THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *23 Avril 2014* **par :**

Guillaume ESCAMOCHER

Forbidden Patterns in Constraint Satisfaction Problems

JURY

MARTIN C. COOPER	Professeur d'Université	Membre du Jury
MICHEL HABIB	Professeur d'Université	Rapporteur
PETER JONSSON	Professor	Rapporteur
FLORENT MADELAINE	Maître de Conférences, HDR	Membre du Jury
PIERRE RÉGNIER	Maître de Conférences, HDR	Membre du Jury

École doctorale et spécialité :

MITT : Domaine STIC : Intelligence Artificielle

Unité de Recherche :

Institut de Recherche en Informatique de Toulouse

Directeurs de Thèse :

Martin C. COOPER et *Pierre REGNIER*

Résumé

Le problème de satisfaction de contraintes (CSP) est NP-complet, même dans le cas où toutes les contraintes sont binaires. Cependant, certaines classes d'instances CSP sont traitables. Récemment, une nouvelle méthode pour définir de telles classes a émergée. Cette approche est centrée autour des motifs interdits, ou l'absence locale de certaines conditions. Elle est l'objet de ma thèse.

Nous définissons formellement ce que sont les motifs interdits, présentons les propriétés qu'ils détiennent, et finalement les utilisons afin d'établir plusieurs résultats de complexité importants. En utilisant différentes versions de motifs, toutes basées sur le même concept de base, nous énumérons un nombre important de nouvelles classes traitables, ainsi que certaines NP-completes. Nous combinons ces résultats pour révéler plusieurs dichotomies, chacune englobant une large gamme de classes d'instances CSP.

Nous montrons aussi que les motifs interdits représentent un outil intéressant pour la simplification d'instances CSPs. Nous donnons plusieurs nouveaux moyens de réduire la taille des instances CSP, que ce soit en éliminant des variables ou en fusionnant les domaines, et montrons comment ces méthodes sont activées par l'absence locale de certains modèles. Comme les conditions de leur utilisation sont entièrement locales, nos opérations peuvent être utilisés sur un large éventail de problèmes.

Mots clés:
problème de satisfaction de contraintes, motif interdit, classe traitable, élimination de variables

Abstract

The Constraint Satisfaction Problem (CSP) is NP-Complete, even in the case where all constraints are binary. However, some classes of CSP instances are tractable. Recently, a new method for defining such classes has emerged. This approach is centered around forbidden patterns, or the local absence of some conditions. It is the focus of my thesis.

We formally define what forbidden patterns are, exhibit the properties they hold, and eventually put them to use in order to establish several important tractability results. Using different versions of patterns, all based on the same core concept, we list a significant number of new tractable classes, as well as some NP-Complete ones. We combine these results to reveal several dichotomies, each one encompassing a large range of classes of CSP instances.

We also show how useful a tool forbidden patterns can be in the field of CSP instance simplification. We give multiple new ways of decreasing the size of CSP instances, whether by eliminating variables or fusioning domains, and prove how all these methods are enabled by the local absence of some patterns. Since the conditions for their use are entirely local, our operations can be used on a wide array of problems.

Keywords:
constraint satisfaction problem, forbidden pattern, tractable class, variable elimination

Thanks

First of all, I would like to thank Martin Cooper. Martin was of tremendous help throughout all of my PhD. Not only was he, as my supervisor, the driving force directing my research efforts, but he was also the one person who gave me the tools of the trade. In order to succeed in an intellectual endeavor, one must possess logical thinking, and find what original ideas will lead to the discovery of new results. Martin gave me the former and showed me the latter. If I acquired a scientific mind during these last three years, I owe it to him.

I would also like to thank my co-supervisor Pierre Régnier. Whenever I had an enquiry, on any topic, he was always available to answer my questions. In particular, Pierre helped me structure my work when I was almost overwhelmed with the task ahead of me. Without him, the quality of this thesis would have been far poorer than it is.

Standa Živný and Dave Cohen were co-authors of most of the papers we published during my PhD. As such, their contributions can be directly seen in my thesis. I thank them immensely for that.

I also thank Frédéric Maris, Philippe Jégou and all other members of the TUPLES project. The project itself provided the necessary funding for my PhD, and the people I met within the project gave me an invaluable insight into the scientific community.

I would like to thank my family and my friends as well. They were often on my mind during all these years.

Finally, I would like to mention the people who kindly accepted to read, report or comment on my thesis. Michel Habib, Peter Jonsson, Florent Madelaine and Emmanuel Hebrard showed great interest in my work and for that I thank them.

Contents

1 Introduction

1.1 From Problems to Patterns

A main goal of Artificial Intelligence, or AI, is to produce decision makers. This can be decomposed in two parts, in the same way that Artificial Intelligence is composed of two words. The first word is "artificial" and it reflects "produce". Artificial means it does not come out of nowhere, it is crafted. Artificial Intelligence does not just study theoretical properties, it crafts algorithms and creates solutions. The second word in Artificial Intelligence is "intelligence" and it reflects "decision". Intelligence is the core of the discipline, its driving force. It is what distinguishes it from the rest of Art and Science. A builder constructs houses. A composer creates symphonies. A pharmacologist produces cough syrup. These are all necessary or desirable items; they bring shelter, entertainment and healthiness, and so their use and interest cannot be questioned. However, none of them possesses intelligence, or the ability to make insightful decisions. They are useful, but lesser by essence than creations able to make decisions. Intelligence is what gives the inanimate the freedom to choose its actions. It is what separates a thinking entity from a golem. It provides the ability to evaluate different options, compare them, and ultimately make a choice. Intelligence is about solving problems.

Some problems cannot be solved by any algorithm. It is the case for instance of the Halting Problem (Turing, 1936). Other problems can be solved by some algorithm, but the time required to do so would be so long, millions of years for the current best computers, that it is often not worth the effort. One such example is the problem of whether or not two different regular expressions represent the same language (Meyer & Stockmeyer, 1972). Fortunately, a lot of problems, including most problems which arise in everyday life, can be solved in a relatively affordable time. They are in NP, the set of problems solvable in polynomial time by a non-deterministic Turing machine. Most work in computer science, or at least most work with the intent of solving problems, focuses on problems from NP. If P$\neq$NP, then NP-Complete problems, the hardest to solve problems in NP, are not solvable in polynomial time, and may require algorithms with an exponential running time to be solved completely. However, by focusing on a single NP-Complete problem, one can still manage to find polynomial time algorithms which give interesting results. This can be done by only focusing on a subset of all possible instances of this problem. If successful, this approach leads to a tractable class. This can also be done by approximating, the act of finding a solution to the problem which is non-optimal, but close enough to be useful. One such work is (Raghavendra, 2008). Additionally, one can use selected randomized algorithms. Such algorithms can give an optimal solution in a polynomial time for any instance, but with a probability of only 1-ϵ, with ϵ very small (Motwani & Raghavan, 1995).

NP is a very large class of very diverse problems. The Subset-Sum problem and the Travelling Salesman problem seem to bear little ressemblance. However, despite this diversity, focusing on each problem individually is not the way to go. Otherwise, all the work done on one problem would be wasted when studying another distinct problem. All NP-Complete problems are reducible in polynomial time to each other, so it is better to first search for efficient algorithms for a select few NP-Complete problems, and then reduce other NP-Complete to these "main" prob-

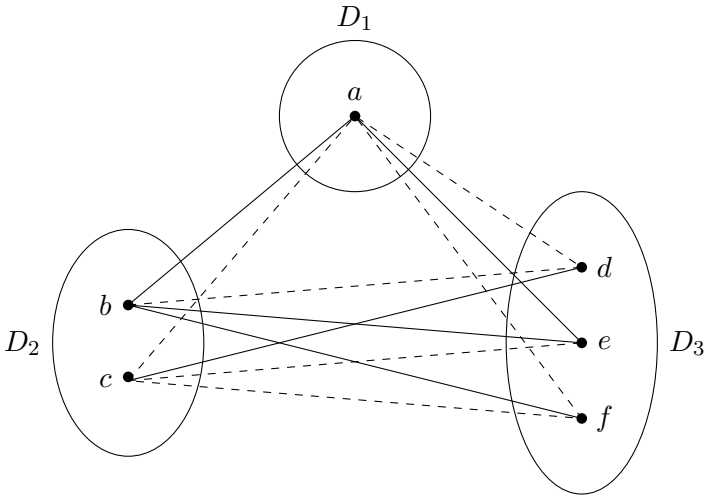


Figure 1: An example of a CSP instance.

lems. Such problems, on which a lot of results have been found and to which many NP-Complete problems are reduced to, include 3-SAT and the Constraint Satisfaction Problem, abbreviated CSP.

A CSP instance can be summarized as follows:

- *Data*
  - A set of n variables $\{v_1, \ldots, v_n\}$.
  - A set of n domains $\{D_1, \ldots, D_n\}$. Each domain D_i is the set of all the possible values the variable v_i can take. $\langle v_i, a_i \rangle$ represents the assignment of the value $a_i \in D_i$ to the variable v_i.
  - A set of m constraints $\{C_1, \ldots, C_m\}$, of respective arities $k_1, \ldots, k_m$. Each constraint C_i has a scope composed of k_i variables $\{v_{i,1}, \ldots, v_{i,k_i}\}$ and describes the k_i-tuples of assignments $\{\langle v_{i,1}, a_{i,1} \rangle, \ldots, \langle v_{i,k_i}, a_{i,k_i} \rangle\}$ that are not allowed for its scope.
- *Question:* Is there a set of n assignments $\{\langle v_1, a_1 \rangle, \ldots, \langle v_n, a_n \rangle\}$ satisfying all constraints?

Not all domains need to have the same size. A set of n assignments satisfying all constraints is called a *solution* to the instance. For $n' \leq n$, a set of n' assignments satisfying all constraints is called a *partial solution*. An example of a CSP instance is given in Figure 1. There are three variables v_1, v_2 and v_3, their associated domains $D_1 = \{a\}$, $D_2 = \{b, c\}$, and $D_3 = \{d, e, f\}$, as well as three binary constraints. Domains are represented by ovals, values in a domain are represented by points in the oval representing the domain, pairs of compatible assignments (also called compatibility edges) are represented by solid lines and pairs of incompatible assignments (also called incompatibility edges) are represented by dashed lines. $\{\langle v_1, a \rangle, \langle v_2, b \rangle, \langle v_3, e \rangle\}$ is a solution to the instance.

Since CSP is a NP-Complete problem, all problems in NP can be reduced in polynomial time to some CSP instance. Furthermore, CSP is a very intuitive problem, so in many cases the reduction will only take a short time. For instance, any 3-SAT instance can be reduced in linear time to a

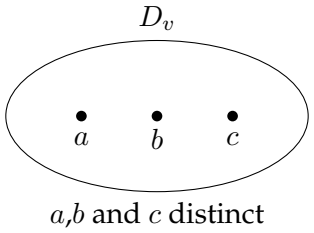


Figure 2: Forbidden pattern in a Binary Boolean CSP instance.

CSP instance. Therefore, finding results about the CSP will not only advance the CSP field, but also help make progress for other seemingly unrelated NP-Complete problems.

Since CSP is NP-Complete, it is not possible to give a polynomial-time algorithm able to solve all CSP instances. However, as stated above, one can look for sets of CSP instances for which such an algorithm exists. Such sets are tractable classes. One example of a tractable class in CSP is the set of binary CSP instances where the size of all domains is at most 2. This result can be acquired by a reduction to 2-SAT. Tractable classes are part of what we have been looking for, along with general improvements to already existing algorithms. These improvements we give can decrease the size of instances satisfying some specific conditions. It does not reduce the worst-case complexity of a given algorithm, but can diminish its running time.

In recent years (as of 2013), a novel approach on CSPs has been considered. This method, which is the main subject of my thesis, focuses on patterns. A pattern is here a local behavior of a CSP instance. The presence (or absence) of these local conditions in a CSP instance can exhibit useful global properties that hold for this instance. Some of these properties include belonging to a tractable class and redundancy of some variables and/or values. In the case of a redundancy, we often can remove the appropriate variable and/or reduce the size of the domain containing the redundant value. These operations result in an instance of smaller size.

Although the concept of forbidden patterns is relevant to any constraint arity, we only considered binary constraints during my PhD. There are two main reasons behind this choice. Firstly, forbidden patterns are more natural in a binary environment. Binary patterns are easier to detect and thus will be more likely to be used in real-life applications. Secondly, binary constraints are more inclined to lead to dichotomic results when dealing with patterns. Dichotomies are often sought after, because they present an efficient way to map the knowledge one has of a given field.

1.2 A New Approach

An example of a known tractable class in CSP that we mentioned previously is the set of all binary CSP instances whose variables can only take two possible values, or Binary Boolean CSP. An equivalent definition for this class would be to consider the set of CSP instances such that there are never three distinct values in any domain. This subproblem, or pattern, which never occurs in such an instance is represented in Figure 2.

Another example of a tractable class is the class ZOA. ZOA stands here for "Zero One All". This class is composed of all CSP instances I such that: if c is a constraint between two variables

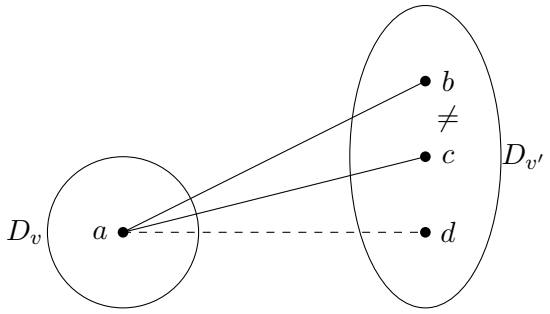


Figure 3: Forbidden pattern in a ZOA instance.

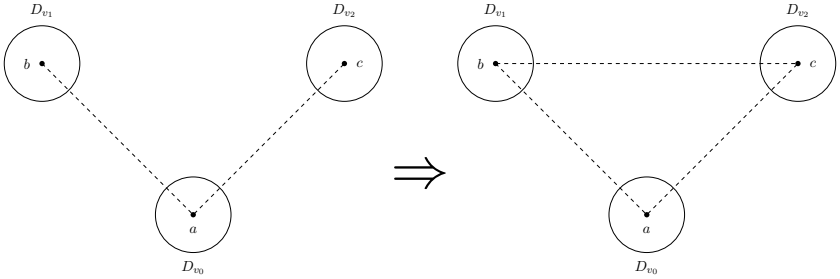


Figure 4: An example of condition on the graph of incompabilities.

v and v' in I, and D_v and $D_{v'}$ are the domains of v and v' respectively, then each value in D_v is compatible with exactly zero values from $D_{v'}$, or exactly one, or all of them. One can notice that in a ZOA instance, the following pattern never occurs: a value a in a domain D_v which is compatible with two distinct values b and c in a domain $D_{v'}$, and which is incompatible with a value d in the same domain $D_{v'}$. This pattern is represented in Figure 3. Binary Boolean CSP is an example of ZOA; since there are at most two values in each domain, the pattern from Figure 3 cannot appear in a Binary Boolean CSP instance.

The class ZOA is defined by restrictions on the constraints. The class Binary Boolean CSP is defined by restrictions on the domains. Since a constraint is defined on several domains, restrictions on the domains can be seen as a special case of restrictions on the constraints. Therefore, Binary Boolean CSP is also defined by restrictions on the constraints.

Studying CSP instances defined by restrictions on their constraints has been one of the main methods of finding tractable classes so far. Another widespread approach is to examine restrictions on some underlying graph of an instance. For instance, consider the following condition on the graph of incompabilities, also known as the colored microstructure complement (Cohen, Cooper, Creed, Marx, & Salamon, 2012): for all triples of domains D_{v_0}, D_{v_1} and D_{v_2}, if there is a value $a \in D_{v_0}$ which is incompatible with a value $b \in D_{v_1}$ and also with another value $c \in D_{v_2}$, then the values b and c are incompatible. Figure 4 presents a graphical representation of this condition. The set of CSP instances whose graph of incompabilities satisfies this condition forms the class NEGTRANS. NEGTRANS is tractable (Cooper & Živný, 2011c) and includes for example All-Diff CSP instances, where all variables must take different values.

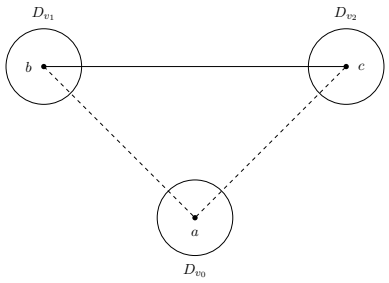


Figure 5: Forbidden pattern in a NEGTRANS instance.

Note that NEGTRANS can also be equivalently defined by the set of CSP instances in which the following pattern never occurs: three values a, b and c in the three distinct domains D_{v_0}, D_{v_1} and D_{v_2} respectively, such that a is incompatible with both b and c, and b is compatible with c. This pattern is represented in Figure 5.

Restrictions on the constraints and restrictions on the subgraphs have been the two main ways to look for tractable classes in CSP. The core of the idea behind the former is to forbid some constraints to occur in an instance, while usually allowing all possible subgraphs. The core of the idea behind the latter is the opposite: forbidding some subgraphs, while generally allowing all possible constraints to appear in an instance. For example, ZOA forbids the constraint represented in Figure 3, but all possible constraint graphs are represented in the ZOA class, with the constraint graph of an instance being the graph whose vertices are composed of the variables of the instance, and where there is an edge between two vertices if and only if there is a constraint between the two corresponding variables in the original instance. As another example, NEGTRANS forbids any projection of the graph of incompatibility edges on k values in k domains to be a tree if $k > 2$, but all possible constraints are represented in the NEGTRANS class.

A forbidden pattern can define a tractable class, if we consider the set of CSP instances in which the pattern never occurs. For example, the pattern from Figure 3 defines the class ZOA and the pattern from Figure 5 defines the class NEGTRANS. In order to find tractable classes, forbidden patterns can be used in methods focusing on the constraints, and they can also be used in methods focusing on the subgraphs. But forbidden patterns are a hybrid class, and therefore offer many more possibilities. Consider the pattern represented in Figure 6. As we will show in section 4.1, the class of CSP instances it defines, which is the set of CSP instances in which it never occurs, is tractable. This pattern combines restrictions on both the constraints (it is actually a generalization of ZOA) and the subgraphs of an instance. By doing so, it allows all possible constraints in the class it defines, as well as all possible constraint graphs.

Thanks to their inherent hybridity, forbidden patterns cover many classes not findable if only using constraints-based restrictions, or only using graph-based restrictions. They can lead to far more refined tractable classes. Furthermore, if a given forbidden pattern P defines a tractable class, then not only there exists an algorithm able to solve in polynomial time any CSP instance not containing P, but the same algorithm can be used to solve CSP instances in which P appears, as long as the number of occurences is limited. This is done by isolating the small part of the

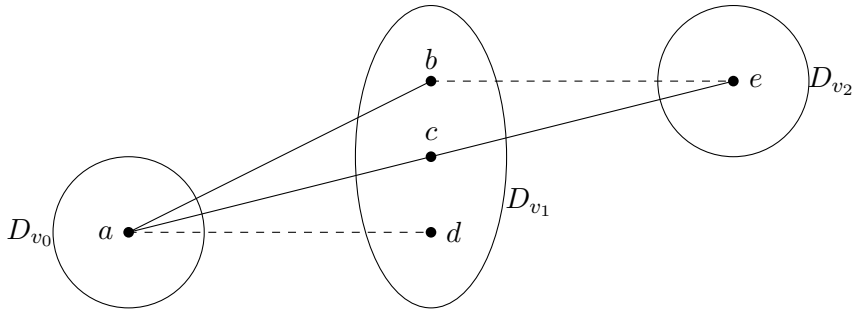


Figure 6: An example of a forbidden pattern.

instance containing P, then solving independently the rest of the instance, in which P does not occur. Additionally, the cost of the detection of a pattern in a CSP instance is polynomial. Thus forbidden patterns can be safely integrated into the search for tractable classes: the presence of a significant number of occurences of a forbidden pattern P in a given instance I won't lead directly to a polynomial-time algorithm able to solve I, but neither will it endanger the polynomiality of the method used to study I.

Forbidden patterns pave the way to new characterizations of tractable classes which would not have been possible otherwise. Unfortunately, not all already-known tractable classes can be defined with a single pattern. For example, the set of CSP instances whose constraint graph is a tree is tractable. This class is graph-based and is not included in any tractable class defined by a single forbidden pattern. However, if we extend the language describing forbidden patterns, we can still exhibit a class based on a forbidden pattern, which contains all CSP instances whose constraint graph is a tree (Cooper, Jeavons, & Salamon, 2010). This class is called the Broken Triangle Property, or BTP, and is defined in the following way: for a given CSP instance I, if there exists an order $(v_1, v_2, \ldots, v_n)$ on the n variables of I such that for all triple $\{v_i, v_j, v_k\}$ of respective domains D_{v_i}, D_{v_j} and D_{v_k}, and with $i < j < k$, for all $a \in D_{v_i}$, $b \in D_{v_j}$, $c, d \in D_{v_k}$, we have (a is compatible with b and c, b is compatible with d) implies (either a is compatible with d or b is compatible with c), then I satisfies BTP. An illustration of the pattern forbidden by BTP is presented in Figure 7. Allowing a forbidden pattern to appear, but controlling the conditions under which the pattern can occur, like BTP does, expands considerably the number and diversity of tractable classes based on a forbidden pattern.

Finding tractable classes is not the only use of forbidden patterns. Patterns can also be a tool for variable and/or value elimination. In some way, they have already been used to this end for a while. Indeed, common operations such as arc-consistency and neighborhood substitution can be viewed as the consequence of the local absence of a specific forbidden pattern. A value a in a domain D_v can be eliminated by arc-consistency if there exists a domain $D_{v'}$ such that no value $b \in D_{v'}$ is compatible with a. This is equivalent to forbidding the value a and the domain $D_{v'}$ to both be part of the forbidden pattern $1C$ composed of a single compatibility edge. The pattern $1C$ is drawn in Figure 8. Note that arc-consistency does not require the pattern $1C$ to not appear at all in the instance. The condition on the absence of $1C$ is only local.

Other simplication operations can also be seen as the local absence of a pattern. Neighborhood

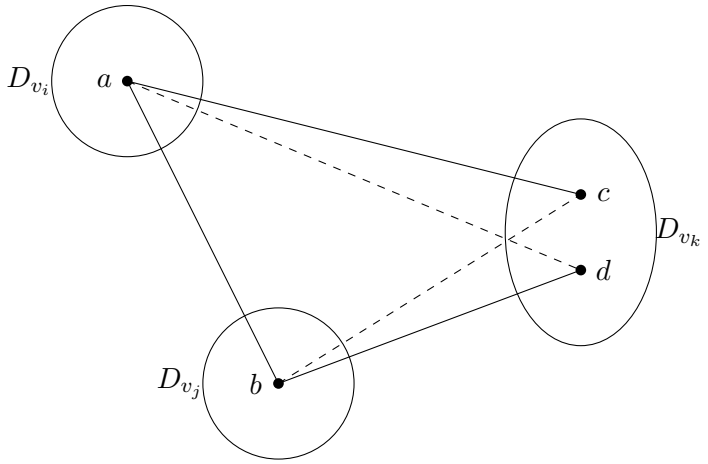


Figure 7: Pattern forbidden by the Broken Triangle Property.

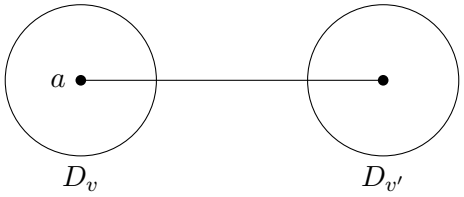


Figure 8: The forbidden pattern $1C$.

substitution, for instance, states that if there exists two values a and b in a same domain D_v such that for any third value c in a different domain $D_{v'}$ we have (b is compatible with c) implies (a is also compatible with c), then we can remove the value b. This is equivalent to forbidding the values a and b to both be part of the forbidden pattern V^{+-}, composed of a compatibility edge and an incompatibility edge intersecting on a value c. The pattern V^{+-} is represented in Figure 9. Here again, the condition of absence on the pattern is only local.

To wrap-up the reasons why forbidden patterns represent a promising new approach to the study of CSP:

- They allow a characterization of CSP classes in a way not possible with any of the previous main methods.

