

**Lemma 4.** Let \(P\) be a non-mergeable pattern. Then for every variable \(v \in P\), there is at most one point in \(A_v\) which is not explicitly incompatible.

**Lemma 5.** Let \(Z\) be the pattern on two variables \(v\) and \(v'\) (shown in Figure 1), with points \(a, b \in A_v\), and points \(c, d \in A_{v'}\) such that \(a\) is compatible with both \(c\) and \(d\), \(b\) is compatible with \(c\) and incompatible with \(d\). \(Z\) is intractable.

**Proof.** Since 3-COLOURING is NP-complete (Garey and Johnson 1979), it suffices to give a polynomial reduction from 3-COLOURING to CSP(\(\mathbb{Z}\)), the set of CSP instances in which the pattern \(Z\) does not occur.

Define the constraint \(R_{s,t} = \{(u, v) | (u = s \land v = t) \lor (u \neq s \land v \neq t)\}\) for \(s, t \in \{1, 2, 3\}\).

It is easy to verify that \(R_{s,t}\) does not contain the pattern \(Z\). Consider the 5-variable gadget with variables \(v_1, v_2, u_0, u_1, u_2\), each with domain \(\{1, 2, 3\}\), and with constraints \(R_{k,k}\) on variables \(v_1, u_k\) for \(k = 1, 2, 3\) and constraints \(R_{i+(k \mod 3), k}\) on variables \(u_k, v_j\) for \(k = 1, 2, 3\). The joint effect of these six constraints is simply to impose the constraint \(v_i \neq v_j\). Any instance \((V, E)\) of 3-COLOURING, with \(V = \{1, \ldots, n\}\), can be reduced to an instance of CSP(\(\mathbb{Z}\)) with variables \(v_1, \ldots, v_n\) by placing a copy of this gadget between every pair of variables \((v_i, v_j)\) such that \(\{i, j\} \in E\). This reduction is clearly polynomial.

**Lemma 6.** Any 2-constraint pattern \(P\) on 3 variables in which both constraints contain an incompatibility edge and two intersecting but non-mergeable compatibility edges is intractable.

**Proof.** We will reduce CSP to CSP(\(\mathcal{P}\)). Let \(I\) be a CSP instance. For each \((v, w)\) in \(I\) such that there is a non-trivial constraint between \(v\) and \(w\), we introduce two new variables \(v''\) and \(w''\) such that the domain of \(v''\) is the same as the domain of \(v\), the domain of \(w''\) is the same as the domain of \(w\). We add equality constraints between \(v\) and \(v''\), and between \(w\) and \(w''\), and we add constraints involving \(v''\) or \(w''\) are trivial. We also replace the constraint between \(v\) and \(w\) by a trivial constraint. Let \(I'\) be the instance obtained after all such transformations have been performed on \(I\). By construction, \(I'\) has a solution if and only if \(I\) has a solution.

We now suppose that we have three variables \(v_0, v_1\) and \(v_2\) in \(I'\) such that there are non-trivial constraints between \(v_0\) and \(v_1\) and between \(v_0\) and \(v_2\). By construction, at least one of these constraints is an equality constraint. By definition, a point in an equality constraint is compatible with one and only one point. Hence, \(P\) cannot occur on \(v_0, v_1\) and \(v_2\). This
any incompatibility edge at all, then the pattern is reducible.

Proof. Let \( P \) be the pattern on two variables \( a, b \in A_\nu \) and \( c, d \in A_\mu \) such that \( a \) is incompatible with \( c \) and compatible with \( d \), and \( b \) is compatible with \( c \) and incompatible with \( d \).

Suppose that the gadget \( X \) occurs in an instance. Suppose \( a \) is in a solution \( S \). Let \( e \in A_\nu \) be such that \( v_2 \neq v_0, v_2 \neq v_1 \) and \( e \in S \). Let \( f \) be the point of \( S \) in \( v_1 \).

If \( b \) is incompatible with \( e \), then \( a, b, d \) and \( e \) form the forbidden pattern. So \( b \) is compatible with \( e \). Similarly, if \( c \) is incompatible with \( e \), then \( a, c, f \) and \( e \) form the forbidden pattern. So \( c \) is compatible with \( e \). So if we replace \( a \) by \( b \) and \( f \) by \( e \) in \( S \), then we have another solution. So if \( a \) is in a solution, then \( b \) is also in a solution. So we can remove \( a \) while preserving the solvability of the instance.

So we can assume from now on that the gadget \( X \) doesn’t occur in the instance. The following lemma indicates when we can perform fusion operations.