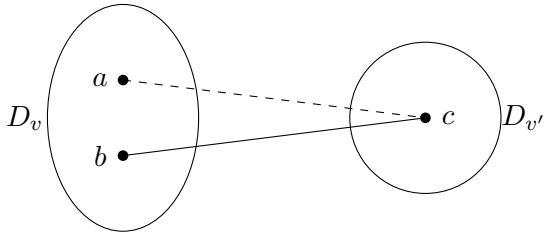


Figure 9: The forbidden pattern V^{+-}.

- They still are able to cover much of the work already done on tractable classes.
- Their language can be easily expanded and generalized, so to not only be used in a global way in the search for tractable classes, but also in a local way in the context of simplification operations.

1.3 Goals

The primary goal of my PhD was to find new tractable CSP classes using forbidden patterns. It has arguably been a success. We have identified numerous novel tractable classes defined by forbidding a specific pattern. We have established a dichotomy on the complexity of forbidden patterns on two constraints; for any pattern on two constraints, we now know if the class it defines, the set of CSP instances in which it never occurs, is tractable or NP-Complete (Cooper & Escamocher, 2012). We also at the same time discovered new tractable classes outside the reach of this dichotomy.

We then decided to generalize the results we found to Max-CSP and VCSP, as well as to constraints of any arity. Max-CSP is the optimization problem corresponding to the decision problem CSP. In a Max-CSP instance, we want to minimize the number of incompatibilities present in a solution. VCSP, short for Valued-CSP, is similar to Max-CSP with arbitrary costs imposed on incompatibility edges.

Our work on Max-CSP was extremely satisfying. We exhibited yet another dichotomy on the complexity of Max-CSP forbidden patterns. We can now determine whether the class defined by a single complete Max-CSP pattern is tractable or NP-Complete (Cooper, Escamocher, & Živný, 2012). We even exposed some minor results on forbidding several Max-CSP patterns at the same time.

Unfortunately, our attempts to generalize our findings to VCSP and constraints of arity 3 or greater were mostly unfruitful. How to properly define a forbidden pattern in a VCSP or in a CSP with constraints of arbitrary arity was the main hurdle we tripped on. We deemed none of the many definitions we tried satisfactory enough.

We then decided to orient our work in a different direction. Instead of forbidding patterns globally in order to find tractable classes, we studied what simplification operations were possible when forbidding patterns locally. We discovered new ways of decreasing the size of a CSP instance using forbidden patterns. Some use a local absence of a pattern to eliminate a variable and/or a value. Others merge several domains into a single domain, resulting in an equivalent instance of smaller size whose satisfiability is the same as that of the original instance. The advances we made in this domain (Cohen, Cooper, Escamocher, & Živný, 2013) not only led to more simplification operations for CSP instances, but also to a more general definition of the concept behind forbidden patterns, which in turn led us to new tractable classes, closing the circle.

1.4 Outline of the Thesis

This introductory chapter aimed to provide a first glance at what a pattern is. Since forbidden patterns are the core notion of my thesis, it was important to familiarize, be it only informally,

the reader with them. We thus provided some explanations, as well as a few examples, helping answer the crucial questions of what forbidden patterns are, why they represent a novel concept, and in what ways we can use them.

The next chapter will present a brief state of the art, in order to indicate certain important research results made in the field of the CSP. We will especially focus on the first few forays into forbidden patterns and similar notions, as well as previous work in problem resolution, such as tractable classes, and instance simplification.

The following chapter will formally define the objects, tools and properties used afterwards. We will begin the section by the definitions of the different types of patterns we use, and of the possible interactions they can have with each other. We then define, or remind the reader of, several important operations on CSP instances. Some of them are known simplification tools which are used ubiquitously in the CSP field. Others stem from our contributions and are needed in the key parts of a couple of proofs. We will end the chapter by the presentation, explanation and examination of the reduction between CSP patterns, a key concept which will be employed throughout the thesis.

After having presented the bases necessary for a perfect understanding of the topic, we will present most of the results discovered during the course of my research. They naturally fall into two separate categories, so I simply elected to spread them over two corresponding chapters. The first one concerns tractable classes. We will present dichotomy results for three different, but related, types of patterns: flat patterns, existential patterns and Max-CSP patterns. We will then close the section with miscellaneous tractability results.

The second chapter deals with instance simplification. It will be divided in two parts. The first one will use the strict definition of forbidden patterns to give a dichotomic result on variable elimination patterns. The second one will generalize a bit the main idea behind forbidden patterns, and show what kind of simplification operations are available when some local conditions are present, or equivalently when the complementary conditions are absent.

Lastly, I will conclude my PhD thesis by recalling where we went with forbidden patterns, and wondering where else we can go with them.

2 State of the Art

2.1 The Constraint Satisfaction Problem

This thesis is centered around the study of the generic combinatorial problem known as the binary constraint satisfaction problem (CSP) in which the aim is to determine the existence of an assignment of values to n variables such that a set of constraints on pairs of variables are simultaneously satisfied.

We can consider an instance of the binary CSP (constraint satisfaction problem) as a labelled graph: vertices are the possible variable-value assignments (which we also call *points*), edges correspond to pairs of compatible points, and each point is labelled by the corresponding variable. Two assignments $\langle v_i, \alpha \rangle$, $\langle v_j, \beta \rangle$ are *compatible* if v_i, v_j are distinct variables and the assignment (α, β) belongs to the relation R_{ij} associated with the constraint on variables (v_i, v_j). This representation of a binary CSP instance is known as its colored microstructure, a version of the microstructure (Jégou, 1993) in which each point is labelled by its corresponding variable. A *solution* to an n-variable CSP instance is simply a size-n set of pairwise compatible points in the (colored) microstructure.

The generic nature of the CSP has led to diverse applications, notably in the fields of Artificial Intelligence and Operations Research (Rossi, Van Beek, & Walsh, 2006). It has also 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; Lecoutre, 2009). Dedicated solvers for constraint satisfaction are at the heart of the programming paradigm known as constraint programming. Theoretical advances on CSPs can thus potentially lead to the improvement of generic combinatorial problem solvers.

Unfortunately, complete solution algorithms for constraint satisfaction are not polynomial time unless P=NP, since the graph coloring problem, which is NP-complete, can be reduced to binary constraint satisfaction (Dechter, 2003).

The binary CSP has diverse applications. In some it is only the satisfiability of the instance which is of interest. For example in planning, to determine whether an action a among a set of available actions A is indispensable (i.e. that it is present in all solution-plans) we need to determine the satisfiability of a binary CSP representing the same planning problem using the set of actions $A \setminus \{a\}$ (Cooper, De Roquemaurel, & Régnier, 2001). All results presented in this thesis are directly applicable to such problems.

2.2 Tractable Classes

A fundamental research question in complexity theory is the identification of tractable subproblems of NP-complete problems. Classical approaches have consisted in identifying types of constraints which imply the existence of a polynomial-time algorithm. Among the most well-known examples, we can cite linear constraints and Horn clauses. In an orthogonal approach, restrictions are placed solely on the (hyper)graph of constraint scopes, known as the constraint (hyper)graph. In some cases, dichotomies have even been proved characterizing all tractable classes definable by

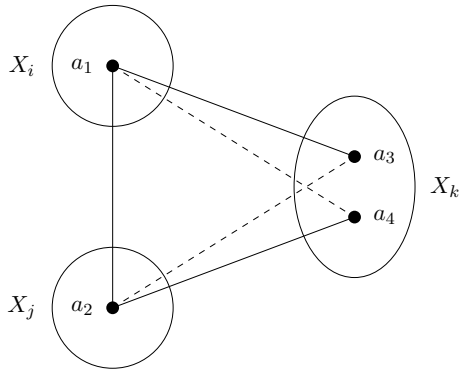


Figure 10: Pattern forbidden by the broken-triangle property.

placing restrictions either on the constraint relations (Bulatov, Jeavons, & Krokhin, 2005; Bulatov, 2006, 2003) or on the constraint (hyper)graph (Grohe, 2007; Marx, 2010a, 2010b). Other complexity results include the tractability of instances possessing a microstructure forming a perfect graph (Salamon & Jeavons, 2008).

Recently, a new avenue of research has been investigated: the identification of tractable classes of CSP instances defined by forbidding a specific (set of) subproblems. Novel tractable classes have been discovered by forbidding simple 3-variable subproblems (Cooper et al., 2010; Cooper & Živný, 2011b). A dichotomy has even been discovered for classes of binary CSP instances defined by forbidding configurations of incompatibilities (Cohen et al., 2012).

One concrete example of a tractable class defined by forbidding a generic subproblem (known as a pattern) is the set of binary CSP instances satisfying the broken-triangle property (Cooper et al., 2010): a binary CSP instance on variables $v_1, \ldots, v_n$ satisfies the broken-triangle property if $\forall i < j < k \in \{1, \ldots, n\}$, whenever the assignments $a_1 = \langle v_i, a\rangle$, $a_2 = \langle v_j, b\rangle$, $a_3 = \langle v_k, c\rangle$, $a_4 = \langle v_k, d\rangle$ are such that the pairs of assignments (a_1, a_2), (a_1, a_3), (a_2, a_4) are compatible, then at least one of the pairs of assignments (a_1, a_4), (a_2, a_3) is also compatible. The forbidden subproblem is shown in Figure 10. For example, any binary CSP instance whose constraint graph is a tree satisfies the broken-triangle property for some ordering of its variables; furthermore such an ordering can be determined in polynomial time. However, tractability is not due to a property of the constraint graph, since instances satisfying the broken-triangle property exist for arbitrary constraint graphs. As we will see later, the broken-triangle property also inspired our development of simplification operations based on the absence of patterns of compatibilities and incompatibilities on particular variables and values (that we refer to as existential patterns).

Two other examples of forbidden patterns which define tractable classes of binary CSP instances are based on the transitivity of compatibilities or incompatibilities (Cooper & Živný, 2012). The former class consists of all binary CSP instances in which for all triples of assignments $a_1 = \langle v_i, a\rangle$, $a_2 = \langle v_j, b\rangle$, $a_3 = \langle v_k, c\rangle$ to three distinct variables, whenever the pairs (a_1, a_2), (a_2, a_3) are both compatible, the third pair (a_1, a_3) is also compatible. The latter class consists of all binary CSP instances in which for all triples of assignments $a_1 = \langle v_i, a\rangle$, $a_2 = \langle v_j, b\rangle$, $a_3 = \langle v_k, c\rangle$ to three distinct variables, whenever the pairs (a_1, a_2), (a_2, a_3) are both incompatible, the third pair (a_1, a_3)

is also incompatible. This property is satisfied, for example, by instances consisting of unary constraints and non-overlapping AllDifferent constraints (since $a = b \wedge b = c \Rightarrow a = c$). The class of binary CSP instances satisfying this negative-transitivity property has been generalized to a large tractable class of optimisation problems involving cost functions of arbitrary arity (Cooper & Živný, 2011b, 2012).

Some tractable classes can be generalized to other versions of constraint satisfaction. For instance the tractable class defined by BTP can be generalized to the QCSP (Gao, Yin, & Zhou, 2011)) and, as mentioned above, the class defined by the negative-transitivity property can be generalized to the Weighted CSP (Cooper & Živný, 2012)).

Any class of instances defined by a forbidden pattern is necessarily recognisable in polynomial time by a simple exhaustive search for the pattern.

2.3 Max-CSPs

Some of the tractable classes we expose belong to the Max-CSP problem. Max-CSP is a generic combinatorial optimization problem which consists in finding an assignment to the variables which satisfies the maximum number of a set of constraints. Max-CSP is NP-hard, but much research effort has been devoted to the identification of classes of instances that can be solved in polynomial time.

One classic approach consists in identifying tractable constraint languages, i.e. restrictions on the constraint relations which imply tractability. For example, if all constraints are supermodular, then Max-CSP is solvable in polynomial time, since the maximization of a supermodular function (or equivalently the minimization of a submodular function) is a well-known tractable problem in Operations Research (Orlin, 2009). Over two-element domains (Creignou, Khanna, & Sudan, 2001), three-element domains (Jonsson, Klasson, & Krokhin, 2006), and fixed-valued languages (Deineko, Jonsson, Klasson, & Krokhin, 2008), a dichotomy has been given: supermodularity is the only basic reason for tractability. However, over four-element domains it has been shown that other tractable constraint languages exist (Jonsson, Kuivinen, & Thapper, 2011). Comprehensive results for the complexity of Max-CSP for all finite constraint languages over finite domains have been given in (Thapper & Živný, 2012, 2013; Kolmogorov, 2013).

Another classic approach consists in identifying structural reasons for tractability, i.e. restrictions on the graph of constraint scopes (known as the constraint graph) which imply the existence of a polynomial-time algorithm. In the case of binary CSP the only class of constraint graphs which ensures tractability (subject to certain complexity theory assumptions) are essentially graphs of bounded tree-width (Dalmau, Kolaitis, & Vardi, 2002; Grohe, 2007). It is well known that structural reasons for tractability generalize to optimisation versions of the CSP (Bertelé & Brioshi, 1972; Dechter, 2003).

Recently, a new avenue of research has led to the discovery of tractable classes of CSP or Max-CSP instances defined by forbidding a specific (set of) subproblem(s). Novel tractable classes have been discovered by forbidding simple 3-variable subproblems (Cooper et al., 2010; Cooper & Živný, 2011b).

In a related avenue of research, other workers have defined tractable classes of binary CSP

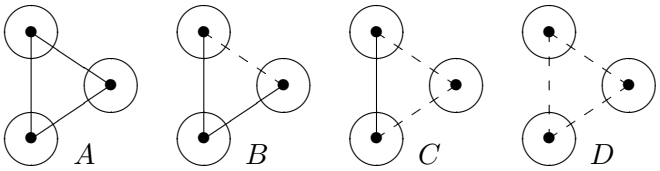


Figure 11: The subproblems A, B, C and D.

instances by excluding (sets of) induced subgraphs in the microstructure of the instance, where the microstructure of a CSP instance is the graph $\langle V, E\rangle$ with V the set of all variable-value assignments and $\{p, q\} \in E$ if and only if the pair of variable-value assignments p, q are compatible (Jégou, 1993; Cohen, 2003; Salamon & Jeavons, 2008). These microstructure-based tractable classes of CSP instances do not generalize to tractable classes of Max-CSP instances. Indeed, we are not aware of any tractable classes of Max-CSP defined exclusively in terms of the microstructure.

The complexity of classes of binary Max-CSP instances defined by local properties of (in)compatibilities have previously been characterized, but only for properties on exactly 3 assignments to 3 distinct variables (Cooper & Živný, 2011c). In the present thesis we consider classes defined by forbidding subproblems of any size and possibly involving several assignments to the same variable, thus allowing more refinement in the definition of classes of Max-CSP instances.

Certain known tractable classes can be defined by forbidding more than one subproblem. For example, in (Cooper & Živný, 2011c) it was shown that $\mathcal{F}(\{A, B\})$, $\mathcal{F}(\{B, D\})$ and $\mathcal{F}(\{A, D\})$ are all tractable (where A, B, C, D are the subproblems given in Fig. 11 and $\mathcal{F}(\{P, P'\})$ denotes the set of Max-CSP instances in which neither P nor P' occurs). The most interesting of these three tractable classes is $\mathcal{F}(\{A, B\})$ which is equivalent to maximum matching in graphs.

2.4 Variable Elimination

During the course of our work, we became interested in finding new variable elimination rules. 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 can be used in preprocessing or during search to reduce search space size. Decreasing the size of the search space is particularly welcome since constraint satisfaction is NP-complete (Dechter, 2003).

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, Conflict-Directed Back-jumping (Prosser, 1993; Chen & Van

Beek, 2001). As well as these look-back techniques it is also possible to look ahead by propagating the consequences of early decisions or of the discovered structure. Of these look-ahead techniques the most common is to maintain the local consistency property called generalized arc-consistency (GAC) (Bessière, Régin, Yap, & Zhang, 2005). This technique identifies certain values for variables that cannot possibly form part of a solution.

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

Variable elimination has been considered before in the literature. It is well known that in an arc-consistent binary CSP instance, a variable x which is constrained by only one other variable y can eliminated; by the definition of arc consistency, each assignment to y is compatible with some assignment to x. It has been observed that a more general property, called the (local) Broken Triangle Property (lBTP) (Cooper et al., 2010), if it holds at some variable, allows us to eliminate that variable. One way of stating the lBTP is that there is no pair of compatible assignments to two other variables y, z which have opposite compatibilities with two assignments to x. The closure of a binary CSP instance under the elimination of all variables that satisfy the lBTP is unique and can be found in $O(ncd^3)$ time, where n is the number of variables, c the number of constraints and d the maximum domain size, which may well prove effective when compared to the exponential cost of backtracking. 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 & Dechter, 2003). In this algorithm a variable v is not simply eliminated. Instead it is replaced by a constraint on its neighbourhood. This new constraint precisely captures those combinations of assignments to the neighbourhood of v which can be extended to a consistent assignment to v. 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. In the present thesis we restrict our attention to the identification of variable elimination strategies which do not require the addition of compensatory constraints.

In this thesis we will characterize 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. 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, Jefferson, & Miguel, 2006). Whenever a variable no longer participates in any obstruction patterns it can safely be eliminated.

Variable elimination patterns can also define tractable classes. Indeed, if a variable elimination

pattern P allows us to eliminate all variables from a given CSP instance I, then I belongs to a tractable class defined by P. Hence we are able to significantly extend the list of known tractable classes defined by forbidden patterns since many tractable patterns, like pivots (Cohen et al., 2012) and JWP (Cooper & Živný, 2012), do not allow variable elimination.

2.5 Other Simplification Operations

Variable elimination is not the only tool for simplifying the complexity of CSP instances. Simplification of instances by eliminating values from domains is at the heart of many constraint solvers via generalized arc consistency (GAC) operations. GAC eliminates domain elements that cannot be part of any solution (an assignment to all variables satisfying all constraints). Special-purpose polynomial-time GAC algorithms have been developed for many types of high-arity constraints (known as global constraints). An alternative approach is the family of elimination rules based on substitution: if all solutions in which variable v is assigned value b remain solutions when the value of variable v is changed to another value a, then the value b can be eliminated from the domain of variable v while conserving satisfiability of the instance. The most well-known polynomial-time detectable substitution operation is neighbourhood substitution (Freuder, 1991), but there are many other existing operations decreasing the size of a domain (Bessière & Debruyne, 2001; Elfe & Freuder, 1996; Bessière, Martinez, & Verfaillie, 1999).

The importance of the classic simplification operations based on consistency go beyond their use in binary CSPs, since they have been generalized to Valued CSPs (Cooper, De Givry, Sanchez, Schiex, Zytnicki, & Werner, 2010) and to global constraints (Rossi et al., 2006).

Besides elimination, merging is another possible simplification operation. One example of a merging rule is that if two assignments $\langle v, a\rangle$, $\langle v, b\rangle$ have identical compatibilities with all assignments to all other variables except concerning at most one other variable, then we can merge a and b while conserving satisfiability of the instance (Likitvivatanavong & Yap, 2013).

The last part of this thesis studies alternative notions of simplification of binary CSP instances which are based on reducing the total number of variable-value assignments. Unlike classical reduction operations, such as arc consistency, the number of variables may decrease and certain variable domains may increase in size, provided that search-space size and the total number of variable-value assignments both decrease.

3 Patterns, Tools, Reductions

In this section, we give mainly definitions and properties. First we define the general notion of what is a forbidden pattern. Then we define different kinds of patterns, for a total of three different versions. After that, we expose several operations on CSP instances. Some of them are well-known, others are new. We conclude the section by presenting the reduction to a pattern, an important tool which enables us to greatly increase the scope of our subsequent results.

3.1 What Is A Pattern?

We first define the notion of a CSP pattern. A pattern can be seen as a generalisation of a binary CSP instance; it represents 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, i.e. a pair $a = \langle v, d \rangle$ where d is in the domain of variable v. 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 an assignment $\langle v, d \rangle$ is simply the variable v and the label of an edge between two vertices describes the compatibility of the pair of assignments corresponding to the pair of vertices.

Definition 1 (pattern). *A* pattern*, or* flat 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 $var : A \to V$.*
- *a set $E \subseteq \binom{A}{2}$ of edges (unordered pairs of elements of A) such that $\{a, b\} \in E \Rightarrow var(a) \neq var(b)$.*
- *a Boolean-valued compatibility function $cpt : E \to \{F, T\}$, where for notational simplicity, we write $cpt(a, b)$ instead of $cpt(\{a, b\})$.*

Definition 2 (CSP instance). *A* binary CSP instance *is a pattern $\langle V, A, var, E, cpt \rangle$ such that $E = \{\{a, b\} \mid (a, b) \subseteq A \times A, var(a) \neq var(b)\}$ (i.e. 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 consistent set of assignments to all the variables, that is a* solution *$\overline{A} \subseteq A$ such that $|\overline{A}| = |V|$, ($\forall a, b \in \overline{A}$, $var(a) \neq var(b)$) and ($\forall e \in \binom{\overline{A}}{2} \cap E$, $cpt(e) = T$)?*

For a pattern $P = \langle V, A, var, E, cpt \rangle$ and a variable $v \in V$, we use A_v to denote the set of assignments $\{a \in A \mid var(a) = v\}$.

Definition 3 (constraint). *The* constraint *on variables $v_1, v_2 \in V$ is the pattern $\langle \{v_1, v_2\}, A_{12}, var|_{A_{12}}, E_{12}, cpt|_{E_{12}} \rangle$ where $A_{12} = A_{v_1} \cup A_{v_2}$ and $E_{12} = \{\{a, b\} \mid a \in A_{v_1}, b \in A_{v_2}\} \cap E$.*

Definition 4 (compatible points, compatibility edge). *If $cpt(a, b) = T$ then the two assignments (points) a, b are* compatible *and $\{a, b\}$ is a* compatibility edge*; if $cpt(a, b) = F$ then the two assignments a, b are* incompatible *and $\{a, b\}$ is an* incompatibility edge*.*

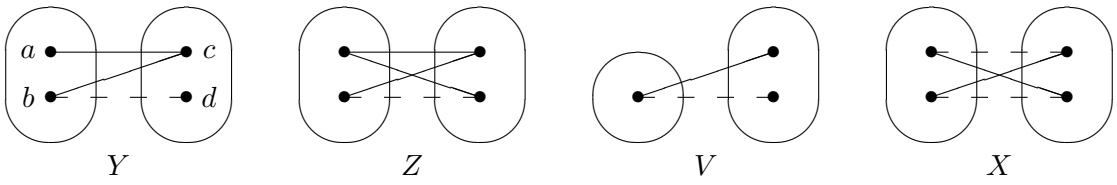


Figure 12: Four patterns.

In a pattern, the compatibility of a pair of points a, b such that $var(a) \neq var(b)$ and $(a, b) \notin E$ is *undefined*. A pattern can be viewed as a compact means of representing the set of all instances obtained by arbitrarily specifying the compatibility of such 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 $\langle V, A, var, E, cpt \rangle$, 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 and the set $\{(d_1, d_2) \mid (\langle v_1, d_1 \rangle, \langle v_2, d_2 \rangle) \in A_{v_1} \times A_{v_2} \wedge cpt(\langle v_1, d_1 \rangle, \langle v_2, d_2 \rangle) = T\}$ of compatible pairs of values that can be assigned to two variables $v_1, v_2 \in V$ the *constraint relation* on v_1, v_2.

Definition 5 (trivial constraint). *The constraint between variables v_1 and v_2 in a CSP instance is* non-trivial *if there is at least one incompatible pair of assignments, i.e. $a \in A_{v_1}$ and $b \in A_{v_2}$ such that $cpt(a, b) = F$.*

Definition 6 (constraint graph). *The* constraint graph *of an instance $\langle V, A, var, E, cpt \rangle$ is $\langle V, H \rangle$, 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 7 (occurence in a pattern). *We say that a pattern P* occurs *in a pattern P' (or that 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 is a sub-pattern of Q (and Q an extension of P): if $P = \langle V_P, A_P, var_P, E_P, cpt_P \rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q \rangle$, then $V_P \subseteq V_Q$, $A_P \subseteq A_Q$, $var_P = var_Q|_{A_P}$, $E_P \subseteq E_Q$, $cpt_P = cpt_Q|_{E_P}$.*

merging *Merging two points in P transforms P into Q: if $P = \langle V_P, A_P, var_P, E_P, cpt_P \rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q \rangle$, then $\exists a, b \in A_P$ such that $var_P(a) = var_P(b)$ and $\forall c \in A_P$ such that $\{a, c\}, \{b, c\} \in E_P$, $cpt_P(a, c) = cpt_P(b, c)$. Furthermore, $V_P = V_Q$, $A_Q = A_P \setminus \{b\}$, $var_Q = var_P|_{A_Q}$, $E_Q = (E_P \setminus \{\{b, x\} \mid \{b, x\} \in E_P\}) \cup \{\{a, x\} \mid \{b, x\} \in E_P\}$ and $cpt_Q(a, x) = cpt_Q(b, x)$ if $\{b, x\} \in E_P$, $cpt_Q(e) = cpt_P(e)$ for all other $e \in E_Q$.*

Consider the four patterns shown in Figure 12. 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, Y consists of 4 points $a, b \in A_{v_0}$, $c, d \in A_{v_1}$ such that $cpt(a, c) = cpt(b, c) = T$ and $cpt(b, d) = F$. Y occurs in Z since Z is an extension of Y. Y occurs in V since V can be obtained from Y by merging points a, b. Y also occurs in X by a merging followed by an extension.