Lemma 7. Consider an instance \( (a, a') \) and \( (b, b') \) of simple or complex fusion of two variables \( v, v' \) in an instance in \( \text{CSP}(T_1) \). Suppose that whenever \( (a, a') \) and \( (b, b') \) are pairs of fused points during this fusion, such that \( a \neq b, a' \neq b' \), then either \( a \) and \( b' \) are incompatible, or \( a' \) and \( b \) are compatible.

Proof. By the definition of (simple or complex) fusion, the only way that \( T_1 \) could be introduced is when the two points in the right-hand variable of \( T_1 \) are created by the fusion of the two points \( a \) and \( a' \) and \( b \) and \( b' \) such that the compatibility of the points \( a, b, a', b' \) and \( a, b, a', b' \) with the two other points \( c, d \) of \( T_1 \) are given by: \( \text{cpt}(c, a) = \text{cpt}(d, a') = T \), \( \text{cpt}(c, b) = \text{cpt}(c, b') = \text{cpt}(a, a') = \text{cpt}(a, b') = \text{cpt}(d, b') = \text{cpt}(d, b') = T \).

Now, if \( a \) and \( b' \) were incompatible, then \( T_1 \) was already present on points \( a, b, b' \) in the original instance, and hence cannot be introduced by the fusion. Similarly, if \( a' \) and \( a \) were incompatible, then \( T_1 \) was already present on points \( b, a', b' \) in the original instance.

Definition 8. For every \( a, b \in A_\nu \), we say that \( b \) is better than \( a \) with respect to \( v' \), which we denote by \( a \leq b \) for \( v, v' \) (or for \( v' \)), if every point in \( A_\nu \) compatible with \( a \) is also compatible with \( b \).
It is easy to see that \( \leq \) is a partial order. We also have the relations \( \geq, <, > \) and \( = \), derived in the obvious way from \( \leq \).

**Lemma 8.** In an instance in CSP\((\mathcal{T}_1)\),

1. \( \forall (v, v') \), the order \( \leq \) on \( A_v \) with respect to \( v' \) is total.
2. \( \forall a, b \in A_v \), there is \( v' \) such that \( a < b \) for \( v' \).
3. \( \forall a, b \in A_v \), there is only one \( v' \) such that \( a < b \) for \( v' \).

**Proof.**

1. Because the gadget \( X \) cannot occur.
2. Otherwise \( b \) is dominated by \( a \) and we can remove it by neighbourhood substitution.
3. Because of the initial forbidden pattern.

By using the properties on the order \( \leq \), it is possible to show that we can partition the instance into three sets of variables, such that we can apply fusion operations as described in Lemma 7 between two variables from a same set, if the constraint between these two variables is non-trivial. We repeat the process until convergence. It can be shown that one of the three sets is reduced by the fusion operations to a single variable and that at this point, we have an instance with two sets of variables \( F \) and \( G = V \setminus F \) such that:

- there is no non-trivial constraint between \( v \) and \( v' \) if \( v, v' \in F \) or \( v, v' \in G \).
- \( \forall v \in G \), \( \forall g \in A_{v'} \), \( g \) is incompatible with points in \( A_{v'} \) for one and only one variable \( v' \in F \). Furthermore, \( g \) is incompatible with all points of \( A_{v'} \), but one.
- The only possible non-trivial constraint between a variable \( v \in G \) and a variable \( v' \in F \) is of the following form: there is a point \( b \in A_v \) incompatible with all but one of the points in \( A_{v'} \), and \( \forall c \in A_v \) with \( c \neq b \), \( c \) is compatible with all points in \( A_{v'} \).

The total number of assignments decreases when we fuse variables, so the total number of fusions that can be performed is linear in the size of the original instance.

We call NOOSAT (for Non-binary Only Ones SAT) the following problem: a set of variables \( V = \{v_1, v_2, \ldots, v_n\} \), a set of values \( A = \{a_1, a_2, \ldots, a_n\} \), and a set of clauses \( C = \{C_1, C_2, \ldots, C_f\} \) such that: each clause is a disjunction of literals, with a literal being in this case of the form \( v_i = a_j \), and \( \forall i, j, p, q((v_i = a_j) \in C_p) \wedge ((v_i = a_j) \in C_q) \Rightarrow p = q \).

**Lemma 9.** CSP\((\mathcal{T}_1)\) can be reduced to NOOSAT in polynomial time.

**Proof.** We suppose we have a binary CSP instance in CSP\((\mathcal{T}_1)\) and preprocessed as described above. We know that the non-trivial constraints between variables \( v \in G \) and \( v' \in F \) are all of the form \( v = b \Rightarrow v' = a \). Furthermore, each variable-value assignment \( v = b \) occurs in exactly one such constraint. For any \( v \in G \), we can replace the set of such constraints \( v = b_i \Rightarrow v_i = a_i \), for all values \( b_i \) in the domain of \( v \), by the clause \((v_1 = a_1) \vee \ldots \vee (v_d = a_d)\). It only remains to prove that no literal appears in two distinct clauses. Suppose that we have a literal \( v_1 = a \) which occurs in two distinct clauses. Then there must have been two constraints \( v_2 = b \Rightarrow v_1 = a \) and \( v_3 = c \Rightarrow v_1 = a \) and with \( v_1 \in F, v_2 \neq v_3 \in G \). Let \( a' \neq a \) be a point in \( A_{v_1} \). Then \( b \) and \( c \) are both incompatible with \( a' \) but compatible with \( a \). But this is precisely the forbidden pattern. This contradiction shows that CSP\((\mathcal{T}_1)\) can be reduced to NOOSAT.

The constraints in NOOSAT are convex when viewed as \( \{0, \infty\}\)-valued cost functions, and the clauses are non-overlapping. So, from (Cooper and Živný 2011a), it is solvable in polynomial time, and hence \( T_1 \) is tractable.

**Proof of tractability of \( T_2 \):** Consider an instance from CSP\((\mathcal{T}_2)\).

Let \( N \) be the gadget shown in Figure 3: two variables \( v_0 \) and \( v_1 \) with points \( a, f \in A_{v_0}, b, c \in A_{v_1} \), with \( b \neq c \), such that \( a \) is compatible with both \( b \) and \( c \), and \( f \) is incompatible with \( c \). Suppose we have the gadget \( N \). Let \( v_2 \) be a variable with \( v_2 \neq v_0, v_2 \neq v_1 \) and let \( e \) be a point in \( A_{v_2} \) such that \( b \) and \( e \) are compatible. If \( c \) is incompatible with \( e \), then we have the forbidden pattern \( T_2 \) on \( a, f, c, b, e \). So \( c \) is compatible with \( e \). If all the points in \( A_{v_0} \) which are compatible with \( b \) are also compatible with \( c \), then we can remove \( b \) by neighbourhood substitution. So, assuming that neighbourhood substitution operations have been applied until convergence, if we have the gadget \( N \), then there is a point \( g \in A_{v_0} \) compatible with \( b \) and incompatible with \( c \).

Let \( v_3 \neq v_0, v_1 \). By arc consistency, there is \( h \in A_{v_0} \) such that \( h \) is compatible with \( b \). If \( c \) and \( h \) are compatible, then we have the forbidden pattern \( T_2 \) on \( a, g, c, b, h \). So \( c \) and \( h \) are compatible. If there is \( i \in A_{v_0} \) such that \( c \) and \( i \) are incompatible, then we have the forbidden pattern on \( h, i, c, b, g \). So \( c \) is compatible with all the points in \( A_{v_1} \). So, if we have the gadget \( N \), then \( c \) is compatible with all the points outside \( A_{v_0} \cup A_{v_1} \).

**Definition 9.** A constraint \( C \) between two variables \( v \) and \( v' \) is functional from \( v \) to \( v' \) if \( \forall a \in A_v \), there is one and only one point in \( A_{v'} \) compatible with \( a \).

Let the gadget \( V^- \) be the pattern comprising three variables \( v_4, v_5, v_6 \) and points \( a \in A_{v_4}, b \in A_{v_5}, c \in A_{v_6} \) such that \( a \) is incompatible with both \( b \) and \( c \).

From now on, since \( V^- \) is a tractable pattern (Cooper and Živný 2011b), we only need to consider the connected components of the constraint graph which contain \( V^- \). We say a point \( p \) is weakly incompatible with a variable \( v \) if there exists some \( q \in A_v \), such that \( p \) is incompatible with \( q \).
Lemma 10. If in an instance from CSP(T_2), we have the gadget V^−, then the constraint between v_5 and v_4 is functional from v_5 to v_4 and the constraint between v_4 and v_6 is functional from v_6 to v_4.

Proof. By symmetry, it suffices to prove functionality from v_5 to v_4. We suppose we have the gadget V^−. Let d ∈ A_v_5 be compatible with a. Since a is weakly incompatible with two different variables, a, b and d cannot be part of the gadget N. So the only point in A_v_5 compatible with d is a. So if a point in A_v_5 is compatible with a, then it is only compatible with a. Likewise, if a point in A_v_6 is compatible with a, then it is only compatible with a.