Notation: Let P be a CSP pattern. We use $\mathrm{CSP}(\overline{P})$ to denote the set of binary CSP instances Q in which P does not occur.

Definition 8 (tractable pattern). *A pattern P is* intractable *if $\mathrm{CSP}(\overline{P})$ is NP-complete. It is* tractable *if there is a polynomial-time algorithm to solve $\mathrm{CSP}(\overline{P})$.*

Definition 9 (mergeable). *A pattern P is* mergeable (non-mergeable) *if P can (cannot) be transformed into a pattern $Q \neq P$ by merging.*

It is worth observing that, in a class of CSP instances defined by forbidding a pattern, there is no bound on the size of domains. Recall, however, that CSP instances have finite domains since the set of all possible assignments is assumed to be given in extension as part of the input.

Clearly, all classes of CSP instances $\mathrm{CSP}(\overline{P})$ defined by forbidding a pattern are hereditary: $I \in \mathrm{CSP}(\overline{P})$ and $I' \subseteq I$ (in the sense that I is an extension of I', according to Definition 7) together imply that $I' \in \mathrm{CSP}(\overline{P})$. Furthermore, if $I \in \mathrm{CSP}(\overline{P})$ and I' is isomorphic to I, then $I' \in \mathrm{CSP}(\overline{P})$. Forbidding a pattern therefore only allows us to define hereditary classes closed under arbitrary permutations of variable domains.

3.2 Different Kinds of Patterns

In this subsection we consider different ways of defining a class of CSP instances by forbidding patterns.

3.2.1 About Quantified Patterns

Definition 10 (quantified pattern). *A* quantified pattern *$\langle V, A, var, E, cpt, v\rangle$ is a pattern $P = \langle V, A, var, E, cpt\rangle$ to which we add an existential quantifier on a distinguished variable $v \in V$.*

Definition 11 (flat simplified pattern). *If $P = \langle V, A, var, E, cpt, v\rangle$ is a quantified pattern, then $P' = \langle V, A, var, E, cpt\rangle$ is the* flat simplified pattern *of P.*

We now give versions of the definitions of extension, merging and occurrence generalized to quantified patterns.

Definition 12 (occurence in a quantified pattern). *We say that a quantified pattern P* occurs *in a quantified pattern P' (or that P'* contains *P) if P' is isomorphic to a quantified pattern Q in the transitive closure of the following two operations (extension and merging) applied to P:*

extension *P is a sub-pattern of Q (and Q an extension of P): if $P = \langle V_P, A_P, var_P, E_P, cpt_P, v_P\rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, v_Q\rangle$, then $V_P \subseteq V_Q$, $A_P \subseteq A_Q$, $var_P = var_Q|_{A_P}$, $E_P \subseteq E_Q$, $cpt_P = cpt_Q|_{E_P}$, and $v_Q = v_P$. We give an example in Figure 13.*

merging *Merging two points in P transforms P into Q: if $P = \langle V_P, A_P, var_P, E_P, cpt_P, v_P\rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, v_Q\rangle$, then $\exists a, b \in A_P$ such that $var_P(a) = var_P(b)$ and $\forall c \in A_P$ such that $\{a, c\}, \{b, c\} \in E_P$, $cpt_P(a, c) = cpt_P(b, c)$. Furthermore, $V_P = V_Q$, $A_Q = A_P \setminus \{b\}$, $var_Q = var_P|_{A_Q}$, $E_Q = (E_P \setminus \{\{b, x\} \mid \{b, x\} \in E_P\}) \cup \{\{a, x\} \mid \{b, x\} \in E_P\}$, $cpt_Q(a, x) = cpt_Q(b, x)$ if $\{b, x\} \in E_P$, $cpt_Q(e) = cpt_P(e)$ otherwise, and $v_Q = v_P$. We give an example in Figure 14.*

Figure 13: Example of extension of a quantified pattern P to produce the quantified pattern Q.

Figure 14: Example of merging in a quantified pattern P to produce the quantified patten Q.

It follows from Definition 12 that an occurrence of a quantified pattern P in a quantified pattern Q can also be viewed as the existence of an *occurrence-function* $f : A_P \to A_Q$ such that

1. $\forall a, b \in A_P$, $var_Q(f(a)) = var_Q(f(b))$ if and only if $var_P(a) = var_P(b)$.
2. $\forall a, b \in A_P$ such that $\{a, b\} \in E_P$, $\{f(a), f(b)\} \in E_Q$ and $cpt_Q(f(a), f(b)) = cpt_P(a, b)$.
3. $\forall a \in A_{v_P}$, $f(a) \in A_{v_Q}$.

Definition 13 (occurence in an instance). *We say that a quantified pattern $P = \langle V, A, var, E, cpt, v\rangle$* occurs in *a CSP instance I if its flat simplified pattern $P' = \langle V, A, var, E, cpt\rangle$ occurs in I.*

3.2.2 About Existential Patterns

Definition 14 (existential pattern). *An* existential pattern *is a pattern P with a set of existential points $e \subseteq A_v$, with $e \neq \emptyset$, for some distinguished variable v.*

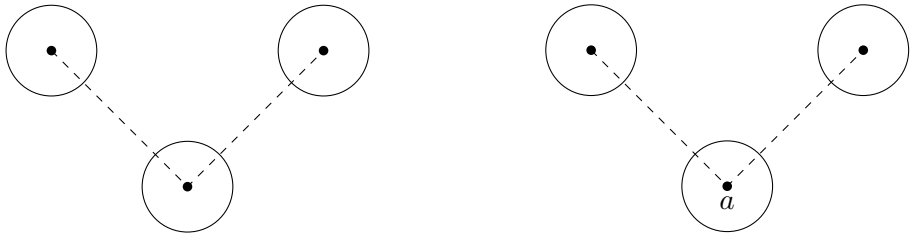
Thus, an existential pattern $\langle V, A, var, E, cpt, e\rangle$ is a pattern $\langle V, A, var, E, cpt\rangle$ to which we add a set of existential points $e \subseteq A_v$ for some distinguished variable $v \in V$.

Definition 15 (quantified simplified pattern). *If $P = \langle V, A, var, E, cpt, e\rangle$ is an existential pattern with some $v \in V$ such that $e \subseteq A_v$, then $P' = \langle V, A, var, E, cpt, v\rangle$ is the* quantified simplified pattern *of P.*

Forbidding an existential pattern P means that for all variables x in the instance I, there is an injective occurence function $f_x : e \to A_x$ such that there is no occurrence of P in I in which each $p \in e$ maps to $f_x(p)$. Forbidding an existential version Q of a pattern P defines a much larger

$1I$ $\exists 1I$

Figure 15: A simple pattern $1I$ and an existential version $\exists 1I$ of the same pattern.



V− V−Middle

Figure 16: A pattern V− and an existential version V−Middle of the same pattern.

class CSP($\overline{Q}$) than CSP($\overline{P}$). Although we first introduced existential patterns to define variable elimination rules (Cohen et al., 2013), they also can define new tractable classes.

As a simple example, consider the pattern $1I$ and its existential version $\exists 1I$ shown in Figure 15. In this figure, the label a on one of the points indicates that a is an existential point. This convention will be used for all subsequent figures representing an existential pattern. Forbidding $1I$ in an instance means that all points are compatible with all other points in the instance, whereas forbidding $\exists 1I$ imposes the less restrictive assumption that for each variable x there exists some point $f_x(a) \in A_x$ which is compatible with all other points of the instance.

To see why existential patterns are not too trivial a tool, consider as a slightly more elaborate example the pattern V− and its existential version V−Middle shown in Figure 16. Forbidding V− in an instance means that all points in the instance are incompatible with points in at most one other variable, whereas forbidding V−Middle imposes the less restrictive assumption that for each variable x there exists some point $f_x(a) \in A_x$ which is incompatible with points in at most one other variable. From Theorem 1, we know that the set of CSP instances in which we forbid the pattern V− is tractable. Actually, if we only consider arc-consistent instances, there even exists a linear time algorithm which can find a solution in any such instance. However, as we show later in Lemma 29, the set of instances in which we forbid the pattern V−Middle is NP-Complete, even when only considering arc-consistent instances.

We now give versions of the definitions of extension, merging, occurrence and tractability generalized to existential patterns.

Definition 16 (occurence in an existential pattern). *We say that an existential pattern P* occurs *in an existential pattern P' (or that P'* contains *P) if P' is isomorphic to an existential pattern Q in the transitive closure of the following two operations (extension and merging) applied to P:*

extension *P is a sub-pattern of Q (and Q an extension of P): if $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P \rangle$ and*

P with $e_P = \{a\}$ Q with $e_Q = \{a\}$

Figure 17: Example of extension of an existential pattern P to produce the existential pattern Q.

P with $e_P = \{a, b\}$ Q with $e_Q = \{a\}$

Figure 18: Example of merging in an existential pattern P to produce the existential patten Q.

$Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, e_Q\rangle$, then $V_P \subseteq V_Q$, $A_P \subseteq A_Q$, $var_P = var_Q|_{A_P}$, $E_P \subseteq E_Q$, $cpt_P = cpt_Q|_{E_P}$, and $e_P \subseteq e_Q$. We give an example in Figure 17.

merging *Merging two points in P transforms P into Q: if $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P\rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, e_Q\rangle$, then $\exists a, b \in A_P$ such that $var_P(a) = var_P(b)$, $a \in e_P \Leftrightarrow b \in e_P$ and $\forall c \in A_P$ such that $\{a, c\}, \{b, c\} \in E_P$, $cpt_P(a, c) = cpt_P(b, c)$. Furthermore, $V_P = V_Q$, $A_Q = A_P \setminus \{b\}$, $var_Q = var_P|_{A_Q}$, $E_Q = (E_P \backslash \{\{b, x\} \mid \{b, x\} \in E_P\}) \cup \{\{a, x\} \mid \{b, x\} \in E_P\}$, $cpt_Q(a, x) = cpt_Q(b, x)$ if $\{b, x\} \in E_P$, $cpt_Q(e) = cpt_P(e)$ otherwise, and $e_Q = e_P \setminus \{b\}$. We give an example in Figure 18.*

It follows from Definition 16 that an occurrence of an existential pattern P in an existential pattern Q can also be viewed as the existence of an *occurrence-function* $f : A_P \to A_Q$ such that

1. $\forall a, b \in A_P$, $var_Q(f(a)) = var_Q(f(b))$ if and only if $var_P(a) = var_P(b)$.
2. $\forall a, b \in A_P$ such that $\{a, b\} \in E_P$, $\{f(a), f(b)\} \in E_Q$ and $cpt_Q(f(a), f(b)) = cpt_P(a, b)$.
3. $\forall a \in A_P$ such that $a \in e_P$, $f(a) \in e_Q$.

Definition 17 (occurence on a set of points). *Let $I = \langle V, A, var, E, cpt\rangle$ be a CSP instance. Let v be a variable in V. Let $S \subseteq A_v$. Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P\rangle$ be an existential pattern.*

If P occurs in the existential pattern $\langle V, A, var, E, cpt, S\rangle$ with occurrence-function $f : A_P \to A$, such that $f|_{e_P}$ is a bijection, then we say that P occurs on S via $f|_{e_P}$. *(If S is a singleton $\{a\}$, then to simplify notation we simply say that P occurs on a).*

Definition 18 (occurence in an instance). *If $I = \langle V, A, var, E, cpt\rangle$ is a CSP instance, then an existential pattern $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P\rangle$* occurs *in I (and I* contains *P) if $\exists v \in V$ with $|A_v| \geq |e_P|$ such that for all subsets S of A_v with $|S| = |e_P|$ and all bijections $g : e_P \to S$, P occurs on S via g. Conversely, P* does not occur *in I if $\forall v \in V$ with $|A_v| \geq |e_P|$, there is an injective occurence function $g : e_P \to A_v$ such that P does not occur on $g(e_P)$ via g.*

It is worth pointing out that we will show later that when $e_P \geq 2$ for some non-trivial existential pattern P, the set of CSP instances not containing P is NP-Complete. This explains why no such pattern appears on the tractability side of the main result of Subsection 4.1.5 and why for most of the thesis we only need to consider existential patterns P in which e_P is a singleton.

Suppose that an existential pattern P does not occur in an instance I since for all variables v, there is a subset S_v of A_v and a bijection $g : e_P \rightarrow S_v$ such that P does not occur on S_v via g. Establishing arc consistency in I may eliminate some of the assignments in the sets S_v with the consequence that P may now occur in the arc-consistent version of I. Since arc consistency is a basic filtering operation applied by all constraint solvers to reduce the size of variable domains, we choose to study only arc consistent CSP instances.

Notation: Let P be an existential pattern. We use $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ to denote the set of arc-consistent binary CSP instances I in which P does not occur.

Definition 19 (tractable existential pattern). *An existential pattern P is* intractable *if $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ is NP-complete. It is* tractable *if there is a polynomial-time algorithm to solve $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$.*

3.3 Operations for CSP Instances

3.3.1 Classical Operations

This subsection describes polynomial-time simplification operations on CSP instances. Assuming that these operations have been applied facilitates the proof of tractability of many patterns.

Definition 20 (elimination of a single-valued variable). *Let $\langle V, A, var, E, cpt\rangle$ be a CSP instance. If for some variable v, A_v 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 as well as all assignments b which are incompatible with a.*

Definition 21 (arc consistency). *Given a CSP instance $\langle V, A, var, E, cpt\rangle$,* arc consistency *consists in eliminating from A all assignments a for which there is some variable $v \neq var(a)$ in V such that $\forall b \in A_v$, $cpt(a, b) = F$.*

The notion of arc consistency can be traced back to the beginnings of Artificial Intelligence (Waltz, 1975; Mackworth, 1977) and there are well known optimal algorithms to establish it (Henderson & Mohr, 1986; Bessière et al., 2005).

Definition 22 (neighborhood substitution). *Given a CSP instance $\langle V, A, var, E, cpt\rangle$, if $var(a) = var(b)$ and for all variables $v \neq var(a)$, $\forall c \in A_v$, $cpt(a, c) = T \Rightarrow cpt(b, c) = T$, then we can eliminate a from A by* neighborhood substitution*, since in any solution in which a appears, we can replace a by b.*

Neighborhood substitution was first introduced in (Freuder, 1991).

Establishing arc consistency and eliminating single-valued variables until convergence produces a unique result, and the result of applying neighborhood substitution operations until convergence is unique modulo isomorphism (Cooper, 1997). Since removing points or variables from

a CSP instance does not introduce any pattern, none of these three operations when applied to an instance in $\mathrm{CSP}(\overline{P})$ can introduce the forbidden pattern P.

3.3.2 Our Operations

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. The semantics of the two fusion operations defined below will become clear with the explanations given in the proof of Lemma 1.

Definition 23 (simple fusion). *Consider a CSP instance $\langle V, A, var, E, cpt\rangle$ with $v_1, v_2 \in V$. Suppose that there is a* fusion function *$f : A_{v_1} \to A_{v_2}$, 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 $\langle V', A', var', E', cpt'\rangle$ defined by:*

- $V' = (V \setminus \{v_1, v_2\}) \cup \{v\}$.
- $A' = A \setminus A_{v_2}$.
- $var'(u) = var(u)$ *for all* $u \in A' \setminus A_{v_1}$ *and* $var'(u) = v$ *for all* $u \in A_{v_1}$.
- $E' = \{\{p, q\} \in \binom{A'}{2} \mid var'(p) \neq var'(q)\}$.
- $cpt'(p, q) = cpt(p, q)$ *if* $p, q \in A' \setminus A_{v_1}$, $cpt'(u, q) = cpt(u, q) \wedge cpt(f(u), q)$ *for all* $u \in A_{v_1}$ *and all* $q \in A' \setminus A_{v_1}$.

Definition 24 (complex fusion). *Consider a CSP instance $\langle V, A, var, E, cpt\rangle$ with $v_1, v_2 \in V$ and a* hinge value *$a \in A_{v_1}$. Suppose that there is a* fusion function *$f : A_{v_1} \setminus \{a\} \to A_{v_2}$, such that $\forall u \in A_{v_1} \setminus \{a\}$, 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 $\langle V', A', var', E', cpt'\rangle$ defined by:*

- $V' = (V \setminus \{v_1, v_2\}) \cup \{v\}$.
- $A' = A \setminus \{a\}$.
- $var'(u) = var(u)$ *for all* $u \in A' \setminus (A_{v_1} \cup A_{v_2})$ *and* $var'(u) = v$ *for all* $u \in (A_{v_1} \setminus \{a\}) \cup A_{v_2}$.
- $E' = \{\{p, q\} \in \binom{A'}{2} \mid var'(p) \neq var'(q)\}$,
- $cpt'(p, q) = cpt(p, q)$ *if* $p, q \in A' \setminus (A_{v_1} \cup A_{v_2})$, $cpt'(u, q) = cpt(u, q) \wedge cpt(f(u), q)$ *for all* $u \in A_{v_1} \setminus \{a\}$ *and all* $q \in A' \setminus (A_{v_1} \cup A_{v_2})$, $cpt'(p, q) = cpt(a, q) \wedge cpt(p, q)$ *for all* $p \in A_{v_2}$ *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, p) where $p \in A_{v_2}$ or of the form $(u, f(u))$ where $u \in A_{v_1} \setminus \{a\}$. So if I is solvable, then I' is solvable. To complete the proof, it suffices to observe that in A' we use $p \in A_{v_2}$ to represent the pair of assignments (a, p) and $u \in A_{v_1} \setminus \{a\}$ to represent $(u, f(u))$. So if I' is solvable, then I is solvable. □

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 $I \in \mathrm{CSP}(\overline{P})$, for some pattern P, a fusion operation will only be useful if it does not introduce the forbidden pattern P.

3.4 Reduction to a Pattern

3.4.1 Reduction in the Flat Case

In a pattern $P = \langle V_P, A_P, var_P, E_P, cpt_P \rangle$, a point a which is linked by a single compatibility edge to the rest of P is known as a *dangling point*. If an arc consistent instance $I = \langle V, A, var, E, cpt \rangle$ with $|V| \geq |V_P|$ 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, since arc consistency is a polynomial-time operation which cannot introduce a forbidden pattern, to decide tractability we only need to consider patterns without dangling points.

Definition 25 (reduction to a flat pattern). *We say that a pattern P can be* reduced *to a pattern Q, and that Q is a* reduction *of P, if $Q = P$ or if Q is in the transitive closure of the two operations merging and dp-elimination applied to P, where dp-elimination is the following operation:*

dp-elimination *Eliminating a dangling point, its corresponding compatibility edge and its corresponding variable v (if A_v becomes empty) from P transforms P into Q. We give an example in Figure 19.*

Figure 19: Example of dp-elimination.

Lemma 2. *Let $P = \langle V_P, A_P, var_P, E_P, cpt_P \rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q \rangle$ be two patterns, such that P can be reduced to a sub-pattern of Q. Let $I = \langle V, A, var, E, cpt \rangle$ be a CSP instance satisfying arc consistency, with $|V| \geq |V_P|$. If Q occurs in I, then P also occurs in I.*

Proof. By definition, reduction is a transitive relation. Therefore, by induction, it suffices to prove the result for each of the individual operations: merging and dp-elimination. We suppose Q occurs in I. If merging two points a and b in P transforms it into a sub-pattern Q' of Q, then P actually covers two different patterns: the one where a and b are different points, and the one where a and b are the same point. The latter pattern is Q'. So the set of instances containing Q is a subset of the set of instances containing (at least one of the two versions of) P and we have the result. If adding a dangling point and its corresponding compatibility edge to a sub-pattern Q' of Q transforms Q' into P, then since I satisfies arc consistency P also occurs in I. □

The following corollary follows immediately from the fact that arc consistency can be established in polynomial time.

Corollary 1. *Let P and Q be two patterns, such that P can be reduced to a sub-pattern of Q. Then*

- *If Q is tractable, then P is tractable.*
- *If P is intractable, then Q is intractable.*

It follows that in order to find new tractable classes we only need to study those patterns that cannot be reduced to a sub-pattern of a known tractable pattern and that do not have as a sub-pattern a reduction of a known intractable pattern.

3.4.2 Reduction in the Quantified Case

In a quantified pattern $P = \langle V_P, A_P, var_P, E_P, cpt_P, v_P \rangle$, a point a which is linked by a single compatibility edge to the rest of P is known as a *dangling point*. If an arc consistent instance $I = \langle V, A, var, E, cpt \rangle$ with $|V| \geq |V_P|$ 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.

Definition 26 (reduction to a quantified pattern). *We say that a quantified pattern P can be* reduced *to a quantified pattern Q, and that Q is a* reduction *of P, if $Q = P$ or if Q is in the transitive closure of the two operations merging and dp-elimination applied to P, where dp-elimination is the following operation:*

dp-elimination *Eliminating a dangling point, its corresponding compatibility edge and its corresponding variable v (if A_v becomes empty and $v \neq v_P$) from P transforms P into Q. We give an example in Figure 20.*

Lemma 3. *Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, v_P \rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, v_Q \rangle$ be two quantified patterns, such that P can be reduced to a sub-pattern of Q. Let $I = \langle V, A, var, E, cpt \rangle$ be a CSP instance satisfying arc consistency, with $|V| \geq |V_P|$. If Q occurs in I, then P also occurs in I.*

Proof. The result can be deduced from Definition 13 and Lemma 2. □

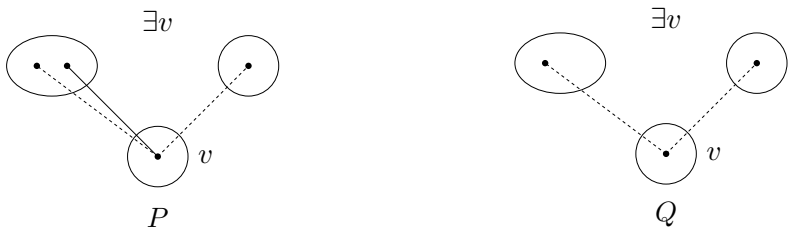


Figure 20: Example of dp-elimination in a quantified pattern P.

Figure 21: Example of dp-elimination in an existential pattern P to produce the existential pattern Q.

3.4.3 Reduction in the Existential Case

The notions of dp-elimination and reduction can be similarly used in the context of existential patterns. In an existential pattern $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P \rangle$, a point $p \notin e_P$ which is linked by a single compatibility edge to the rest of P is known as a *dangling point*. If an arc consistent instance $I = \langle V, A, var, E, cpt \rangle$ with $|V| \geq |V_P|$ does not contain the existential pattern P then it does not contain the pattern P' which is equivalent to P in which the dangling point p and the corresponding compatibility edge have been deleted. Thus, to decide the tractability of $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ we only need to consider patterns P without dangling points.

Definition 27 (reduction to an existential pattern). *We say that an existential pattern P can be* reduced *to an existential pattern Q, and that Q is a* reduction *of P, if $Q = P$ or if Q is in the transitive closure of the two operations merging and dp-elimination applied to P, where dp-elimination is the following operation:*

dp-elimination *Eliminating a dangling point, its corresponding compatibility edge and its corresponding variable v (if A_v becomes empty) from P transforms P into Q. We give an example in Figure 21.*

Lemma 4. *Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P \rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, e_Q \rangle$ be two existential patterns, such that P is a sub-pattern of Q. Let $I = \langle V, A, var, E, cpt \rangle$ be an arc-consistent CSP instance. If Q occurs in I, then P also occurs in I.*

Proof. Suppose that Q occurs in I. So $\exists v \in V$ such that Q occurs on all subsets S of A_v of size $|S| = |e_Q|$ and for all bijections $g : e_Q \to S$. Let T be any subset of A_v of size $|e_P|$ and let $h : e_P \to T$ be any bijection. We have to show that P occurs in I on T via h.