Let f ≠ a be a point in A_v_5. By arc consistency, we have d ∈ A_v_5 and e ∈ A_v_6 such that a is compatible with d and with e. From the previous paragraph, we know that both d and e are incompatible with f.

So d, e and f form the gadget V^−. So each point in A_v_5 and A_v_6 compatible with f is compatible with only one point of A_v_5. So each point in A_v_5 and A_v_6 compatible with a point in A_v_5 is compatible with only one point of A_v_5. By arc consistency, each point of A_v_5 and A_v_6 is compatible with exactly one point of A_v_4. So the constraint between v_4 and v_5 is functional from v_5 to v_4.

Lemma 11. In a connected component of the constraint graph containing V^− of an instance from CSP(T_2), all constraints are either functional or trivial.

Proof. Let P(V) be the following property: V is a connected subgraph of size at least two of the constraint graph and all constraints in V are either functional or trivial.

P(\{v_3, v_5\}) is true from Lemma 10.

Let V_all be the set of all variables of the connected subgraph of the constraint graph containing V^−. Let V be a maximum (with respect to inclusion) subset of V_all for which P(V). Let V' = V_all \ V. Let v' ∈ V'. Let v ∈ V be such that C(v, v') (the constraint on v, v') is non-trivial. So there is d ∈ A_v and e ∈ A_v such that d and e are incompatible. Since V is connected and of cardinality at least two, then there is v'' ∈ V such that C(v, v'') is functional. By arc consistency and elimination of single-valued variables, there is necessarily a point f ∈ A_v such that d and f are incompatible. So d, e and f form the gadget V^−. From Lemma 10 we know C(v, v') is functional. So P(V) is true for all subsets of V_all.

Lemma 12. In an instance from CSP(T_2), for all variables v, all points in A_v are weakly incompatible with the exact same set of variables.

Proof. Let a ∈ A_v be weakly incompatible with v'. So C(v, v') is non-trivial. So C(v, v') is functional.

If C(v, v') is functional from v to v', then a point in A_v can be compatible with only one point in A_v'. We can assume, by elimination of single-valued variables, that there are at least two points in A_v', so every point in A_v is weakly incompatible with v'.

If C(v, v') is functional from v' to v, then let b ≠ a in v. By arc consistency, we know there is c ∈ A_v such that a and c are compatible. Since C(v, v') is functional from v' to v, then c is compatible with only one point in A_v, in that case a, so b is incompatible with c. So every point in A_v is weakly incompatible with v'.

So ∀(v, v'), a ∈ A_v, weakly incompatible with v' ⇒ ∃b ∈ A_v, b weakly incompatible with v'.

Definition 10. A sequence of variables (v_0, v_1, ..., v_k) is a path of functionality if ∀0 ≤ i ≤ k : C(v_i, v_{i+1}) is functional from v_i to v_{i+1}.

Lemma 13. In an instance from CSP(T_2), for all pairs of variables v, v', either v' is a leaf in the constraint graph, or there is a path of functionality from v to v'.

Proof. Since we are in a connected component, there is a path of incompatibility (v_0 = v, v_1, v_2, ..., v_k = v') with all v_i different. If v' is not a leaf, then we have a path of incompatibility (v_0, v_1, v_2, ..., v_{k-1}, v_k, v_{k+1}) with v_{k+1} ≠ v_{k-1}. From Lemma 12 we have a path of incompatibility (a_0 ∈ A_v, a_1 ∈ A_v, ..., a_k ∈ A_v, a_{k+1} ∈ A_v). So ∀1 ≤ i ≤ k, a_{i-1}, a_i and a_{i+1} form the gadget V^−. So from Lemma 10, ∀1 ≤ i ≤ k, C(v_{i-1}, v_i) is functional from v_{i-1} to v_i. So we have a path of functionality from v to v'.

Leaves can be added to an existing solution by arc consistency. So once we have removed all the points we can (from the gadget N) we only have to set an initial variable v_0 and see if the q chains of implications (with q being the number of points in A_v) lead to a solution. So the pattern T_2 is tractable.

Proof of tractability of T_3: Consider an instance from CSP(T_3).

Suppose that the gadget N, shown in Figure 3, occurs in the instance and let d be a point in A_v_2, with v_2 ≠ 0, v_1. If d is compatible with c but not with b, then we have the forbidden pattern T_3. So if c is compatible with a point outside of A_v_0, then b is also compatible with the same point.

Let S be a solution containing c. Let e be the point of S in A_v_0. If e is compatible with b, then we can replace c by b in S while maintaining the correctness of the solution, since all the points in the instance outside of A_v_0 which are compatible with c are also compatible with b.