Let S be any subset of A_v of size $|e_Q|$ such that $T \subseteq S$ and let $g : e_Q \to S$ be any bijection such that $g|_{e_P} = h$. We know that Q occurs in the existential pattern $\langle V, A, var, E, cpt, S \rangle$ with an

occurrence-function f such that $f|_{e_Q} = g$. Since P is a sub-pattern of Q, P occurs in the existential pattern $\langle V, A, var, E, cpt, S \cap f(e_P)\rangle$ with the occurrence-function $f|_{A_P}$. Since $e_P \subseteq e_Q$ by the definition of a sub-pattern, we have $f|_{e_P} = g|_{e_P} = h$. Thus P occurs in I on $T = h(e_P)$ via h and we are done. □

Lemma 5. *Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P\rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cpt_Q, e_Q\rangle$ be two existential patterns, such that P can be reduced to a sub-pattern of Q. Let $I = \langle V, A, var, E, cpt\rangle$ be an arc-consistent CSP instance with $|V| \geq |V_P|$. If Q occurs in I, then P also occurs in I.*

Proof. By definition, reduction is a transitive relation. Therefore, by induction, it suffices to prove the result for each of the individual operations: merging and dp-elimination.

If merging two points a and b in P transforms it into a sub-pattern Q' of Q, then P actually covers two different patterns: the one where a and b are different points, and the one where a and b are the same point. The latter pattern is Q' which occurs in I, by Lemma 4, since it is a sub-pattern of Q. So the set of instances containing Q is a subset of the set of instances containing (at least one of the two versions of) P and we have the result.

We now suppose that eliminating a dangling point $c \in v_c$, with $v_c \in V_P$, and its corresponding compatibility edge from P transforms P into a sub-pattern Q' of Q, where
$Q' = \langle V_{Q'}, A_{Q'}, var_{Q'}, E_{Q'}, cpt_{Q'}, e_{Q'}\rangle$. Since c is a dangling point, then from the definition of dp-elimination we know that $c \notin e_P$. So $e_{Q'} = e_P$. Let d be the point such that $\{c, d\}$ is the compatibility edge eliminated from P to produce Q'. Since Q' is a sub-pattern of Q, by Lemma 4, we know that Q' occurs in $I = \langle V, A, var, E, cpt\rangle$. So $\exists v \in V$ such that for all $S \subseteq A_v$ with $|S| = |e_{Q'}|$ and for all bijections $g : e_{Q'} \to S$, Q' occurs on S via g. Let f be the corresponding occurrence-function. Since $e_P = e_{Q'}$, it suffices to show that P also occurs on S via g. If $v_c \in V_{Q'}$, then let $v'_c = var(f(v_c))$ be the variable in I corresponding to v_c in this occurrence of Q'. If $v_c \notin V_{Q'}$ (due to being eliminated during dp-elimination), then $|V_{Q'}| < |V_P| \leq |V|$, and so we can set $v'_c \in V$ to be a variable of I not corresponding to any variable in $V_{Q'}$ in this occurrence of Q'. In both cases, since I satisfies arc consistency, there is a point $c' \in v'_c$ compatible with $f(d)$. We can thus extend f to an occurrence-function f' of P in I by setting: $f'(c) = c'$, and $f'(p) = f(p)$ for all $p \in A_P \setminus \{c\} = A_{Q'}$. Hence P also occurs on S via g, since f and f' are identical on e_P, which completes the proof. □

The following corollary follows immediately from the fact that arc consistency can be established in polynomial time.

Corollary 2. *Let P and Q be two existential patterns, such that P can be reduced to a sub-pattern of Q. Then*

- *If Q is tractable, then P is tractable.*
- *If P is intractable, then Q is intractable.*

It follows that in order to find new tractable classes we only need to study those existential patterns that cannot be reduced to a sub-pattern of a known tractable existential pattern and that do not have as a sub-pattern a reduction of a known intractable existential pattern.

3.4.4 Reduction to a Different Kind of Pattern

Let $P = \langle V, A, var, E, cpt \rangle$ be a flat pattern and let $P' = \langle V, A, var, E, cpt, v \rangle$ be a quantified pattern such that P is the flat simplified pattern of P'. From Definition 13, we know that if P' occurs in a CSP instance I, then P also occurs in I. Therefore, we can say that P can be reduced to P' without altering the fundamental properties of the reduction to a pattern.

Similarly, let $P = \langle V, A, var, E, cpt, v \rangle$ be a quantified pattern and let $P' = \langle V, A, var, E, cpt, e \rangle$ be an existential pattern such that P is the quantified simplified pattern of P'. From Definitions 13 and 18, we know that if P' occurs in a CSP instance I, then P also occurs in I. Therefore, we can say that P can be reduced to P' without altering the fundamental properties of the reduction to a pattern.

4 Tractable Classes

This section is the first chapter in which we present our results. It focuses on complexity. We give several tractable classes, and also prove some NP-Completeness results. There are two main dichotomies, the first one is about patterns on two constraints, the other one is about Max-CSP subproblems. The last part of the section exposes a lot of miscellaneous complexity results for patterns on three variables.

4.1 Dichotomy for Forbidden Patterns on Two Constraints

4.1.1 Flat Patterns on One Constraint

In this subsection we prove a dichotomy for patterns composed of a single constraint. We also prove some results concerning 1-constraint patterns that are essential for the proof of the 2-constraint dichotomy given in Section 4.1.3.

Lemma 6. *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). To prove the lemma it suffices to give a polynomial reduction from SAT1 to CSP($\overline{P}$). 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 $l_i^1 \vee \cdots \vee l_i^{c_i}$. 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 $\overline{v_i}$ in $A_{v'_i}$.
- $\forall v'_i$ with $n + 1 \leq i \leq n + k$, c_{i-n} points $l_{i-n}^1, \ldots, l_{i-n}^{c_{i-n}}$ in $A_{v'_i}$.
- $\forall 1 \leq i \leq k$, $\forall 1 \leq j \leq c_i$, an incompatibility edge between the point $l_i^j \in A_{v'_{n+i}}$ and the point in $A_{v'_1}, \ldots, A_{v'_n}$ corresponding to the literal $\overline{l_i^j}$.

A solution to I' consists of a set of literals assigned `true` in a solution s to I together with for each clause a literal from this clause which is assigned `true` in s. Therefore, 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, this constraint C 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, there is only one incompatibility edge in C. So I' does not contain the pattern P. So we have reduced SAT1 to CSP($\overline{P}$), as required. □

Definition 28 (explicitely compatible). *Given a pattern $P = \langle V, A, var, E, cpt\rangle$, 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).*

Lemma 7. *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.*

Proof. Suppose we have a pattern P such that there are two points a and b with $var(a) = var(b)$ such that neither a nor b is explicitly incompatible. So no point in the pattern is incompatible with either a or b. Hence, we can merge a and b, which is a contradiction. □

Let Z be the pattern on two variables v and v', shown in Figure 12, 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.

Lemma 8. *Z is intractable.*

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

For $s, t \in \{1, 2, 3\}$, define the relation $R_{s,t} \subseteq \{1, 2, 3\}^2$ by

$$R_{s,t} \;=\; \{\langle u, v\rangle | (u = s \wedge v = t) \vee (u \neq s \wedge v \neq t)\}$$

It is easy to verify that $R_{s,t}$ does not contain the pattern Z. Consider the 5-variable gadget with variables v_i, v_j, u_1, u_2, u_3, each with domain $\{1, 2, 3\}$, and with constraint relations $R_{k,k}$ on variables (v_i, u_k) $(k = 1, 2, 3)$ and constraint relations $R_{1+(k \bmod 3),k}$ on variables (u_k, v_j) $(k = 1, 2, 3)$. The joint effect of these six constraints is simply to impose the constraint $v_i \neq v_j$. Any instance $\langle V, E\rangle$ of 3-COLORING, with $V = \{1, \ldots, n\}$, can be reduced to an instance of CSP($\overline{Z}$) 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. □

Let $1I$ be the pattern on two variables v and v' with points $a \in A_v$ and $b \in A_{v'}$ such that a and b are incompatible. $1I$ is a trivial tractable pattern, because any CSP instance not containing $1I$ contains only trivial constraints.

Lemma 9. *Let P be a pattern on one constraint. Then either P is reducible to a sub-pattern of $1I$, and thus is tractable, or P is intractable.*

Proof. Let P be a pattern on one constraint between two variables v and v'. From Lemma 6, we know that if P has two non mergeable incompatibility edges, then P is intractable. If there is no incompatibility edge at all in P, then P is reducible by merging and/or dp-elimination to the empty pattern, which is a sub-pattern of $1I$. We therefore suppose that there is exactly one incompatibility edge in P, or that P can be reduced by merging to a pattern with only one incompatibility edge. Let $a \in A_v$ and $b \in A_{v'}$ be the points defining this edge. From Lemma 7, we know that we only

need to consider at most one other point $c \neq a$ in A_v and at most one other point $d \neq b$ in $A_{v'}$. If all three edges $\{a, d\}$, $\{c, b\}$ and $\{c, d\}$ are compatibility edges, then P is intractable from Lemma 8. If only two or less of these edges are compatibility edges, then P is reducible by merging and/or dp-elimination to $1I$. So we have the lemma. □

4.1.2 Flat Patterns on Two Constraints

Lemma 10. *Let P be a pattern composed of two separate one-constraint patterns: P_1 on variables v_0, v_1 and P_2 on variables v_2, v_3, where all four variables are distinct. Then*

1. *If either P_1 or P_2 is intractable, then P is intractable too.*
2. *If both P_1 and P_2 are tractable, then P is tractable.*

Proof. 1. P_1 and P_2 are sub-patterns of P. So if one of them is intractable, then P is intractable too, by Corollary 1.

2. Suppose that both P_1 and P_2 are tractable. So there are two polynomial algorithms A_1 and A_2 which solve CSP($\overline{P_1}$) and CSP($\overline{P_2}$), respectively. Let I be a CSP instance such that P does not occur in I. If P_1 does not occur in I then this can be detected in polynomial time and I can be solved by A_1. If P_1 occurs on variables u, v in I_1, then for each assignment of values to the pair of variables u, v, the resulting instance I' cannot contain P_2 and hence can be solved by A_2.

□

The following lemma concerns a pattern in which some structure is imposed on domain elements. It is essential for our two-constraint dichotomy.

Let $2V$ be the pattern on three variables v_0, v_1 and v_2 with three points $a, b, c \in A_{v_1}$, three points $d, e, f \in A_{v_2}$ and six points $g, h, i, j, k, l \in A_{v_0}$, such that a is compatible with h, b is compatible with g and h, c is incompatible with i, d is incompatible with j, e is compatible with k and l, f is compatible with l. The pattern $2V$ also has the associated structure ($a \neq b$ or $g \neq h$) and ($e \neq f$ or $k \neq l$). The pattern $2V$ is pictured in Figure 22. When a pattern has an associated structure given by a property $\mathcal{P}$, the property $\mathcal{P}$ must be preserved by extension and reduction operations. For example, if $\mathcal{P}$ is $a \neq b$ then the points a and b cannot be merged during a reduction. It is worth pointing out that in a CSP instance, all points are assumed to be distinct and hence a property such as $a \neq b$ is necessarily satisfied.

Lemma 11. *$2V$ is intractable.*

Proof. Let the gadget V^+ be the pattern on two variables v_0, v_1 with points $a \in A_{v_0}$ and $b, c \in A_{v_1}$ such that a is compatible with both b and c, together with the structure $b \neq c$. In the pattern $2V$, either b is compatible with two different points g and h, or h is compatible with two different points a and b. So, if $2V$ occurs in a CSP instance on variables v_0', v_1', v_2', then the gadget V^+ necessarily occurs in the constraint between v_0' and v_1'. By an identical argument, the gadget V^+ must also occur in the constraint between v_0' and v_2'.

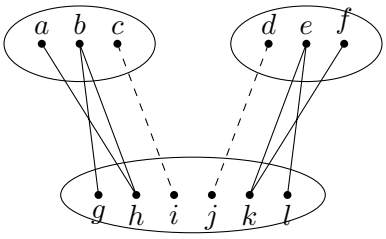


Figure 22: The pattern $2V$.

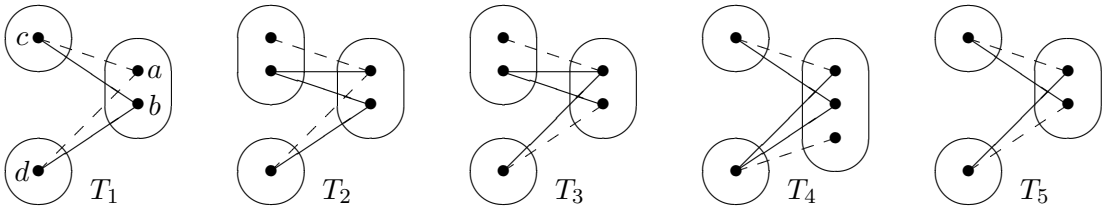


Figure 23: The set of tractable patterns T.

We define an equality constraint between two variables v and v' as the constraint consisting of compatibility edges between identical values in the domains of v and v' and incompatibility edges between all pairs of distinct values. Thus, by definition, a point in an equality constraint is compatible with only one point. Since the gadget V^+ contains a point a compatible with two different points, V^+ does not occur in an equality constraint.

We will reduce CSP to $\mathrm{CSP}(\overline{2V})$. Let I be a CSP instance. For each pair of variables 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 between v' and w' the same constraint as there was between v and w. All other constraints involving v' or w' are trivial. We also replace the constraint between v and w by a trivial constraint. After this transformation, v and w' are the only variables which share a non trivial constraint with v'. Let I' be the instance obtained after all such transformations are simultaneously 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. Hence, the gadget V^+ cannot occur in both of these constraints. It follows that $2V$ cannot occur in I'. So we have reduced I to an instance without any occurrence of the pattern $2V$. This polynomial reduction from CSP to $\mathrm{CSP}(\overline{2V})$ shows that $2V$ is intractable. □

Let T be the set $\{T_1, T_2, T_3, T_4, T_5\}$ of patterns shown in Figure 23.

No pattern in T can be reduced to a sub-pattern of a different pattern in T. As we will show, each T_i ($i = 1, \ldots, T_5$) defines a tractable class of binary CSP instances. For example, T_4 defines a class of instances which includes as a proper subset all instances with zero-one-all constraints (Co-

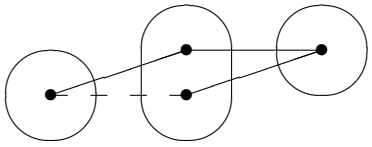


Figure 24: The pattern Diamond.

hen, Cooper, & Jeavons, 1994). Zero-one-all constraints can be seen as a generalisation of 2SAT clauses to multi-valued logics.

Let $2I$ represent the pattern composed of two separate copies of $1I$, i.e. $2I$ consists of four points a, b, c, d such that $var(a)$, $var(b)$, $var(c)$, $var(d)$ are all distinct and both a, b and c, d are pairs of incompatible points.

Definition 29 (irreducible pattern). *We say that a pattern P is* irreducible *if we cannot apply merging or dp-elimination on P.*

Theorem 1. *Let P be an irreducible flat pattern on two constraints. Then P is tractable if and only if P is a sub-pattern of one of the patterns in $T \cup \{2I\}$.*

4.1.3 Proof of Theorem 1

Proof. $\Rightarrow$: A two-constraint pattern involves either three or four distinct variables. Consider first the latter case, in which P is composed of two separate irreducible one-constraint patterns P_1 and P_2 on four distinct variables. By Lemma 10, P is tractable if and only if both P_1 and P_2 are tractable. Furthermore, by Lemma 9, all tractable one-constraint irreducible patterns are sub-patterns of $1I$. Thus, if P is tractable, then it is a sub-pattern of $2I$, by a combination of P_1 and P_2 being sub-patterns of $1I$. It only remains to study two-constraint patterns on *three* variables.

From Lemma 6, Lemma 8 and Corollary 1, we know that we only have to study patterns P with at most one incompatibility edge in each constraint such that P does not contain the pattern Z. If one of the constraints does not contain any incompatibility edge at all, then the pattern is reducible by merging and/or dp-elimination to a pattern with only one constraint or to the pattern Diamond, shown in Figure 24, which is a sub-pattern of T_2, T_3 and T_4. So we can assume from now on that there is exactly one incompatibility edge ($p \in A_{v_0}, b \in A_{v_1}$) between v_0 and v_1, and also exactly one incompatibility edge ($p' \in A_{v_0}, c \in A_{v_2}$) between v_0 and v_2. The "skeleton" of incompatibility edges of an irreducible tractable pattern can thus take two forms according to whether $p = p'$ (skeleton of type 1) or $p \neq p'$ (skeleton of type 2).

From Lemma 7 we know that $|A_v| \leq 2$ for each variable v with only one explicitly incompatible point, and that $|A_v| \leq 3$ for each variable v with two explicitly incompatible points. We know from Lemmas 8 and 11 that both Z and $2V$ are intractable, so by Corollary 1 we must look for patterns in which neither one occurs. We know that we have two possible incompatibility skeletons to study, each one implying a maximum number of points appearing in the pattern.

We first consider the incompatibility skeleton of type 1, shown in Figure 25.

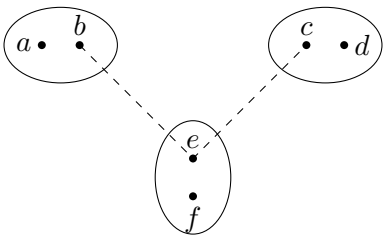


Figure 25: Incompatibility skeleton of type 1.

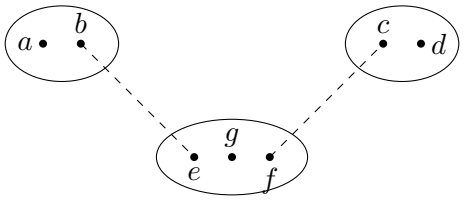


Figure 26: Incompatibility skeleton of type 2.

Suppose that a is a point in the pattern. Then there must be a compatibility edge between a and e, otherwise we could merge a and b. There also must be a compatibility edge between a and f, otherwise a would be a dangling point. Similarly, if d is a point in the pattern, then there must be compatibility edges between d and e, and between d and f. So if both a and d are points in the pattern, then the pattern $2V$ occurs. So, by Lemma 11 and Corollary 1, a and d cannot be both points of the pattern. Since they play symmetric roles, we only have two cases to consider: either a is a point in the pattern and not d, or neither a nor d is a point in the pattern.

If a is a point in the pattern and not d, then the only remaining edges to consider are $\{f, b\}$ and $\{f, c\}$. $\{f, b\}$ cannot be a compatibility edge, because otherwise the pattern Z would occur. $\{f, c\}$ must be a compatibility edge, otherwise we could merge f and e. Thus the pattern is T_2.

On the other hand, if neither a nor d is a point in the pattern, then the only remaining edges to consider are $\{f, b\}$ and $\{f, c\}$. If one of them is a compatibility edge but not the other, then f would be a dangling point. So either both $\{f, b\}$ and $\{f, c\}$ are compatibility edges, or neither of them is. However, the latter case is a sub-pattern of the former one which is T_1. So the only possible irreducible tractable patterns with this incompatibility skeleton are sub-patterns of T_1 or T_2.

We now consider the incompatibility skeleton of type 2, shown in Figure 26.

If g is a point in the pattern, then there must be a compatibility edge between g and b, otherwise we could merge g and e. There also must be a compatibility edge between g and c, otherwise we could merge g and f. We suppose, for a contradiction, that a is a point in the pattern. Then there is a compatibility edge between a and e, otherwise we could merge a and b. There is also a compatibility edge either between a and f or between a and g, otherwise a would be a dangling point. We cannot have a compatibility edge between a and g, otherwise the pattern Z would occur. So there is a compatibility edge between a and f. There is a compatibility edge either between b and f or between c and e, otherwise we could merge e and f. We cannot have a compatibility

edge between b and f, otherwise the pattern Z would occur. We cannot have a compatibility edge between c and e, otherwise the pattern $2V$ would occur. So a cannot be a point in the pattern. Since a and d play symmetric roles, we can also deduce that d cannot be a point in the pattern. So the only remaining edges are $\{b, f\}$ and $\{c, e\}$. At least one of them is a compatibility edge, otherwise we could merge e and f. If both of them are compatibility edges, the pattern $2V$ occurs. So exactly one of them is a compatibility edge. Since they play symmetric roles, we can assume for instance that $\{b, f\}$ is a compatibility edge while $\{c, e\}$ is an unknown edge which means that the pattern is T_4.

We now consider the case in which g is not a point in the pattern. Suppose that a is a point in the pattern. There is a compatibility edge between a and e, otherwise we could merge a and b. There is also a compatibility edge between a and f, otherwise a would be a dangling point. Similarly, if d is a point in the pattern, then there must be compatibility edges between d and e, and between d and f. At least one of the edges $\{b, f\}$ and $\{c, e\}$ must be a compatibility edge, otherwise we could merge e and f. In either case, Z occurs in the pattern. So a and d cannot both be points of the pattern. Since they play symmetric roles, we only have two cases to consider: either a is a point in the pattern and not d, or neither a nor d is a point in the pattern.

If a is a point in the pattern, then the only remaining edges to consider are $\{b, f\}$ and $\{c, e\}$. At least one of them is a compatibility edge, otherwise we could merge e and f. There is no compatibility edge between b and f, otherwise the pattern Z would occur. So there is a compatibility edge between c and e. Hence the pattern is T_3.

If neither a nor d is a point in the pattern, then the only remaining edges are $\{b, f\}$ and $\{c, e\}$. At least one of them is a compatibility edge, otherwise we could merge e and f. So either exactly one of them is a compatibility edge, or they both are. However, the former case is a sub-pattern of the latter which corresponds to pattern T_5. So the only possible irreducible tractable patterns with this incompatibility skeleton are sub-patterns of T_3, T_4 or T_5.

So if P is a tractable irreducible pattern on two constraints, then P is reducible to a sub-pattern of one of the patterns in $T \cup \{2I\}$.

$\Leftarrow$: We now give the tractability proofs for all patterns in $T \cup \{2I\}$. We assume throughout that we have applied until convergence the preprocessing operations: arc consistency, neighborhood substitution and single-valued variable elimination. The proof of tractability of T_1 is by far the longest of these proofs and will require a dozen lemmas showing that many simplification operations can be applied to instances in $\mathrm{CSP}(\overline{T_1})$ without introducing the pattern T_1 and describing the structure of the simplified instance. The final step consists in observing that the simplified instance belongs to a known tractable class (Cooper & Živný, 2011a). The proofs of tractability of the other patterns are based on the same principle: simplification operations can be applied which do not introduce the pattern and the resulting simplified instance belongs to a known, sometimes trivial, tractable class.

Proof of tractability of T_1 : Let I be an instance in $\mathrm{CSP}(\overline{T_1})$. Let the gadget X be the pattern on two variables v_0, v_1, shown in Figure 12, with points $a, b \in A_{v_0}$ and $c, d \in A_{v_1}$ such that a is incompatible with c and compatible with d, and b is compatible with c and incompatible with d.

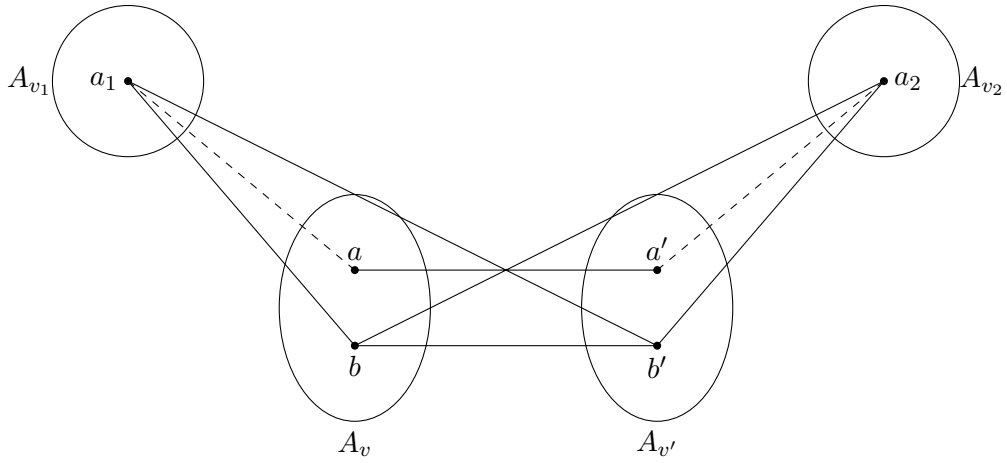


Figure 27: Introduction of the pattern T_1.