If e is not compatible with b, then edges {b, e}, {e, c} and {c, a} form the gadget N. So, by our previous argument, if e is compatible with a point outside of A_v_0, then a is also compatible with the same point. We can then replace c by b and e by a in S while maintaining the correctness of the solution, since all the points in the instance outside of A_v_0 which are compatible with c are also compatible with b, and e is compatible with a, and e is also compatible with the same point. We can then replace c by b and e by a in S while maintaining the correctness of the solution, since all the points in the instance outside of A_v_0 which are compatible with c are also compatible with b, and all the points in the instance outside of A_v_0 which are compatible with e are also compatible with a. So if a solution contains c, then there is another solution containing b. Thus we can remove c while preserving solvability.

So each time the gadget N is present, we can remove one of its points and hence eliminate N. The gadget N is a known tractable pattern since forbidding N is equivalent to saying that all constraints are either trivial or bijections.
So if it is not present, then the instance is tractable. It follows that the pattern \( T_3 \) is tractable.

**Proof of tractability of \( T_4 \):** Consider an instance from CSP(\( T_4 \)).

Let \( W \) be the gadget shown in Figure 3: two variables \( v_0 \) and \( v_1 \) such that we have \( a \in A_{v_0}, b, c, g \in A_{v_1} \), with \( b \neq c \), \( a \) compatible with both \( b \) and \( c \), and \( a \) incompatible with \( g \). Suppose we have \( W \) in the instance.

Let \( f \) be a point in \( A_{v_0} \), with \( v_2 \neq v_0, v_1 \). If \( f \) is compatible with \( b \) but not with \( c \) (or compatible with \( c \) but not with \( b \)), then we have the forbidden pattern \( T_f \). So all the points of the instance not in \( A_{v_0} \) or \( A_{v_1} \) have the same compatibility towards \( b \) and \( c \).

If all points in \( A_{v_0} \), compatible with \( b \) are also compatible with \( c \), then all the points in the instance compatible with \( b \) are also compatible with \( c \) and by neighborhood substitution we can remove \( b \). Thus we can assume there is \( d \) in \( A_{v_0} \) such that \( d \) is compatible with \( b \) but not with \( c \).

Let \( S \) be a solution containing \( c \). Let \( e \) be the point of \( S \) in \( v_0 \). If \( e \) is compatible with \( b \), then we can replace \( e \) by \( b \) in \( S \) while maintaining the correctness of the solution, since \( b \) and \( c \) have the same compatibility towards all the points in the instance outside of \( A_{v_0} \) and \( A_{v_1} \). If \( e \) is not compatible with \( b \), then edges \( \{b, e\}, \{b, a\} \) and \( \{b, d\} \) form the gadget \( W \). So, by our argument above, \( a \) and \( d \) have the same compatibility towards all the points in the instance outside of \( A_{v_0} \) and \( A_{v_1} \). Similarly, edges \( \{c, d\}, \{c, a\} \) and \( \{c, e\} \) form the gadget \( W \). So \( a \) and \( e \) have the same compatibility towards all the points in the instance outside of \( A_{v_0} \) and \( A_{v_1} \). Thus we can replace \( c \) by \( b \) and \( e \) by \( d \) in \( S \) while maintaining the correctness of the solution, since \( b \) and \( c \) have the same compatibility towards all the points in the instance outside of \( A_{v_0} \) and \( A_{v_1} \). So \( d \) and \( e \) have the same compatibility towards all the points in the instance outside of \( A_{v_0} \) and \( A_{v_1} \). Thus we can remove \( c \).

Therefore, each time the gadget \( W \) is present, we can remove one of its points. The gadget \( W \) is a known tractable pattern since forbidding \( W \) is equivalent to saying that all constraints are zero-one-all (Cooper, Cohen, and Jeavons 1994). So if it is not present, the instance is tractable. Hence the pattern \( T_4 \) is tractable.

**Proof of tractability of \( T_5 \):** The pattern \( T_5 \) is a sub-pattern of the broken-triangle pattern BTP, a known tractable pattern (Cooper, Jeavons, and Salamon 2010) on three constraints. So the pattern \( T_5 \) is tractable. \( \square \)

**Conclusion**

We have proved a dichotomy for classes of binary CSP instances defined by forbidding 2-constraint patterns. This has allowed us to identify novel tractable classes, including, for example, a new generalisation of zero-one-all constraints. An avenue for future research is to investigate the possible generalisations of the five tractable classes defined by forbidding patterns \( T_1,\ldots, T_5 \), by replacing binary constraints by \( k \)-ary constraints \((k > 2)\) or by adding extra constraints to the patterns.

**References**