Suppose that the gadget X is a sub-pattern of the instance I. Suppose a is in a solution S. Let $e \in A_{v_2}$ 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 c 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 is not a sub-pattern of the instance. We say that an instance $I \in \mathrm{CSP}(\overline{T_1})$ is *simplified* if we have applied neighborhood substitution operations until convergence and all gadgets X have been eliminated from I. We say that I is *fusion-simplified* if it is simplified and all (simple or complex) fusion operations have been performed that do not introduce T_1. The following lemma indicates when we can perform fusion operations.

Lemma 12. *Consider a (simple or complex) fusion of two variables v, v' in an instance $I \in \mathrm{CSP}(\overline{T_1})$. Suppose that whenever (a, a') and (b, b') are pairs of fused points during this fusion, such that $a \neq b \in A_v$ and $a' \neq b' \in A_{v'}$, either a and b' were incompatible in I or b and a' were incompatible in I. Then the pattern T_1 cannot be introduced by this fusion.*

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 central variable of T_1 are created by the fusion of pairs of points (a, a') and (b, b') such that the compatibilities of the points $a, b \in A_v$ and $a', b' \in A_{v'}$ with the two other points a_1, a_2 of T_1 are as shown in Figure 27.

Now, if a and b' were incompatible, then T_1 was already present on points a_1, a, b, b' in the original instance, and hence cannot be introduced by the fusion. Similarly, if b and a' were incompatible, then T_1 was already present on points b, a', b', a_2 in the original instance. □

Definition 30 (better than a point with respect to a variable). *$\forall v, v'$, $\forall a, b \in A_v$, 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 the variable v is obvious from the context), if every point in $A_{v'}$ 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 $\leq$. We write $a \equiv b$ if $a \leq b$ and $b \leq a$.

Lemma 13. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$*

1. *$\forall (v, v')$, the order $\leq$ on A_v with respect to v' is total.*
2. *$\forall v$, $\forall a, b \in A_v$, there is v' such that $a < b$ for v'.*
3. *$\forall v$, $\forall a, b \in A_v$, 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 neighborhood substitution.
3. Because of the initial forbidden pattern.

□

Lemma 14. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, if $a < b < c$ for (v_0, v_1), then there exists $v_2 \neq v_1$ such that $c < b < a$ for (v_0, v_2).*

Proof. Since we have $a < b$ for (v_0, v_1), from Lemma 13.2 there is some v_2 such that $b < a$ for (v_0, v_2). Since $b < c$ for (v_0, v_1), $c \leq b$ for (v_0, v_2) by Lemma 13.3. If $c < b$ for v_2, then we have the lemma. Otherwise, we have $c \equiv b < a$ for v_2. Since $b < c$ for v_1, there exists $v_3 \neq v_1, v_2$ such that $c < b$ for v_3. Since $a < b$ for v_1, $b \leq a$ for v_3. So $c < b \leq a$ for v_3. So we have $c < a$ for both v_2 and v_3, which is not possible. So we must have $c < b < a$ in v_2. □

Lemma 15. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, $\forall a, b, c, d \in A_{v_0}$, for all $v_1 \neq v_0$ none of the following is true:*

1. *$a \equiv b < c < d$ for v_1.*
2. *$a < b \equiv c < d$ for v_1.*
3. *$a < b < c \equiv d$ for v_1.*

Proof. We give the proof only for the case 1, since the proofs of cases 2 and 3 are almost identical. Since we have $a < c < d$ for v_1, from Lemma 14 there exists v_2 such that $d < c < a$ for v_2. Likewise, since $b < c < d$ for v_1, there exists v_2' such that $d < c < b$ for v_2'. Since $d < c$ for both v_2 and v_2', $v_2 = v_2'$ by Lemma 13.3. This leaves three possibilities:

1. $d < c < b < a$ for v_2: from Lemma 14 we know there is v_3 such that $a < b < c$ for v_3. So we have $a < c$ for both v_1 and v_3 with $v_1 \neq v_3$ (since $a \equiv b$ for v_1), which is not possible by Lemma 13.3. So we cannot have this possibility.

2. $d < c < b \equiv a$ for v_2: since $a \equiv b$ for both v_1 and v_2, by Lemma 13.2 there is a different v_3 such that $a < b$ for v_3. Since $c < b$ for v_2 and $v_3 \neq v_2$, $b \leq c$ for v_3. So $a < c$ for v_3. But we also have $a < c$ for v_1 and $v_1 \neq v_3$. So by Lemma 13.3 we cannot have this possibility.

3. $d < c < a < b$ for v_2: equivalent to the case $d < c < b < a$ after interchanging a and b.

□

Corollary 3. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, if for some (v_0, v_1), we have at least three equivalence classes in the order on A_{v_0} with respect to v_1 then:*

1. *The order on A_{v_0} with respect to v_1 is strict.*

2. *There is v_2 such that the order on A_{v_0} with respect to v_2 is the exact opposite to the order on A_{v_0} with respect to v_1.*

3. *$\forall v_3$ such that $v_3 \neq v_0, v_1, v_2$, there is only one equivalence class in the order on A_{v_0} with respect to v_3.*

Proof. Points 1,2 and 3 follow respectively from Lemma 15, Lemma 14 and Lemma 13. □

Lemma 16. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, $\forall a, b, c, d \in A_{v_0}$, there is no v_1 such that $a \equiv b < c \equiv d$ for v_1.*

Proof. By Lemma 13.2, we know there is some v_2 such that $a < b$ for v_2. Since we have $a < c$ and $a < d$ for v_1, by Lemma 13.3, we have $c \leq a$ and $d \leq a$ for v_2. From Corollary 3, we cannot have $c < a < b$ or $d < a < b$ for v_2, so we have $d \equiv c \equiv a < b$ for v_2. Since we have $c \equiv d$ for both v_1 and v_2, we have a different variable v_3 such that $c < d$ for v_3. Since $c < b$ for v_2 and $v_3 \neq v_2$, $b \leq c$ for v_3. So $b < d$ for v_3. But we also have $b < d$ for v_1 and $v_1 \neq v_3$. So, by Lemma 13.3, we cannot have this possibility. □

Lemma 17. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, if for some (v, v') there are at least three equivalence classes in the order on A_v with respect to v', then there are the same number of points in both A_v and $A_{v'}$ and both the order on A_v with respect to v' and the order on $A_{v'}$ with respect to v are strict.*

Proof. Let d be the number of points in A_v and d' the number of points in $A_{v'}$. From Lemma 15 we know that the order on A_v with respect to v' is strict. So we have $a_1 < a_2 < \cdots < a_d$ for (v, v'). So we have $(a'_1, a'_2, \ldots, a'_{d-1})$ such that $\forall i \in \{1, \ldots, d\}$, a_i and a'_i are incompatible but a_{i+1} and a'_i are compatible. So $\forall i \in \{2, \ldots, d\}$ we have a_i and a'_i which are incompatible but a_i and a'_{i-1} are compatible. So, by Lemma 13.1 we have $a'_1 > a'_2 > \cdots > a'_{d-1}$ for v. Moreover, since a_1 is incompatible with a'_1, a_1 is incompatible with all a'_i for $1 \leq i < d$. By arc consistency, we have a'_0 such that a_1 and a'_0 are compatible. So we have $a'_0 > a'_1 > a'_2 > \cdots > a'_{d-1}$. So we have $d \leq d'$ and at least three equivalence classes in the order on $A_{v'}$ with respect to v. By switching v and v' in the proof, we can prove the remaining claims of the Lemma. □

We say that the pair of variables (v, v') is a *3-tiers pair* if there are at least 3 classes of equivalence in the order on A_v with respect to v'; we say that it is a *2-tiers pair* otherwise.

Lemma 18. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, suppose we have v and v' such that (v, v') is a 3-tiers pair. Then we can perform the simple fusion of v, v' without introducing T_1.*

Proof. Let d be the number of points in A_v. From Lemma 17 we know that the points in A_v can be denoted $a_1 < a_2 < \cdots < a_d$ for v' and the points in $A_{v'}$ can be denoted $b_1 < b_2 < \cdots < b_d$ for v. We will show that we can perform a simple fusion of v and v' with fusion function f given by $f(a_i) = b_{d+1-i}$ ($i = 1, \ldots, d$).

Claim: $\forall 1 \leq i \leq d$, $\{b_{d+1-i}, b_{d+1-i+1}, \ldots, b_d\}$ is the exact set of points compatible with a_i.

If we have $a_i < a_j$ for v', it means a_i is compatible with strictly less points in $A_{v'}$ than a_j. By arc consistency, every point in A_v is compatible with a point in $A_{v'}$. So $\forall 1 \leq i \leq d$, we have d possibilities $(1, 2, \ldots, d)$ for the number of points compatible with a_i. Since we have d points in A_v, it means that $\forall 1 \leq i \leq d$, a_i is compatible with i points in $A_{v'}$. By definition of the order on a variable with respect to another variable, the points in $A_{v'}$ compatible with a point $a_i \in A_v$ are the greatest points for v. So we have the claim.

We now show that $\forall 1 \leq i \leq d$, if a_i is in a solution S, then there is a solution S' such that both b_{d+1-i} and a_i are in S'. Let b be the point of S in v'. If $b_{d+1-i} = b$, then we have the result. Otherwise, let $c \neq b$ be a point of S. If $c = a_i$, then from the above claim we know that c is compatible with b_{d+1-i}. Otherwise, let $v_c = var(c)$. So $v_c \neq v$. From the above claim we have $b_{d+1-i} < b$ for v. So $b \leq b_{d+1-i}$ for v_c. So b_{d+1-i} is compatible with c. So b_{d+1-i} is compatible with all the points in S. So we have a solution S' obtained by replacing b by b_{d+1-i} in S which contains both a_i and b_{d+1-i}.

We now perform the simple fusion of v and v' with fusion function $f(a_i) = b_{d+1-i}$ for $1 \leq i \leq d$; we have just shown that this is a valid simple fusion. It only remains to show that the resulting instance is in $\mathrm{CSP}(\overline{T_1})$, since by Lemma 1 it is solvable if and only if the original instance was solvable. Let a, b be two distinct points in A_v. Without loss of generality, suppose that $a < b$ for v'. By choice of the fusion function f, b is the smallest (according to the order $<$ for v') of the points in A_v compatible with $f(b)$. Therefore, a and $f(b)$ are incompatible. The result then follows from Lemma 12. □

Therefore, from now on, in a fusion-simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, we can assume that each pair (v, v') is a 2-tiers pair. We call *winner* for (v, v') the points in the greater equivalence class in the order for (v, v'). The other points are called *losers* for this order. A same point can (and actually will) be a winner for a given order and a loser for another order. If for a given order there is only one equivalence class, then all the points are considered winners.

The winners for (v, v') are compatible with all the points in $A_{v'}$. The losers for (v, v') are only compatible with the winners for (v', v).

We say that a variable v is *one-winner* if $\forall v' \neq v$, either only one point of A_v is a winner for (v, v') or all the points in A_v are. Similarly, we say that a variable v is *one-loser* if $\forall v' \neq v$, either only one point of A_v is a loser for (v, v') or all the points of A_v are winners for (v, v').

Lemma 19. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, $\forall v$, if there is v' such that there is only one winner for (v, v'), then v is one-winner. Similarly, if there is v' such that there is only one loser for (v, v'), then v is one-loser.*

Proof. Let $a, b, c, d, e, f \in A_v$ be such that there are $v_1 \neq v_2$ with $a \equiv b < c$ for v_1, $d < e \equiv f$ for v_2, $a \neq b$ and $e \neq f$. If $d \neq c$, then from Lemma 16, we have $a \equiv b \equiv d \equiv c$ for v_1 and $d < e \equiv f \equiv c$ for v_2. So $d < c$ for both v_1 and v_2 with $v_1 \neq v_2$ (which is a contradiction by Lemma 13.3). So we cannot have $d \neq c$. So $d = c$. So we have $c < e \equiv f$ for v_2. From Lemma 16 we have $c < e \equiv f \equiv a \equiv b$ for v_2. Since we have $a \equiv b$ for both v_1 and v_2, by Lemma 13.2 there is a different variable v_3 such that $a < b$ for v_3. Since $a < c$ for v_1, $c \leq a$ for v_3. So $c < b$ for v_3. So $c < b$ for both v_2 and v_3 with $v_2 \neq v_3$. This is impossible by Lemma 13.3. So we have the Lemma. □

Corollary 4. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, $\forall v$, either v is one-winner or v is one-loser.*

Proof. Lemma 13.2 tells us that there exists v' and $a, b \in A_v$ such that $a < b$ for v'. By Lemma 16, either there is only one winner for (v, v') or only one loser. The result follows directly from Lemma 19. □

Let E be the set of one-winner variables and $F = V \setminus E$ with V being the set of all variables. From Corollary 4, the variables in F are one-loser. Let $v_a, v_b \in E$ be such that there is a non-trivial constraint between v_a and v_b. Since $v_a \in E$, there is only one winner a for v_b in v_a. Similarly, there is only one winner b for v_a in v_b. We can perform a complex fusion of v_a and v_b with hinge value a and fusion function the constant function $f = b$.

By Lemma 1, the instance resulting from this fusion is solvable if and only if the original instance was solvable.

Lemma 20. *The complex fusion of two one-winner variables v_a and v_b in a simplified instance of $\mathrm{CSP}(\overline{T_1})$ does not create the forbidden pattern.*

Proof. Suppose that (c, c') and (d, d') are corresponding pairs of points during this fusion, with $c \neq d \in A_{v_a}$ and $c' \neq d' \in A_{v_b}$. Since v_a only has one winner for v_b, we know that either c or d is a loser for v_b. Without loss of generality, suppose d is a loser for v_b. Since v_b only has one winner for v_a, and losers are only compatible with winners, we know that d is incompatible with c' (since it is necessarily compatible with d' for the fusion to take place). The result now follows directly from Lemma 12. □

We have shown that we can fusion any pair of variables in E between which there is a non-trivial constraint. We now do the same for F, the set of one-loser variables.

Lemma 21. *In a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$, let $v_a, v_b \in F$ (where F is the set of one-loser variables of I) be such that there is a non-trivial constraint between v_a and v_b. Let $a \in A_{v_a}$ and $b \in A_{v_b}$ be such that a is incompatible with b. If $a' \in A_{v_a}$ is in a solution S and $a' \neq a$, then b is in a solution S' containing a'.*

Proof. Let b' be the point of S in v_b. If $b' = b$, then we have the result. Since v_a is a one-loser variable, we know that all points in A_{v_a} other than a are winners. Thus a' is compatible with b. By a symmetric argument, b' is compatible with a. If we have $c \in S$ such that b is incompatible with c, then a, b', c and b form the forbidden pattern. So b is compatible with all the points in S. So if we replace b' by b in S we get a solution S' containing both a' and b. □

Lemma 22. *Let v_a,v_b both be one-loser variables in a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$ such that $a \in A_{v_a}$ and $b \in A_{v_b}$ are incompatible. Then we can perform the complex fusion of v_a and v_b with hinge value a and fusion function the constant function $f = b$ without introducing the forbidden pattern T_1.*

Proof. It follows from Lemma 21 that we only need to consider solutions containing a or b. We can therefore perform a complex fusion of v_a and v_b with hinge value a and fusion function the constant function $f = b$.

In all pairs (c, c') of corresponding points in this fusion, we must have either $c = a$ or $c' = b$. Suppose that (c, c') and (d, d') are corresponding pairs of points during the fusion, with $c \neq d \in A_{v_a}$ and $c' \neq d' \in A_{v_b}$. Without loss of generality, we can assume that $c = a$ and $d' = b$. But we know that a was incompatible with b. From Lemma 12 we can deduce that the fusion does not introduce the pattern T_1. □

We say a point a is *weakly incompatible* with a variable v if there exists some $b \in A_v$ such that a is incompatible with b.

Lemma 23. *Let v be a one-loser variable in a simplified instance $I \in \mathrm{CSP}(\overline{T_1})$. Let f be a point in A_v. Then f is weakly incompatible with one and only one variable.*

Proof. From the definition of a one-loser variable, we know that there is some variable v' such that f is a loser for (v, v'). So f is weakly incompatible with v'. From Lemma 13.3, we know that f is a loser only for (v, v'). Furthermore, by arc consistency we know that f is compatible with all points of A_u, for all variables u such that f is not a loser for (v, u). So f is weakly incompatible with one and only one variable, namely v', and we have the Lemma. □

We have shown that after all possible fusions of pairs of variables, we have two sets of variables E (the set of one-winner variables) and $F = V \setminus E$ (the set of one-loser variables) such that:

- $\forall v, v' \in E$, there is no non-trivial constraint between v and v'.
- $\forall v, v' \in F$, there is no non-trivial constraint between v and v'.
- $\forall v \in F, \forall f \in A_v$, f is weakly incompatible with one and only one variable $v' \in E$. This is from Lemma 23. Furthermore, f is incompatible with all points of $A_{v'}$ but one (since $v' \in E$ is a one-winner variable).
- The only possible non-trivial constraint between a variable $v_1 \in E$ and another variable $v_2 \in F$ is the following with d_1 being the size of the domain of v_1:
  - There is a point $b \in A_{v_2}$ incompatible with exactly $d_1 - 1$ points in A_{v_1}.

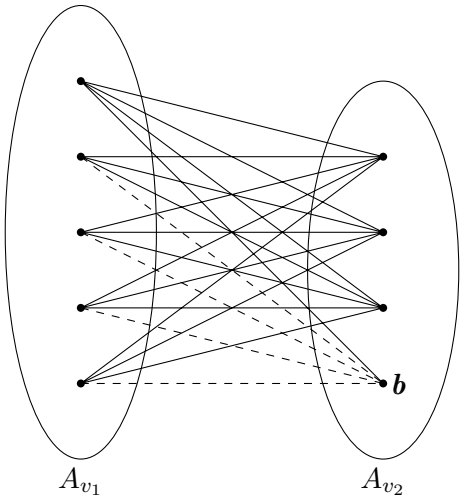


Figure 28: Constraint between a one-winner variable v_1 and a one-loser variable v_2.

  - $\forall b' \in A_{v_2}$ with $b' \neq b$, b' is compatible with all points in A_{v_1}.

  This is illustrated in Figure 28. It is easily seen that this constraint can be written $(v_2 = b) \Rightarrow (v_1 = a)$.

We call NOOSAT (for Non-binary Only Once Sat) the following problem:

- A set of variables $V = \{v_1, v_2, \ldots, v_e\}$.
- A set of values $A = \{a_1, a_2, \ldots, a_n\}$.
- 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 $v_i = a_j$.
  - $\forall i, j, p, q((v_i = a_j) \in C_p) \wedge ((v_i = a_j) \in C_q) \Rightarrow p = q$.

Lemma 24. *CSP($\overline{T_1}$) can be reduced to NOOSAT in polynomial time.*

Proof. The total number of assignments decreases when we fuse variables, so the total number of (simple or complex) fusions that can be performed is linear in the size of the original instance. Hence we can produce a fusion-simplified version of an instance $I \in \mathrm{CSP}(\overline{T_1})$ in polynomial time. Thus suppose we have a fusion-simplified instance in $\mathrm{CSP}(\overline{T_1})$. We have shown that the non-trivial constraints between variables $v \in F$ and $v' \in E$ are all of the form $v = b \Rightarrow v' = a$. Furthermore, from Lemma 23 and the third bullet point in the description of a post-fusions instance, each variable-value assignment $v = b$ occurs in exactly one such constraint. For any $v \in F$, 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.

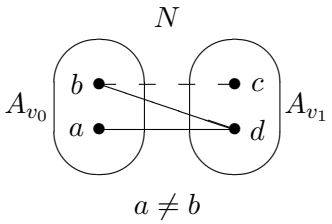


Figure 29: The gadget N.

Then there must have been two constraints $v_2 = b \Rightarrow v_1 = a$ and $v_3 = c \Rightarrow v_1 = a$ and with $v_1 \in E, v_2 \neq v_3 \in F$. 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($\overline{T_1}$) can be reduced to NOOSAT. □

The constraints in NOOSAT are convex when viewed as $\{0, \infty\}$-valued cost functions on the assignment-sets $\{\langle v_1, a_1\rangle, \ldots, \langle d_d, a_d\rangle\}$ (the cost being infinite if and only if the number of assignments in this set is 0) and these assignment-sets (corresponding to clauses) are non overlapping. So, from (Cooper & Živný, 2011a), it is solvable in polynomial time. Hence the forbidden pattern T_1 is tractable.

Proof of tractability of T_2 : Let N be the gadget shown in Figure 29: two variables v_0, v_1 with points $a, b \in A_{v_0}$ and $c, d \in A_{v_1}$, such that a,b are both compatible with d, b is incompatible with c, and with the structure $a \neq b$.

Suppose we are given a CSP instance containing 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 a and e are compatible. If b is incompatible with e, then we have the forbidden pattern T_2 on d, c, b, a, e. So b is compatible with e. If all the points in A_{v_1} which are compatible with a are also compatible with b, then we can remove a by neighborhood substitution. So, assuming that neighborhood substitution operations have been applied until convergence, if we have the gadget N, then there is a point $g \in A_{v_1}$ compatible with a and incompatible with b.

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

Definition 31 (functional constraint). *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 & Živný, 2012), we only need to consider the connected components of the constraint graph which contain V^-.

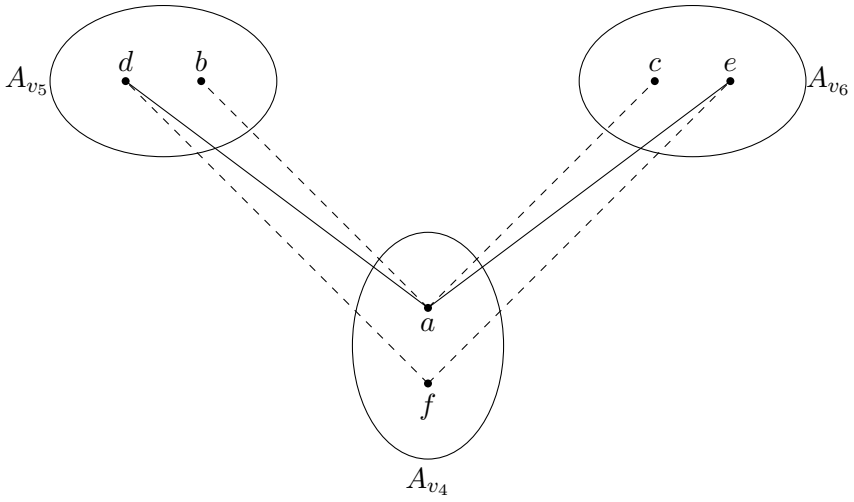


Figure 30: The three variables v_4, v_5 and v_6.

Lemma 25. *If in an instance from* $\mathrm{CSP}(\overline{T_2})$*, 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 $d \in A_{v_5}$ 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_4} 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 $f \neq a$ be a point in A_{v_4}. By arc consistency, we have $d \in A_{v_5}$ and $e \in A_{v_6}$ 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 we have the situation illustrated in Figure 30.

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_4}. So each point in A_{v_5} and A_{v_6} compatible with a point in A_{v_4} is compatible with only one point of A_{v_4}. 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 26. *In a connected component of the constraint graph containing* V^- *of an instance from* $\mathrm{CSP}(\overline{T_2})$*, 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.

$P(\{v_4, v_5\})$ is true from Lemma 25.

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 $P(V)$ is true. Let $V' = V_{\text{all}} \backslash V$. Let $v' \in V'$. Let $v \in V$ be such that $C(v, v')$ (the constraint on v, v') is non-trivial. So there is $d \in A_v$ and $e \in A_{v'}$ such that d and e are incompatible. Since V is connected and of

cardinality at least two, there is $v'' \in V$ such that $C(v, v'')$ is functional. By arc consistency and elimination of single-valued variables, there is necessarily a point $f \in A_{v''}$ such that d and f are incompatible. So d, e and f form the gadget V^-. From Lemma 25 we know $C(v, v')$ is functional. So $P(V)$ is true for all subsets of V_{all}. □

Lemma 27. *In an instance from* $\text{CSP}(\overline{T_2})$, $\forall v$ *such that* v *is in a connected component of the constraint graph containing* V^-*, all points in* A_v *are weakly incompatible with the exact same set of variables.*

Proof. Let $a \in A_v$ 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 $b \neq a$ in v. By arc consistency, we know there is $c \in A_{v'}$ such that a and c are compatible. Since $C(v, v')$ is functional from v' to v, c is compatible with only one point in A_v, namely a, so b is incompatible with c. So every point in A_v is weakly incompatible with v'.

So $\forall (v, v'), a \in A_v$ weakly incompatible with $v' \Rightarrow \forall b \in A_v, b$ weakly incompatible with v'. □

Definition 32 (path of functionality). *A sequence of variables* $(v_0, v_1, \ldots, v_k)$ *is a* path of functionality *if* $\forall i \in \{0, \ldots, k-1\}$, $C(v_i, v_{i+1})$ *is functional from* v_i *to* v_{i+1}.

Lemma 28. *In a connected component of the constraint graph containing* V^- *of an instance from* $\text{CSP}(\overline{T_2})$, $\forall v, v'$*, either* v' *is connected to only one other variable 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 $(v_0 = v, v_1, v_2, \ldots, v_k = v')$ with all v_i different. If v' is connected to at least two other variables in the constraint graph, then we have a path of incompatibility $(v_0, v_1, v_2, \ldots, v_{k-1}, v_k, v_{k+1})$ with $v_{k+1} \neq v_{k-1}$. From Lemma 27 we have a path of incompatibility $(a_0 \in A_{v_0}, a_1 \in A_{v_1}, \ldots, a_k \in A_{v_k}, a_{k+1} \in A_{v_{k+1}})$. So $\forall i \in \{1, \ldots, k\}$, a_{i-1}, a_i and a_{i+1} form the gadget V^-. So from Lemma 25, $\forall i \in \{1, \ldots, k\}$, $C(v_{i-1}, v_i)$ is functional from v_{i-1} to v_i. So we have a path of functionality from v to v'. □

Variables which are connected to at most one other variable in the constraint graph can be removed from the instance I since, by arc consistency, any solution on the remaining variables can be extended to a solution for I. Once we have removed all such variables, for each connected component of the constraint graph, 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_0}) lead to a solution. Since this is clearly polynomial-time, the pattern T_2 is tractable.

Proof of tractability of T_3 : Consider an instance from $\text{CSP}(\overline{T_3})$.

Suppose that the gadget N, shown in Figure 29, is a sub-pattern of the instance and let e be a point in A_{v_2}, with $v_2 \neq v_0, v_1$. If e is compatible with b but not with a, then we have the forbidden

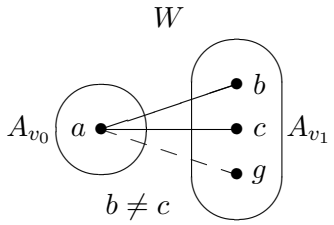


Figure 31: The gadget W.

pattern T_3. So if b is compatible with a point outside of A_{v_1}, then a is also compatible with the same point.

Let S be a solution containing b. Let f be the point of S in A_{v_1}. If f is compatible with a, then we can replace b by a in S while maintaining the correctness of the solution, since all the points in the instance outside of A_{v_1} which are compatible with b are also compatible with a.

If f is not compatible with a, then edges $\{a, f\}$, $\{f, b\}$ and $\{b, d\}$ form the gadget N. So, by our previous argument, if f is compatible with a point outside of A_{v_1}, then d is also compatible with the same point. We can then replace b by a and f by d in S while maintaining the correctness of the solution, since all the points in the instance outside of A_{v_1} which are compatible with b are also compatible with a and all the points in the instance outside of A_{v_0} which are compatible with f are also compatible with d. So if a solution contains b, then there is another solution containing a. Thus we can remove b while preserving solvability.

So each time the gadget N is present in an instance $I \in \text{CSP}(\overline{T_3})$, we can remove one of its points and hence eliminate N. Absence of the gadget N in I is equivalent to saying that all constraints are either trivial or bijections and hence (a subclass of) zero-one-all constraints (Cohen et al., 1994). Since all gadgets N can be removed in polynomial time and CSP instances with zero-one-all constraints can be solved in polynomial time, it follows that the pattern T_3 is tractable.

Proof of tractability of T_4 : Consider an instance from $\text{CSP}(\overline{T_4})$.

Let W be the gadget shown in Figure 31: 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_2}, with $v_2 \neq v_0, v_1$. If f is compatible with b but not with c, then we have the forbidden pattern T_4. Likewise, if f is compatible with c but not with b, then we have the forbidden pattern T_4. 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 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 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}. 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 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} and e and d have the same compatibility towards all the points in the instance outside of A_{v_0} and A_{v_1}. So if a solution contains c, then there is another solution containing b. 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 (Cohen et al., 1994). So if it is not present, the instance is tractable. Hence 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 et al., 2010) on three constraints. So the pattern T_5 is tractable by Corollary 1.

Proof of tractability of $2I$: Since $2I$ is the disjoint union of two copies of the trivially tractable pattern $1I$, the tractability of $2I$ follows directly from Lemma 10. □

Let $V_3 = \{v_0, v_1, v_2\}$, $A_3 = \{a_0, a_1, a_2\}$, $var_3(a_i) = v_i$ for $i \in \{0, 1, 2\}$, $E_3 = \{\{a_0, a_1\}, \{a_0, a_2\}\}$ and $cpt_3(a_0, a_1) = cpt_3(a_0, a_2) = F$. Let V−Middle=$\langle V_3, A_3, var_3, E_3, cpt_3, \{a_0\}\rangle$ be the existential pattern shown on the left of Figure 32 and V−Side=$\langle V_3, A_3, var_3, E_3, cpt_3, \{a_1\}\rangle$ be the existential pattern shown on the right of Figure 32.

4.1.4 Some NP-Complete Existential Patterns on Two Constraints

In order to identify all tractable existential patterns, we begin by showing that many simple existential patterns are NP-complete.

Definition 33 (copy of a variable). *Let I be a CSP instance. We say that $v' \in V$ is a* copy *in I of $v \in V$ on (A_0, A_1) with $A_0 \subset A_v$ and $A_1 \subset A_{v'}$ if:*

- $|A_0|$=$|A_1|$.
- $\forall a \in A_0$, $\exists b \in A_1$ *such that* $cpt(a, b) = T$ *and* $\forall c \neq b$ *in* A_1 *we have* $cpt(a, c) = F$.
- $\forall b \in A_1$, $\exists a \in A_1$ *such that* $cpt(a, b) = T$ *and* $\forall c \neq a$ *in* A_0 *we have* $cpt(c, b) = F$.
- $\forall a \in A_0$, $\forall b \in A_1$ *such that* $cpt(a, b) = T$, $\forall c \in A \backslash \{A_v, A_{v'}\}$, *we have* $cpt(b, c) = cpt(a, c)$.

For notational simplicity, we say that v' is a copy of v in I if $A_0 = A_v$ and $A_1 = A_{v'}$.

Let $V_3 = \{v_0, v_1, v_2\}$, $A_3 = \{a_0, a_1, a_2\}$, $var_3(a_i) = v_i$ for $i \in \{0, 1, 2\}$, $E_3 = \{\{a_0, a_1\}, \{a_0, a_2\}\}$ and $cpt_3(a_0, a_1) = cpt_3(a_0, a_2) = F$. Let V−Middle=$\langle V_3, A_3, var_3, E_3, cpt_3, \{a_0\}\rangle$ be the existential

Figure 32: Two intractable existential patterns on three variables.

pattern shown on the left of Figure 32 and V−Side=$\langle V_3, A_3, var_3, E_3, cpt_3, \{a_1\}\rangle$ be the existential pattern shown on the right of Figure 32.

Lemma 29. *V−Middle and V−Side are NP-Complete.*

Proof. Let $I = \langle V, A, var, E, cpt\rangle$ be an arc-consistent CSP instance. Let $v_1, \ldots, v_k$ be the variables of I. Let $I' = \langle V', A', var', E', cpt'\rangle$ be the CSP instance on variables $v'_1, \ldots, v'_{2k}$ such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i\}$ for all $1 \leq i \leq k$ and $A_{v'_i} = B_{v_{i-k}} \cup \{a_i\}$ for all $k+1 \leq i \leq 2k$, where $|B_i| = |A_i|$ $(1 \leq i \leq k)$. We can think of variables v'_i and v'_{i+k} as having the same domain except for the special value corresponding to a_i.
- For all $1 \leq i \leq k$, a_i is incompatible with a_{i+k} and compatible with all other points of I'. For all $k+1 \leq i \leq 2k$, a_i is incompatible with a_{i-k} and compatible with all other points of I'. For all $1 \leq i \leq k$, this prevents both a_i and a_{i+k} to be part of the same solution. The idea here is that for any solution S' to I', for all $1 \leq i \leq k$, either the point of S' in v'_i or the point of S' in v'_{i+k} will be an original point from I, or a copy of an original point from I.
- For all $1 \leq i < j \leq k$, for all $a \in A_i$, for all $b \in A_j$, $cpt'(a, b) = cpt(a, b)$. For all $1 \leq i \leq k$, v'_{i+k} is a copy of v'_i in I' on $(A_{v_i}, A_{v'_{i+k}} \setminus \{a_{i+k}\})$.

By construction, I' has a solution if and only if I has a solution, since (1) a solution to I can be duplicated to produce a solution to I', and (2) a solution to I' without the assignments a_i and after elimination of duplicates is a solution to I.

Furthermore, for all $1 \leq i \leq k$, a_i is incompatible with only one other point in I'. So for all $1 \leq i \leq k$, neither V−Middle nor V−Side occurs on a_i. So neither V−Middle nor V−side appears in I'. Thus we can reduce any CSP instance I to an arc-consistent CSP instance I' in which neither V−middle nor V−side appears. It follows that V−Middle and V−Side are NP-Complete. □

Let $V_2 = \{v_0, v_1\}$, $A_2 = \{a_0, a_1, a_2\}$, $var_2(a_0) = v_0$, $var_2(a_1) = var_2(a_2) = v_1$, $E_2 = \{\{a_0, a_1\}, \{a_0, a_2\}\}$ and $cpt_2(a_0, a_1) = cpt_2(a_0, a_2) = T$. Let V+Middle be the existential pattern $\langle V_2, A_2, var_2, E_2, cpt_2, \{a_0\}\rangle$ shown on the left of Figure 33 and V+Side the existential pattern $\langle V_2, A_2, var_2, E_2, cpt_2, \{a_1\}\rangle$ shown on the right of Figure 33.

Let ExpandedV+ = $\langle V, A, var, E, cpt, \{a_0\}\rangle$ be the existential pattern shown in Figure 34 and given by: $V = \{v_0, v_1, v_2\}$, $A = \{a_0, a_1, a_2, a_3\}$, $var(a_0) = v_0$, $var(a_1) = var(a_2) = v_1$, $var(a_3) = v_2$, $E = \{\{a_0, a_1\}, \{a_0, a_2\}, \{a_3, a_1\}, \{a_3, a_2\}\}$, $cpt(a_0, a_2) = cpt(a_3, a_1) = cpt(a_3, a_2) = T$ and $cpt(a_0, a_1) = F$.

V+Middle V+Side

Figure 33: Two intractable existential patterns on two variables.

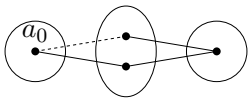


Figure 34: The existential pattern ExpandedV+.

Lemma 30. *V+Middle, V+Side and ExpandedV+ are NP-Complete.*

Proof. Let $I = \langle V, A, var, E, cpt \rangle$ be an arc-consistent CSP instance on variables $v_1, \ldots, v_k$. Let $I' = \langle V', A', var', E', cpt' \rangle$ be the CSP instance on variables $v'_1, \ldots, v'_{3k}$ such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i, b_i\}$ for all $1 \leq i \leq k$, $A_{v'_i} = B_{v_{i-k}} \cup \{a_i, b_i\}$ for all $k+1 \leq i \leq 2k$, and $A_{v'_i} = C_{v_{i-2k}} \cup \{a_i, b_i\}$ for $2k+1 \leq i \leq 3k$, where $|C_i| = |B_i| = |A_i|$ $(1 \leq i \leq k)$. We can think of variables v'_i, v'_{i+k} and v'_{i+2k} as having the same domain except for the special values corresponding to a_i, b_i.

- For all $1 \leq i \leq 3k$, for all $1 \leq j \leq 3k$ such that $i \neq j$, a_i is compatible with b_j and incompatible with all other points of $A_{v'_j}$. For all $1 \leq i \leq 3k$, for all $1 \leq j \leq 3k$ such that $i \neq j$, b_i is compatible with a_j and incompatible with all other points of $A_{v'_j}$. For all $1 \leq i \leq k$, this prevents any three points from $\{a_i, b_i, a_{i+k}, b_{i+k}, a_{i+2k}, b_{i+2k}$ to be part of the same solution. The idea here is that for any solution S' to I', for all $1 \leq i \leq k$, at least one of the points of S' in v'_i, v'_{i+k} and v'_{i+2k} will be an original point from I, or a copy of an original point from I.

- For all $1 \leq i < j \leq k$, for all $a \in A_{v_i}$, for all $b \in A_j$, $cpt'(a, b) = cpt(a, b)$. For all $1 \leq i \leq k$, v_{i+k} is a copy of v_k in I' on $(A_{v_i}, A_{v'_{i+k}} \setminus \{a_{i+k}, b_{i+k}\})$ and v_{i+2k} is a copy of v_k on $(A_{v_i}, A_{v'_{i+2k}} \setminus \{a_{i+2k}, b_{i+2k}\})$.

By construction, I has a solution if and only I' has a solution. Furthermore, for all $1 \leq i \neq j \leq k$, a_i is only compatible with b_j in A_{v_j} and b_j is itself only compatible with a_i in A_{v_i}. So for all $1 \leq i \leq k$, neither V+Middle nor V+Side occurs on a_i. Moreover, for all $1 \leq i, j, h \leq k$, a_i is only compatible with b_j in A_{v_j}, b_j is only compatible with a_h in A_{v_h} and a_h is only compatible with b_j in A_{v_j}. So for all $1 \leq i \leq k$, ExpandedV+ does not occur on a_i. So none of V+Middle, V+side or ExpandedV+ appear in I'. Hence V+Middle and V+Side are NP-Complete. □

Let V+− = $\langle V, A, var, E, cpt, \{a_1\} \rangle$ be the existential pattern shown in Figure 35 and given by $V = \{v_0, v_1\}$, $A = \{a_0, a_1, a_2\}$, $var(a_0) = v_0$, $var(a_1) = var(a_2) = v_1$, $E = \{\{a_0, a_1\}, \{a_0, a_2\}\}$, $cpt(a_0, a_1) = T$ and $cpt(a_0, a_2) = F$.

Lemma 31. *V+− is NP-Complete.*

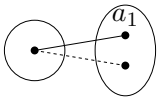


Figure 35: The existential pattern V+−.

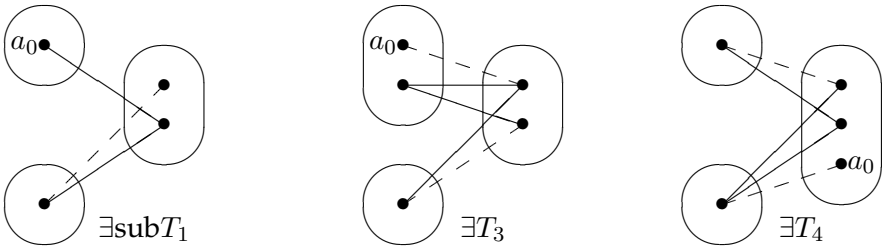


Figure 36: Three intractable existential patterns.

Proof. Let I be an arc-consistent CSP instance on variables $v_1, \ldots, v_k$ with at most one incompatibility edge in each constraint. Let I' be the CSP instance on variables $v'_1, \ldots, v'_k$ such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i\}$ for all $1 \leq i \leq k$.
- For all $1 \leq i < j \leq k$, a_i is incompatible with a_j. For all $1 \leq i < j \leq k$, for all $b \in A_{v'_j}$, a_i is incompatible with b if b is incompatible with a point $c \in A_{v'_i}$ and a_i is compatible with b otherwise.
- For all $1 \leq i < j \leq k$, for all $a \in A_{v_i}$, for all $b \in A_{v_j}$, $cpt'(a, b) = cpt(a, b)$.

For all $1 \leq i \neq j \leq k$, we know that a_i is compatible with a point $a' \in A_{v_j}$ if and only if a' is compatible with all points in A_{v_i}. Since at most one point in A_{v_j} is incompatible with a point in A_{v_i}, and since $|A_{v_j}| \geq 2$, there always exists such a point $a' \in A_{v_j}$. So I' is arc consistent. Furthermore, for all $1 \leq i \neq j \leq k$ we know that if a_i is compatible with a point $a' \in A_{v_j}$, then a' is compatible with all points in A_{v_i}. So for all $1 \leq i \leq k$, V+− does not occur on a_i and we can remove a_i by neighborhood substitution. So V+− does not appear in I' and the solvability of I' is the same as that of I. So we can reduce any CSP instance with at most one incompatibility edge in each constraint I to a CSP instance I' in which V+− does not appear. From Lemma 6, the set of CSP instances with at most one incompatibility edge in each constraint is NP-Complete. Thus V+− is NP-Complete. □

Let $\exists T_3 = \langle V, A, var, E, cpt, \{a_0\}\rangle$ be the existential pattern shown in the middle of Figure 36 and defined by $V = \{v_0, v_1, v_2\}$, $A = \{a_0, a_1, a_2, a_3, a_4\}$, $var(a_0) = var(a_1) = v_0$, $var(a_2) = var(a_3) = v_1$, $var(a_4) = v_2$, $E = \{\{a_0, a_2\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_4\}, \{a_3, a_4)\}\}$, $cpt(a_1, a_2) = cpt(a_1, a_3) = cpt(a_2, a_4) = T$ and $cpt(a_0, a_2) = cpt(a_3, a_4) = F$.

Lemma 32. *$\exists T_3$ is NP-Complete.*

Proof. Let I be an arc-consistent CSP instance on variables $v_1, \ldots, v_k$. Let I' be the CSP instance on variables $v'_1, \ldots, v'_{2k}$, $u'_1, \ldots, u'_k$ with compatibility function cpt' such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i, b_i\}$ for all $1 \leq i \leq k$, $A_{v'_i} = B_{v_{i-k}} \cup \{a_i, b_i\}$ for all $k+1 \leq i \leq 2k$, where $|B_i| = |A_i|$ $(1 \leq i \leq k)$, and $A_{u'_i} = \{c_i, d_i, e_i\}$ for $1 \leq i \leq k$. We can think of variables v'_i and v'_{i+k} as having the same domain except for the special values corresponding to a_i, b_i. The role of the variables u'_i is to ensure that a_i, b_i cannot be part of any solution to I'.

- For all $1 \leq i \leq 2k$, for all $1 \leq j \leq 2k$ such that i and j are not equal modulo k, a_i and b_i are compatible with all points of $A_{v'_j}$. For all $1 \leq i \leq k$: a_i is compatible with a_{i+k} and incompatible with all other points of $A_{v'_{i+k}}$; a_{i+k} is incompatible with all points in $A_{v'_i} \setminus \{a_i\}$; b_i is compatible with b_{i+k} and incompatible with all other points of $A_{v'_{i+k}}$; b_{i+k} is incompatible with all points in $A_{v'_i} \setminus \{b_i\}$.

- For all $1 \leq i \leq k$: c_i is incompatible with $a_i, b_i, a_{i+k}, b_{i+k}$ and compatible with all other points of A'; d_i is incompatible with all points in $(A_{v'_i} \setminus \{b_i\}) \cup (A_{v'_{i+k}} \setminus \{a_{i+k}\})$ and compatible with all other points in A'; e_i is incompatible with all points in $(A_{v'_i} \setminus \{a_i\}) \cup (A_{v'_{i+k}} \setminus \{b_{i+k}\})$ and compatible with all other points in A'.

- For all $1 \leq i < j \leq k$, for all $a \in A_{v_i}$, for all $b \in A_j$, $cpt'(a, b) = cpt(a, b)$. For all $1 \leq i \leq k$, v_{i+k} is a copy of v_k in I' on $(A_{v_i}, A_{v'_{i+k}} \setminus \{a_{i+k}, b_{i+k}\})$.

The points a_i, b_i, d_i, e_i do not belong to any solution to the sub-instance of I' on variables v'_i, v'_{i+k}, u'_i, whereas c_i is compatible with all points in the original instance I. Furthermore, apart from these special points, variables v'_i, v'_{i+k} are just copies of variable v_i. Thus, by construction, I has a solution if and only I' has a solution.

We will now show that the existential pattern $\exists T_3$ cannot occur on any a_i, with $1 \leq i \leq 2k$. Suppose that there is some i, with $1 \leq i \leq 2k$, such that the existential pattern $\exists T_3$ occurs on a_i. Let v be the variable of a_i. Since $\exists T_3$ occurs on a_i, there is a variable v' and a point $a' \in A_{v'}$ such that a_i and a' are incompatible. By construction, v' can only be one of the following variables: v'_{i+k} (or v'_{i-k} if $i > k$) and u'_i (or u'_{i-k} if $i > k$). Since $\exists T_3$ occurs on a_i, there is a point in A_v which is compatible with two different points in $A_{v'}$. However, from the second and fourth bullet points we know that there is no point in A_v compatible with two different points in $A_{v'_i}$ ($A_{v'_{i-k}}$ if $i > k$), and from the third bullet point we also know that there is no point in A_v compatible with two different points in $A_{u'_i}$ ($A_{u'_{i-k}}$ if $i > k$). So $\exists T_3$ cannot occur on a_i. So the existential pattern $\exists T_3$ cannot occur on any a_i, with $1 \leq i \leq 2k$.

Similarly, it is easy to verify that the existential pattern $\exists T_3$ does not occur on c_i (for all $1 \leq i \leq k$). Hence $\exists T_3$ does not appear in I'. It follows that $\exists T_3$ is NP-complete. □

Let $\exists \text{sub}T_1 = \langle V, A, var, E, cpt, \{a_0\}\rangle$ be the existential pattern shown on the left of Figure 36 and defined by $V = \{v_0, v_1, v_2\}$, $A = \{a_0, a_1, a_2, a_3\}$, $var(a_0) = v_0$, $var(a_1) = var(a_2) = v_1$, $var(a_3) = v_2$, $E = \{\{a_0, a_1\}, \{a_1, a_3\}, \{a_2, a_3\}\}$, $cpt(a_0, a_1) = cpt(a_1, a_3) = T$ and $cpt(a_2, a_3) = F$.

Lemma 33. *$\exists subT_1$ is NP-Complete.*

Proof. Let I be an arc-consistent binary CSP instance on variables $v_1, \ldots, v_n$, where $n > 3$. Let I' be the CSP instance on variables $v'_1, \ldots, v'_n$ with compatibility function cpt' such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i\} \cup \{b_{ij} \mid j = 1, \ldots, i-1, i+1, \ldots, n\}$ for all $1 \leq i \leq n$.
- For all $1 \leq i, j \leq k$ with $i \neq j$, for all $p \in A_{v'_i}$ and for all $q \in A_{v'_j} \setminus A_{v_j}$, $cpt'(p, q) = T$ if and only if $p = b_{ij}$ or $q = b_{ji}$.
- For all $1 \leq i < j \leq n$, for all $a \in A_{v_i}$, for all $b \in A_{v_j}$, $cpt'(a, b) = cpt(a, b)$.

It is easy to verify that none of the points a_i or b_{ij} belong to a solution to any 4-variable sub-instance of I'. This implies that the solutions to I' are exactly the solutions I.

To complete the proof, it remains to show that for each $i = 1, \ldots, n$, $\exists \text{sub}T_1$ does not occur on a_i in I'. Let v'_i, v'_j, v'_k be any three distinct variables in I'. The point a_i is only compatible with b_{ji} in $A_{v'_j}$ which is only compatible with b_{kj} in $A_{v'_k}$. Since b_{kj} is compatible with all points in $A_{v'_j}$, the existential pattern $\exists \text{sub}T_1$ does not occur on a_i in I'. □

Let $\exists T_4 = \langle V, A, var, E, cpt, \{a_0\}\rangle$ be the existential pattern shown on the right of Figure 36 and defined by $V = \{v_0, v_1, v_2\}$, $A = \{a_0, a_1, a_2, a_3, a_4\}$, $var(a_0) = var(a_1) = var(a_2) = v_0$, $var(a_3) = v_1$, $var(a_4) = v_2$, $E = \{\{a_0, a_4\}, \{a_1, a_3\}, \{a_1, a_4\}, \{a_2, a_3\}, \{a_2, a_4\}\}$, $cpt(a_1, a_3) = cpt(a_1, a_4) = cpt(a_2, a_4) = T$ and $cpt(a_0, a_4) = cpt(a_2, a_3) = F$.

Lemma 34. *$\exists T_4$ is NP-Complete.*

Proof. Let $I = \langle V, A, var, E, cpt\rangle$ be an arc-consistent binary CSP instance on variables $v_1, \ldots, v_n$. We will construct an equivalent instance I' in which we add an assignment a_i for each variable so that $\exists T_4$ does not occur on a_i. For each such point a_i, we will also add a 3-variable gadget to prevent a_i from being part of a solution. Let $I' = \langle V', A', var', E', cpt'\rangle$ be the CSP instance on variables $v'_1, \ldots, v'_n$, $w'_1, \ldots, w'_n$, $x'_1, \ldots, x'_n$, $y'_1, \ldots, y'_n$ such that:

- $A_{v'_i} = A_{v_i} \cup \{a_i\}$ for all $1 \leq i \leq n$.
- For all $1 \leq i \leq n$, $|A_{w'_i}| = |A_{v'_i}|$ and the constraint between v'_i and w'_i is a permutation constraint, i.e. there is a bijection $\pi : A_{v'_i} \to A_{w'_i}$ such that $\forall p \in A_{v'_i}$, $\forall q \in A_{w'_i}$, $cpt'(p, q) = T$ if and only if $q = \pi(p)$. For all $1 \leq i \leq n$, we denote $\pi(a_i)$ by b_i.
- For all $1 \leq i, j \leq k$ with $i \neq j$: $A_{x'_i} = \{c_i, d_i\}$ and $A_{y'_i} = \{e_i, f_i\}$; the point c_i is compatible with all points in A' except e_i; the point e_i is compatible with all points in A' except c_i; the point d_i is compatible with all points in A' except b_i and f_i; the point f_i is compatible with all points in A' except b_i and d_i. This implies that b_i, and hence a_i cannot be part of any solution on the variables v'_i, w'_i, x'_i, y'_i.
- For all $1 \leq i \neq j \leq n$: for all $p \in A_{v_i}$, for all $q \in A_{v_j}$, $cpt'(p, q) = cpt(p, q)$; for all $q \in A_{v'_j}$, $cpt(a_i, q) = T$.
- For all $1 \leq i < j \leq n$, for all $p \in A_{w_i}$, for all $q \in A_{w_j}$, $cpt(p, q) = T$.

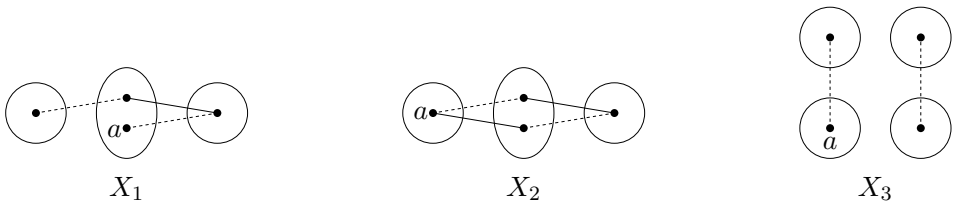


Figure 37: Three tractable existential patterns.

For all $1 \leq i \leq n$, let g_i be any point in $A_{w'_i} \setminus \{b_i\}$. By construction, the existential pattern $\exists T_4$ does not occur on any of the points a_i, g_i, c_i or e_i in I'. Hence $\exists T_4$ does not appear in the instance I'.

For all $1 \leq i \leq n$, the point a_i cannot be extended to a solution to the sub-instance on variables v'_i, w'_i, x'_i, y'_i, whereas all other points in $A_{v'_i}$ can. It follows that the solutions to I' are exactly the solutions to I. Hence $\exists T_4$ is NP-complete. □

4.1.5 Existential Patterns on Two Constraints

Definition 34 (irreducible existential pattern). *We say that an existential pattern P is* irreducible *if we cannot apply merging or dp-elimination to P.*

Lemma 35. *Let $P = \langle V, A, var, E, cpt\rangle$ be a pattern and $P' = \langle V, A, var, E, cpt, \{a\}\rangle$ be an existential version of P. Then P' is tractable only if P is tractable.*

Proof. The result follows directly from Definition 16 of extension and Corollary 2. □

Let $X_1 = \langle V, A, var, E, cpt, \{a\}\rangle$ be the following existential pattern (shown on the left of Figure 37): $V = \{v_0, v_1, v_2\}$, $A = \{a, b, c, d\}$, $var(a) = var(b) = v_0$, $var(c) = v_1$, $var(d) = v_2$, $E = \{\{a, c\}, \{b, c\}, \{b, d\}\}$, $cpt(a, c) = cpt(b, d) = F$ and $cpt(b, c) = T$.

Let $X_2 = \langle V, A, var, E, cpt, \{a\}\rangle$ be the following existential pattern (shown in the middle of Figure 37): $V = \{v_0, v_1, v_2\}$, $A = \{a, b, c, d\}$, $var(a) = v_0$, $var(b) = var(c) = v_1$, $var(d) = v_2$, $E = \{\{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}\}$, $cpt(a, b) = cpt(c, d) = F$ and $cpt(a, c) = cpt(b, d) = T$.

Let $X_3 = \langle V, A, var, E, cpt, \{a\}\rangle$ be the following existential pattern (shown on the right of Figure 37): $V = \{v_0, v_1, v_2, v_3\}$, $A = \{a, b, c, d\}$, $var(a) = v_0$, $var(b) = v_1$, $var(c) = v_2$, $var(d) = v_3$, $E = \{\{a, b\}, \{c, d\}\}$, $cpt(a, b) = cpt(c, d) = F$.

We say that an existential pattern is a *singleton existential pattern* if its set of existential points is a singleton. We first characterize the tractability of irreducible singleton 2-constraint existential patterns. This will then directly lead to a dichotomy for general existential patterns.

Theorem 2. *Let P=$\langle V, A, var, E, cpt, \{a_P\}\rangle$ be an irreducible singleton existential pattern on two constraints. Then P is tractable if and only if P is a sub-pattern of one of the existential patterns X_1, X_2, X_3.*

Proof. $\Rightarrow$: Let $P = \langle V, A, var, E, cpt, \{a_P\}\rangle$ be a tractable irreducible existential pattern on two constraints. A two-constraint existential pattern involves either three or four variables. From Lemma 35 and Theorem 1, all potentially-tractable irreducible singleton existential patterns on

four variables are sub-patterns of X_3. Therefore we only need to consider two-constraint existential patterns on three variables.

By Lemma 35 and Theorem 1, we only need to consider patterns P such that the corresponding non-existential pattern $P' = \langle V, A, var, E, cpt\rangle$ is a sub-pattern of one of T_1, T_2, T_3, T_4, T_5.

If P' is a sub-pattern of T_1, then the irreducible singleton 3-variable existential pattern P must contain one of V$-$Side, V$-$Middle, V$+-$ or $\exists$subT_1, and hence by Lemma 29, Lemma 31 and Lemma 33 P is intractable.

If P' is a sub-pattern of T_2, then the irreducible singleton 3-variable existential pattern P must contain one of V$-$Side, V$-$Middle, V$+-$, V+Side, V+Middle, ExpandedV+ or $\exists$subT_1. Hence, by Lemma 29, Lemma 30, Lemma 31 and Lemma 33, P is intractable.

If P' is a sub-pattern of T_3, then the irreducible singleton 3-variable existential pattern P *either* contains one of V+Side, V+Middle, ExpandedV+, V$+-$, or $\exists T_3$ *or* is a subpattern of X_1 or X_2. Hence, by Lemma 30, Lemma 31 and Lemma 32, P is either intractable or is a subpattern of X_1 or X_2.

If P' is a sub-pattern of T_4, then the irreducible singleton 3-variable existential pattern P *either* contains one of V+Side, V+Middle, ExpandedV+, V$+-$, $\exists$subT_1 or $\exists T_4$ *or* is a subpattern of X_1 or X_2. Hence, by Lemma 30, Lemma 31, Lemma 33 and Lemma 34, P is either intractable or is a subpattern of X_1 or X_2.

If P' is a sub-pattern of T_5, then the irreducible singleton 3-variable existential pattern P *either* contains V$+-$ *or* is a subpattern of X_1 or X_2. Hence, by Lemma 31, P is either intractable or is a subpattern of X_1 or X_2.

$\Leftarrow$: We now give the tractability proofs for the patterns X_1, X_2, X_3.

Proof of tractability of X_1 : Let $I = \langle V, A, var, E, cpt\rangle$ be an arc-consistent CSP instance such that X_1 does not appear in I. So $\forall v_i \in V$, $\exists a_i \in A_{v_i}$ such that X_1 does not occur on a_i. Suppose that we have a partial solution $S_k = \{s_1 \in A_{v_1}, \ldots, s_k \in A_{v_k}\}$, with $0 \leq k < |V|$. If a_{k+1} is compatible with all s_i for $1 \leq i \leq k$, then $S_k \cup a_{k+1}$ is a partial solution for variables $(v_1, \ldots, v_{k+1})$. Suppose that for some $1 \leq i \leq k$, we have that s_i and a_{k+1} are incompatible. By arc-consistency, we know there is $b \in A_{v_{k+1}}$ such that s_i and b are compatible. Since X_1 does not occur on a_{k+1}, b is compatible with all points in $A \backslash (A_{v_i} \cup A_{v_{k+1}})$, in particular all s_j for $j \neq i$. So b is compatible with all points in S_k. So $S_k \cup b$ is a partial solution for variables $(v_1, \ldots, v_{k+1})$. So if we have a partial solution for I on k variables, then we also have a partial solution for I on $k+1$ variables. Hence, assuming $A \neq \emptyset$, there is always a solution for I. So X_1 is tractable.

Proof of tractability of X_2 : Let $I = \langle V, A, var, E, cpt\rangle$ be an arc-consistent CSP instance such that X_2 does not appear in I. So $\forall v_i \in V$, $\exists a_i \in A_{v_i}$ such that X_2 does not occur on a_i. Suppose that we have a partial solution $S_k = \{s_1 \in A_{v_1}, \ldots, s_k \in A_{v_k}\}$, with $0 \leq k < |V|$. Let Y be the set $i \leq k$ such that s_i and a_{k+1} are compatible and let $\overline{Y}$ be the set of $i \leq k$ such that s_i and a_{k+1} are compatible. By arc consistency, $\forall i \in \overline{Y}$, $\exists t_i$ such that t_i and a_{k+1} are compatible. Let

$S' = \{s'_1 \in A_{v_1}, \ldots, s'_{k+1} \in A_{v_{k+1}}\}$ with

$$s'_i = \begin{cases} a_i & \text{if } i = k+1 \\ s_i & \text{if } i \in Y \\ t_i & \text{if } i \in \overline{Y} \end{cases}$$

Let $i \in Y$ and $j \in \overline{Y}$. Since S is a partial solution, s_i and s_j are compatible. We know that a_{k+1} is compatible with t_j and incompatible with s_j. Since X_2 does not occur on a_{k1}, s_i and t_j are compatible.
Let $i, j \in \overline{Y}$. From the argument in the previous paragraph, we know that s_i and t_j are compatible. We also know that a_{k+1} is compatible with t_i and incompatible with s_j. Since X_2 does not occur on a_{k+1}, t_i and t_j are compatible.
So all the points in S' are compatible with each other. So S' is a partial solution for variables $(v_1, \ldots, v_{k+1})$. So if we have a partial solution for I on k variables, then we also have a partial solution for I on $k+1$ variables. Hence, assuming $A \neq \emptyset$, there is always a solution for I. So X_2 is tractable.

Proof of tractability of X_3 : Let $I = \langle V, A, var, E, cpt \rangle$ be an arc-consistent CSP instance such that X_3 does not appear in I. So $\forall v_i \in V$, $\exists a_i \in A_{v_i}$ such that X_3 does not occur on a_i. If all a_i are compatible with all points in I, then the set of all a_i is a solution for I. Otherwise, let i and j be such that $\exists b \in A_{v_j}$ such that a_i and b are incompatible. Since X_3 does not occur on a_i, there is no incompatibility edge between two points of $A \setminus (A_{v_i} \cup A_{v_j})$. Thus we can perform the fusion of v_i and v_j into a new variable v_{ij} such that points in $A_{v_{ij}}$ correspond to compatibility edges between v_i and v_j. Since there is no incompatibility edge between two points outside of $A_{v_{ij}}$, applying arc consistency on v_{ij} will determine whether there is a solution for I. □

4.1.6 The Dichotomy

We can now combine Theorems 1 and 2 to obtain a complete dichotomy for irreducible 2-constraint flat and existential patterns.

Theorem 3. *Let $P = \langle V, A, var, E, cpt, e \rangle$ be an irreducible flat or existential pattern on two constraints. Then $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ is solvable in polynomial-time if P is a sub-pattern of one of the patterns T_1, T_2, T_3, T_4, X_1, X_2, X_3; if not $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ is NP-complete.*

Proof. We first make the observation that for flat patterns, $\mathrm{CSP}_{\mathrm{AC}}(\overline{P})$ is solvable in polynomial time if and only if $\mathrm{CSP}(\overline{P})$ is solvable in polynomial time, since flat patterns cannot be introduced by establishing arc consistency. Thus the flat case corresponds exactly to the dichotomy for flat patterns given in Theorem 1. Note that the patterns T_5 and $2I$ are reducible to sub-patterns, respectively, of X_2 and X_3 which is why we do not explicitly mention them in the statement of the theorem.

The case $|e| = 1$ corresponds exactly to Theorem 2. For the case $|e| > 1$, by Lemma 35, we only need to consider existential versions of sub-patterns of T_1, T_2, T_3, T_4, T_5, $2I$. But all existential

patterns P with $|e| > 1$ which are sub-patterns of one of T_1, T_2, T_3, T_4, T_5, $2I$ must contain either V+− or V−Side and hence are NP-complete by Lemma 29 and Lemma 31. □

We have investigated the computational complexity of classes of binary CSP instances defined by forbidding 2-constraint patterns. We have given a dichotomy for irreducible 2-constraint patterns which has brought to light several novel tractable classes.

4.2 Forbidding Max-CSPs Subproblems

In this section we study Max-CSP patterns.

4.2.1 Definitions and Basic Properties

Definition 35 (max-CSP pattern). *A* Max-CSP pattern *is a pattern $P = \langle V, A, var, E, cost\rangle$ where $cost$ is a cost function from E to $\{0, 1\}$, and such that $\forall v \in V$, $\forall a, b \in A_v$, a and b are considered distinct.*

Binary Max-CSP instances are defined similarly.

Definition 36 (max-CSP instance). *A* Max-CSP instance *is a Max-CSP pattern $I = \langle V, A, var, E, cost\rangle$ where $E = \{\{a, b\} \mid a \in A, b \in A, var(a) \neq var(b)\}$ and $cost$ is a cost function from E to $\{0, 1\}$.*

A solution S for a Max-CSP instance I is a set of k assignments, where k is the number of variables in V, such that no two assignments from S belong to the same variable. The cost $C(S)$ of a solution S is the sum of all $cost(a, b)$ for all $a, b \in S$. An optimal solution S_0 for a Max-CSP instance I is a solution of minimal cost. Solving a Max-CSP instance I is finding an optimal solution for I.

In this section, we only consider complete patterns, or subproblems. A subproblem P is simply a binary Max-CSP instance.

Definition 37 (occurence in a Max-CSP subproblem). *We say that a Max-CSP subproblem P* occurs *in a Max-CSP subproblem P' (or that P'* contains *P) if P' is isomorphic to a pattern Q, such that Q is the extension of P, with extension being the following operation:*

extension *A subproblem P is a sub-instance of a subproblem Q (and Q an extension of P): if $P = \langle V_P, A_P, var_P, E_P, cost_P\rangle$ and $Q = \langle V_Q, A_Q, var_Q, E_Q, cost_Q\rangle$, then $V_P \subseteq V_Q$, $A_P \subseteq A_Q$, $var_P = var_Q|_{A_P}$, $E_P \subseteq E_Q$, $cost_P = cost_Q|_{E_P}$.*

To illustrate this notion, consider the instance I and the three subproblems P, P', P'' shown in Fig. 38. In this example, subproblem P occurs in I with the corresponding isomorphism $p \mapsto a$, $q \mapsto b$, $r \mapsto c$. Similarly, P' occurs in I with the corresponding isomorphism $t \mapsto c$, $u \mapsto d$, $v \mapsto a$. On the other hand, P'' does not occur in I.

Since we are only considering subproblems, the definition of occurence in a Max-CSP subproblem does not include merging. Also, since the notion of arc consistency is not applicable to Max-CSP problems, the operation of dp-elimination will not be used in this section.

In this section we denote by $\mathcal{F}(P)$ the set of Max-CSP instances in which the subproblem P is forbidden, i.e. does not occur. Thus if I, P' and P'' are as shown in Fig. 38, $I \in \mathcal{F}(P'')$ but

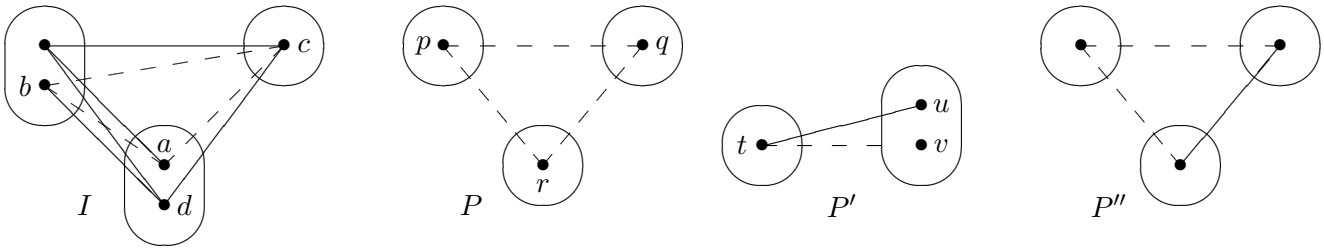


Figure 38: The instance I contains P and P' as subproblems but not P''.

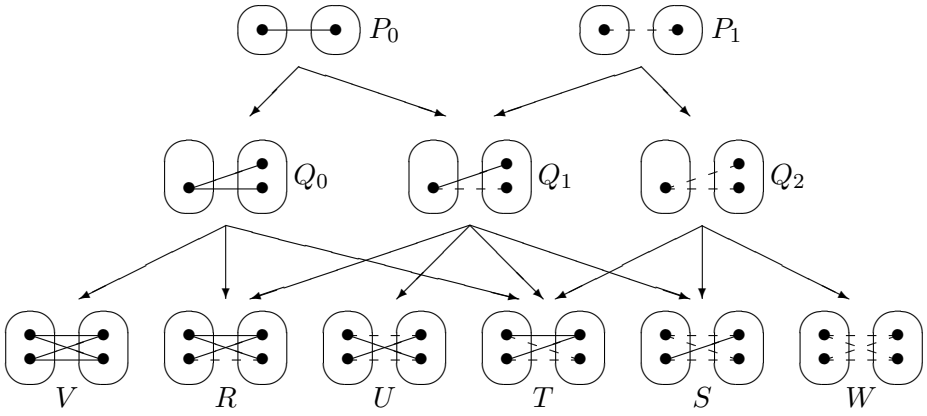


Figure 39: Subproblems on two variables (showing inclusions between subproblems).

$I \notin \mathcal{F}(P')$. If $\Sigma = \{P_1, \ldots, P_s\}$ is a set of subproblems, then we use $\mathcal{F}(\Sigma)$ or $\mathcal{F}(P_1, \ldots, P_s)$ to denote the set of Max-CSP instances in which no subproblem $P_i \in \Sigma$ occurs. The following lemma follows from the above definitions, by transitivity of the occurrence relation.

Lemma 36. *If $\forall P \in \Sigma_1$, $\exists Q \in \Sigma_2$ such that Q occurs in P, then $\mathcal{F}(\Sigma_2) \subseteq \mathcal{F}(\Sigma_1)$.*

We say that $\mathcal{F}(\Sigma)$ is *tractable* if there is a polynomial-time algorithm to solve it. We say that $\mathcal{F}(\Sigma)$ is *intractable* if it is NP-hard. Suppose that $\mathcal{F}(\Sigma_1) \subseteq \mathcal{F}(\Sigma_2)$. Clearly, $\mathcal{F}(\Sigma_1)$ is tractable if $\mathcal{F}(\Sigma_2)$ is tractable and $\mathcal{F}(\Sigma_2)$ is intractable if $\mathcal{F}(\Sigma_1)$ is intractable.

4.2.2 Dichotomy for Forbidding a Single Subproblem

We assume that $P \neq NP$. Our aim is to characterize the tractability of $\mathcal{F}(P)$ for all subproblems P. We first show that we only need to consider subproblems with domains of size at most 2.

Lemma 37. *Let P be a subproblem with three or more values in the domain of some variable and let $\mathcal{F}(P)$ be the set of Max-CSP instances in which the subproblem P is forbidden. Then $\mathcal{F}(P)$ is intractable.*

Proof. Max-Cut is intractable and can be reduced to Max-CSP on Boolean domains (Creignou et al., 2001). Thus $\mathcal{F}(P)$ is intractable since it includes all instances of Max-CSP on Boolean domains. □

We now consider the subproblems on just two variables shown in Figure 39. Modulo independent permutations of the variables and of the two domains, these are the only possible subproblems with domains of size at most 2.

Lemma 38. *If Q_1 is the subproblem shown in Fig. 39, then $\mathcal{F}(Q_1)$ is tractable.*

Proof. Let I be an instance in $\mathcal{F}(Q_1)$. It is easy to see that all binary cost functions between any pair of variables in I must be constant. Hence I is equivalent to a trivial Max-CSP instance with no binary cost functions. □

Lemma 39. *If Q_0 and U are as shown in Fig. 39, then $\mathcal{F}(\{Q_0, U\})$ is intractable.*

Proof. Max-Cut can be coded as Max-CSP over Boolean domains in which all constraints are of the form $X_i \neq X_j$. We can replace each constraint $X_i \neq X_j$ by an equivalent gadget G with two extra variables Y_{ij}, Z_{ij}, where G is given by $\neg X_i \wedge Y_{ij}$, $\neg Y_{ij} \wedge \neg X_j$, $X_i \wedge \neg Z_{ij}$, $Z_{ij} \wedge X_j$. It is easily verified that placing the gadget G on variables X_i, X_j is equivalent to imposing the constraint $X_i \neq X_j$; when $X_i = X_j$ at most one of these constraints can be satisfied and when $X_i \neq X_j$ at most two constraints can be satisfied.

For each pair of variables X, X' in the resulting instance of Max-CSP such that there is no constraint between X and X', we place a binary constraint on X, X' of constant cost 1. In the resulting Max-CSP instance, there are no two zero costs in the same binary cost function. Thus, this polynomial reduction from Max-Cut produces an instance in $\mathcal{F}(\{Q_0, U\})$. Intractability of $\mathcal{F}(\{Q_0, U\})$ then follows from the NP-hardness of Max-Cut. □

Lemma 40. *If Q_2 and U are as shown in Fig. 39, then $\mathcal{F}(\{Q_2, U\})$ is intractable.*

Proof. As in the proof of Lemma 39, the proof is again by a polynomial reduction from Max-Cut. This time each constraint $X_i \neq X_j$ is replaced by the gadget G' where G' is $\neg X_i \vee Y_{ij}$, $\neg Y_{ij} \vee \neg X_j$, $X_i \vee \neg Z_{ij}$, $Z_{ij} \vee X_j$. When $X_i = X_j$ at most three of these constraints can be satisfied and when $X_i \neq X_j$ all four constraints can be satisfied.

For each pair of variables X, X' in the resulting instance of Max-CSP such that there is no constraint between X and X', we place a binary constraint on X, X' of constant cost 0. The resulting instance is in $\mathcal{F}(\{Q_2, U\})$. □

This provides us with a dichotomy for subproblems on just two variables.

Theorem 4. *If P is a 2-variable binary Max-CSP subproblem, then $\mathcal{F}(P)$ is tractable if and only if P occurs in Q_1 (shown in Fig. 39).*

Proof. By Lemma 37, we only need to consider subproblems in which each domain is of size at most two.

Since each of P_0 and P_1 occur in Q_1, it follows from Lemma 38 and Lemma 36 that $\mathcal{F}(P_0)$ and $\mathcal{F}(P_1)$ are also tractable. Since Q_0 occurs in R, T, V and Q_2 occurs in S, W, it follows from Lemmas 39, 40 and Lemma 36 that $\mathcal{F}(Q_0)$, $\mathcal{F}(Q_2)$, $\mathcal{F}(R)$, $\mathcal{F}(S)$, $\mathcal{F}(T)$, $\mathcal{F}(U)$, $\mathcal{F}(V)$, $\mathcal{F}(W)$ are all intractable. This covers all the possible subproblems with domains of size at most 2 as shown in Fig. 39. □

We recall the following result which follows directly from Theorem 5 of (Cooper & Živný, 2011c).

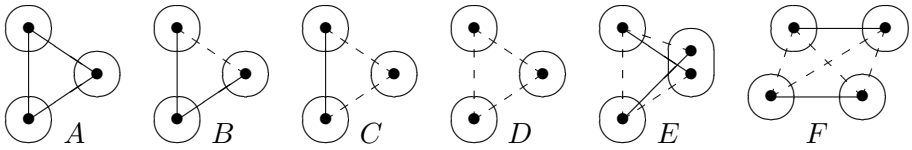


Figure 40: Subproblems on three or four variables.

Lemma 41. *A class of binary Max-CSP instances defined by forbidding a single subproblem comprised of a triangle of three assignments to three distinct variables is tractable if and only if the three binary costs are 0,1,1.*

Binary Max-CSP instances in which the triple of binary costs 0,1,1 does not occur in any triangle satisfy the so-called joint-winner property (Cooper & Živný, 2011b). This class has been generalized to the hierarchically-nested convex class which is a tractable class of Valued CSP instances involving cost functions of arbitrary arity (Cooper & Živný, 2011a). The following corollary is just a translation of Lemma 41 into the notation of forbidden subproblems.

Corollary 5. *Let A, B, C, D be the subproblems shown in Fig. 40. Then $\mathcal{F}(C)$ is tractable, but $\mathcal{F}(A)$, $\mathcal{F}(B)$, and $\mathcal{F}(D)$ are intractable.*

Lemma 42. *Given the subproblem E shown in Fig. 40 and the set $\mathcal{F}(E)$ of Max-CSP instances in which the subproblem E is forbidden, then $\mathcal{F}(E)$ is intractable.*

Proof. The constraint graph of a Max-CSP instance is the graph $\langle V, E\rangle$ where V is the set of variables and $\{X_i, X_j\} \in E$ if there is a pair of assignments p, q with $var(p) = X_i$, $var(q) = X_j$ and such that $cost(p, q) = 1$. Clearly the constraint graph of any instance in which E occurs contains a triangle. Max-Cut is NP-hard even on triangle-free graphs (Lewis & Yannakakis, 1980). Any such instance of Max-Cut coded as an instance I of binary Max-CSP does not contain E as a subproblem since the constraint graph of I is triangle-free. Hence $\mathcal{F}(E)$ is intractable. □

Lemma 43. *The only 3-variable subproblem P for which the set $\mathcal{F}(P)$ is tractable is the subproblem C shown in Fig. 40.*

Proof. Let P be a 3-variable subproblem. For $\mathcal{F}(P)$ to be tractable, P must not have as a subproblem any of Q_0, Q_2, A, B, D, E which have all been shown to define intractable classes (Lemmas 39, 40, 42 and Corollary 5). The only 3-variable subproblem which does not contain any of Q_0, Q_2, A, B, D, E is C. The result then follows from Lemma 36. □

It turns out that the tractable classes we have already identified, defined by forbidden subproblems on two or three variables, are the only possible tractable classes defined by forbidding a single subproblem. To complete our dichotomy, we require one final lemma.

Lemma 44. *If F is the subproblem shown in Fig. 40, then $\mathcal{F}(F)$ is intractable.*

Proof. It is known that Max-Cut on C_4-free graphs is NP-hard (Kamiński, 2010). To see this, let G be a graph and G' a version of G in which each edge is replaced by a path composed of three

edges. Clearly, G' is C_4-free and the maximum cut of G' is of the same size as the maximum cut of G.

When a Max-Cut instance on a C_4-free graph is coded as a Max-CSP instance I, the subproblem F cannot occur since there can be no length-4 cycles of non-trivial constraints in I. Hence $\mathcal{F}(F)$ is intractable. □

By looking at all possible combinations of edges in a subproblem, it is possible to show that F is the only subproblem on four variables in which neither A, B, D nor E shown in Fig. 40 occur. Since $\mathcal{F}(A)$, $\mathcal{F}(B)$, $\mathcal{F}(D)$ and $\mathcal{F}(E)$ are intractable, then from Lemma 44 the classes of Binary Max-CSP instances defined by forbidding a single subproblem on four or more variables are all intractable and we can now state our dichotomy.

Theorem 5. *If P is a binary Max-CSP subproblem, then $\mathcal{F}(P)$ is tractable if and only if P occurs either in Q_1 (shown in Fig. 39) or in C (shown in Fig. 40).*

It follows that $\mathcal{F}(P)$ is tractable only for $P = P_0$, P_1, Q_1 (shown in Fig. 39) or C (shown in Fig. 40). It follows that the only non-trivial tractable class defined by a forbidden subproblem corresponds to the set of instances satisfying the so-called joint-winner property. The joint-winner property encompasses, among other things, codings of non-intersecting graph-based or variable-based SoftAllDiff constraints together with arbitrary unary constraints (Cooper & Živný, 2011b). It is worth pointing out that Theorem 5 is independent of the presence of unary cost functions, in the sense that tractable classes remain tractable when arbitrary unary costs are allowed and NP-hardness results are valid even if no unary costs are allowed.

4.2.3 Requirements for the Tractability of a Set of Subproblems

In this section we give a necessary condition for a forbidden set of subproblems to define a tractable class of binary Max-CSP instances.

Definition 38 (Boolean problem). *A subproblem (or an instance) P is* Boolean *if the size of the domain of each variable in P is at most two.*

Definition 39 (negative edge pair). *A* negative edge pair *is a set of variable-value assignments p, q, r, s such that $var(p) = var(r) \neq var(q) = var(s)$, $cost(p, q) = cost(r, s) = 1$ and $p \neq r$. A* positive edge pair *is a set of variable-value assignments p, q, r, s such that $var(p) = var(r) \neq var(q) = var(s)$, $cost(p, q) = cost(r, s) = 0$ and $p \neq r$.*

Definition 40 (negative cycle). *A* negative cycle *is a set of variable-value assignments $p_1, p'_1, \ldots, p_m, p'_m$, with $m > 2$, such that the variables $var(p_i)$ $(i = 1, \ldots, m)$ are all distinct and $var(p_i) = var(p'_i)$ $(i = 1, \ldots, m)$, $cost(p_i, p'_{i+1}) = 1$ $(i = 1, \ldots, m - 1)$ and $cost(p_m, p'_1) = 1$. A* positive cycle *is a set of assignments $p_1, p'_1, \ldots, p_m, p'_m$, with $m > 2$, such that the variables $var(p_i)$ $(i = 1, \ldots, m)$ are all distinct and $var(p_i) = var(p'_i)$ $(i = 1, \ldots, m)$, $cost(p_i, p'_{i+1}) = 0$ $(i = 1, \ldots, m - 1)$ and $cost(p_m, p'_1) = 0$.*

Definition 41 (negative pivot point). *A* negative pivot point *is a variable-value assignment p such that there are two assignments q, r with $var(p)$, $var(q)$, $var(r)$ all distinct and $cost(p, q) = cost(p, r) = 1$.*

A positive pivot point *is an assignment p such that there are two assignments q, r with $var(p)$, $var(q)$, $var(r)$ all distinct and $cost(p, q) = cost(p, r) = 0$.*

Proposition 1. *If Σ is a finite set of subproblems, then $\mathcal{F}(\Sigma)$ is tractable only if*

1. *There is a Boolean subproblem $P \in \Sigma$ such that P contains no negative edge pair, no negative cycle and at most one negative pivot point, and*
2. *There is a Boolean subproblem $Q \in \Sigma$ such that Q contains no positive edge pair, no positive cycle and at most one positive pivot point, and*
3. *There is a Boolean subproblem $B \in \Sigma$ such that B contains neither Q_0 nor Q_2 (as shown in Fig. 39).*

Proof. Suppose that condition (1) is not satisfied. We will show that $\mathcal{F}(\Sigma)$ is NP-hard. Let t be an odd integer strictly greater than the number of variables in any subproblem in Σ. As in Lemma 40 the proof is by a polynomial reduction from Max-Cut. This time each constraint $X_i \neq X_j$ is replaced by the gadget G_t where G_t is $\neg X_i \vee Y_1$, $\neg Y_k \vee Y_{k+1}$ ($k = 1, \ldots, t-1$), $\neg Y_t \vee \neg X_j$, and $X_i \vee \neg Z_1$, $Z_k \vee \neg Z_{k+1}$ ($k = 1, \ldots, t-1$), $Z_t \vee X_j$. The gadget G_t is equivalent to $X_i \neq X_j$ since when $X_i = X_j$ one of its constraints must be violated, but when $X_i \neq X_j$ all of its constraints can be satisfied. For each pair of variables X, X' in the resulting instance of Max-CSP such that there is no constraint between X and X', we place a binary constraint on X, X' of constant cost 0.

The resulting instance I has no domain of size greater than two, and contains no negative edge pair, no negative cycle of length at most t and no two negative pivot points at a distance at most t. Let $P \in \Sigma$. Since (1) is not satisfied, and by definition of t, either P has a domain of size more than two, or contains a negative edge pair, a negative cycle of length at most t or two negative pivot points at a distance at most t. It follows that P cannot occur in I. Thus, we have demonstrated a polynomial reduction from Max-Cut to $\mathcal{F}(\Sigma)$.

The proof for condition (2) is similar. This time each constraint $X_i \neq X_j$ is replaced by the gadget G'_t given by $\neg X_i \wedge Y_1$, $\neg Y_k \wedge Y_{k+1}$ ($k = 1, \ldots, t-1$), $\neg Y_t \wedge \neg X_j$, and $X_i \wedge \neg Z_1$, $Z_k \wedge \neg Z_{k+1}$ ($k = 1, \ldots, t$), $Z_t \wedge X_j$. The gadget G'_t is equivalent to the constraint $X_i \neq X_j$; when $X_i = X_j$ at most t of its constraints can be satisfied and when $X_i \neq X_j$ at most $t+1$ of its constraints can be satisfied. For each pair of variables X, X' in the resulting instance of Max-CSP such that there is no constraint between X and X', we place a binary constraint on X, X' of constant cost 1.

The resulting instance I has no domain of size greater than two, and contains no positive edge pair, no positive cycle of length at most t and no two positive pivot points at a distance at most t. Let $P \in \Sigma$. If condition (2) is not satisfied, no $P \in \Sigma$ can occur in I. Thus, this polynomial reduction is from Max-Cut to $\mathcal{F}(\Sigma)$.

If condition (3) is not satisfied, then, by Lemma 36, $\mathcal{F}(\Sigma)$ contains all Boolean instances in $\mathcal{F}(Q_0, Q_2)$. But $\mathcal{F}(Q_0, Q_2)$ is equivalent to the set of Boolean instances of Max-CSP in which for each pair of variables X_i, X_j there is a constraint between X_i and X_j with this constraint being either $X_i = X_j$ or $X_i \neq X_j$. This set of Max-CSP instances is equivalent to the 2-Cluster Editing problem whose decision version is known to be NP-complete (Shamir, Sharan, & Tsur, 2004). Hence $\mathcal{F}(\Sigma)$ is NP-hard if condition (3) is not satisfied. □

We have given a dichotomy concerning the tractability of classes of binary Max-CSP instances defined by forbidding a single subproblem. We have also given a necessary condition for the tractability of classes defined by forbidding sets of subproblems.

Classes defined by forbidding (sets of) subproblems are closed under permutations of the set of variables and independent permutations of each variable domain.

4.3 Forbidden Patterns on Three Variables

This section leaves behind Max-CSP patterns to give miscellaneous complexity results for flat CSP patterns on three variables.

4.3.1 Necessary Conditions for Tractability

We first give a lemma which will be used in the proof of the main result in this section.

Lemma 45. *Let P be a pattern on three variables with at least one incompatibility edge in each constraint. Then P is NP-Complete.*

Proof. Let P be a pattern on three variables, such that there is at least one incompatibility edge in each constraint of P. We are going to give a polynomial reduction from the general binary CSP to CSP($\overline{P}$). Let $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ be a binary CSP instance. Let v and w two variables in V_I such that there is a non-trivial constraint between v and w. We then add two additional variables v' and w' to V_I such that:

- $A_{v'}$ is a copy of A_v and $A_{w'}$ is a copy of A_w.
- The constraint between v' and w' is a copy of the constraint between v and w.
- The constraint between v and v' and the constraint between w and w' are equality constraints.
- The constraint between v and w is replaced by a trivial constraint.
- All other constraints between v' or w' on one hand and any variable of V_I on the other hand are trivial constraints.

Let I' be the resulting instance. Suppose that there is a solution S for I. Let a be the point of S in A_v and let b be the point of S in A_w. Let a' be the copy of a in $A_{v'}$, and let b' be the copy of b in $A_{w'}$. Let $S' = S \cup \{a', b'\}$. a and a' satisfy the equality constraint between v and v'. Similarly, b and b' satisfy the equality constraint between w and w'. Since a and b satisfy the constraint between v and w in I, a' and b' satisfy by construction the constraint between v' and w' in I'. Therefore, S' is a solution for I'. So if there is a solution S for I, then there is a solution S' for I'.

Suppose now that there is a solution S' for I'. Let a, a', b' and b be the points of S' in A_v, $A_{v'}$, $A_{w'}$ and A_w respectively. Let $S = S' \backslash \{a', b'\}$. Since S' is a solution for I', and since the constraints between v and v' and between w and w' are equality constraints, a' is the copy of a in $A_{v'}$ and b' is the copy of b in $A_{w'}$. So, since a' and b' satisfy the constraint between v' and w' in I', a and b satisfy

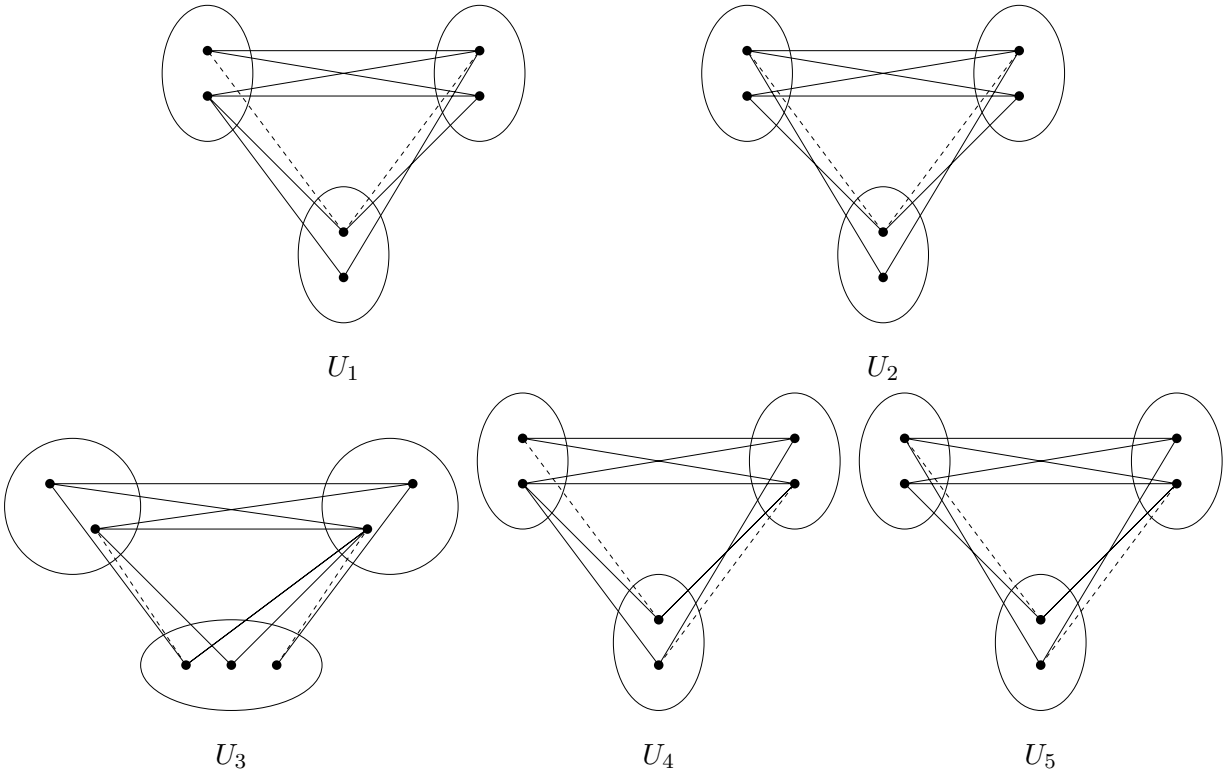


Figure 41: The Set $\mathcal{U}$.

the constraint between v and w in I. So S is a solution for I. So if there is a solution S' for I', then there is a solution S for I.

So I' has the same satisfiability as I. Let $J = \langle V_J, A_J, var_J, E_J, cpt_J \rangle$ be the instance obtained from I after replacing non recursively all non-trivial constraints in I in the way we just described. We know that J has the same satisfiability as I. It remains yet to show that J belongs to $\mathrm{CSP}(\overline{P})$. Let $V' = V_J \backslash V_I$. Suppose that the pattern P occurs in J. So there are three variables v_1, v_2 and v_3 in V_J such that there is an incompability in the constraint between v_1 and v_2, in the constraint between v_2 and v_3, and in the constraint between v_3 and v_1.

Suppose that one of these three variables, say for instance v_1, is in V'. By construction, $\forall v' \in V'$, there is exactly one variable v in V_I such that there is a non-trivial in J constraint between v' and v, there is exactly one variable w' in V' such that there is a non-trivial constraint in J between v' and w', and there is a no non-trivial constraint between v and w'. So none of three variables v_1, v_2 and v_3 can be in V'. So all three variables v_1, v_2 and v_3 are in V_I. But by construction, there are no non-trivial constraint in J between any two variables of V_I. So P cannot occurs in J. So $J \in \mathrm{CSP}(\overline{P})$.

We have reduced the general binary CSP, which is NP-Complete, to $\mathrm{CSP}(\overline{P})$. Therefore, P is NP-Complete. □

Let $\mathcal{U} = \{U_1, U_2, U_3, U_4, U_5\}$, with U_1, U_2, U_3, U_4 and U_5 as pictured in Figure 41.

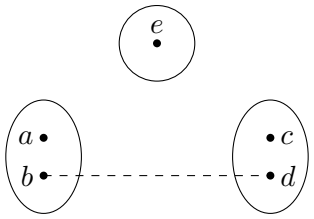


Figure 42: Incompatibility skeleton of type 1.

Proposition 2. *Let P be a non-mergeable tractable pattern on three variables. Then P can be reduced to a sub-pattern of a pattern belonging to U.*

Proof. From Lemma 45, we know that a pattern on three variables with at least one incompatibility edge in each constraint is NP-Complete. From Lemma 6, we know that a pattern with two or more distinct incompatibility edges between the same couple of domains is also NP-Complete. Therefore, if a pattern P on three variables is tractable, then the subpattern formed by its incompatibility edges can take four possible forms:

1. No incompatibility edge at all.
2. One incompatibility edge.
3. Two incompatibility edges sharing a point.
4. Two incompatibility edges not sharing a point.

So for each of the four cases, the only unknown part in a tractable pattern P are the compatibility edges.
In the first case, from Lemma 7, we know that the pattern P is a sub-pattern of the pattern consisting of three compatibility edges forming a triangle. This pattern is itself a sub-pattern of any of the patterns in U.

So there only remains three possible incompatibility skeletons to study, each one implying from Lemma 7 a maximum number of points appearing in the pattern P. We know from Lemmas 8 and 11 that the patterns Z (from Figure 12) and $2V$ (from Figure 22) are both intractable, so by Corollary 1 we must look for patterns in which neither one occurs.

We now consider the incompatibility skeleton of type 1, shown in Figure 42.

Suppose that there is no compatibility edge between b and c. Then we can merge c and d. If there is no compatibility edge between a and d, then we can merge a and b, and whatever we decide for the compatibility of the two remaining edges, the resulting pattern will be a subpattern of any of the patterns in U. If there is a compatibility edge between a and d, then there also must be a compatibility edge between a and e, otherwise we could remove a by dp-elimination, and whatever we decide for the compatibility of the two remaining edges, the resulting pattern will be a subpattern of U_1.

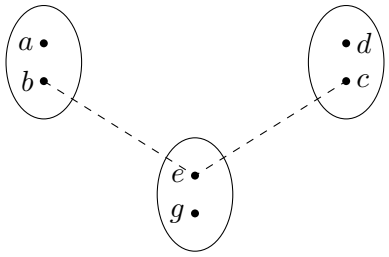


Figure 43: Incompatibility skeleton of type 2.

Similarly, suppose that there is no compatibility edge between a and d. Then we can merge a and b. If there is no compatibility edge between b and c, then we can merge c and d, and whatever we decide for the compatibility of the two remaining edges, the resulting pattern will be a subpattern of any of the patterns in U. If there is a compatibility edge between b and c, then there also must be a compatibility edge between c and e, otherwise we could remove c by dp-elimination, and whatever we decide for the compatibility of the two remaining edges, the resulting pattern will be a subpattern of U_1.

Suppose now that there are both a compatibility edge between b and c and a compatibility edge between a and d. There cannot be a compatibility edge between a and c, because otherwise Z would occur. So there must be a compatibility edge between a and e, otherwise we could remove by dp-elimination. Similarly, there must be a compatibility edge between c and e, oterwise we could remove c by elimination. Whatever we decide for the compatibility of the two remaining edges, the resulting pattern will be a subpattern of U_1.

We now consider the incompatibility skeleton of type 2, shown in Figure 43.

Suppose that there is a compatibility edge between a and g. Then there must be a compatibility edge between a and e, otherwise we could merge a and b. There cannot be a compatibility edge between b and g, otherwise the pattern Z would occur. So there must be a compatibility edge between g and c, otherwise we could merge g and e. Then there cannot be a compatibility edge between d and g, because otherwise $2V$ would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_1.

Symmetrically, suppose that there is a compatibility edge between d and g. Then there must be a compatibility edge between d and e, otherwise we could merge d and c. There cannot be a compatibility edge between c and g, otherwise the pattern Z would occur. So there must be a compatibility edge between g and b, otherwise we could merge g and e. Then there cannot be a compatibility edge between a and g, because otherwise $2V$ would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_1.

Suppose there is no compatibility edge between a and g, nor between d and g. Then whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_2.

We finally consider the incompatibility skeleton of type 3, shown in Figure 44.

Suppose that g is a point in the pattern. Then there must be a compatibility edge between g and b, otherwise we could merge g and e. There must also be a compatibility edge between g and

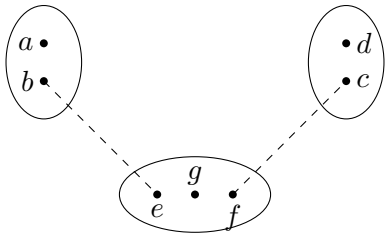


Figure 44: Incompatibility skeleton of type 3.

c, otherwise we could merge g and f. There must be a compatibility edge either between e and c or between b and f, otherwise we could merge e and f. Without loss of generality, we assume that there is a compatibility edge between e and c.

Suppose that a is not in the pattern but d is. Then there must be a compatibility edge between d and f, otherwise we could merge d and c. Then there cannot be a compatibility edge between d and e, nor between d and g, because either way the pattern Z would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_3.

Suppose that a is in the pattern but not d. Then there must be a compatibility edge between a and e, otherwise we could merge a and b. Then there cannot be a compatibility edge between a and g, nor between a and f, because either way the pattern $2V$ would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_3.

Suppose that both a and d are in the pattern. There must be a compatibility edge between d and f, otherwise we could merge d and c. There also must be a compatibility edge between a and e, otherwise we could merge a and b. Then there cannot be a compatibility edge between a and g, a and f, d and e nor d and g, because otherwise either the pattern Z or the pattern $2V$ would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_3.

Suppose that neither a nor d is in the pattern. Then whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_3.

Suppose now that g is not in the pattern. There must be a compatibility edge either between e and c or between b and f, otherwise we could merge e and f. Without loss of generality, we assume that there is a compatibility edge between e and c.

Suppose that neither a nor d is in the pattern. Then whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_4.

Suppose that a is not in the pattern but d is. Then there must be a compatibility edge between d and f, otherwise we coul merge d and c. Then there cannot be a compatibility edge between d and e, otherwise the pattern Z would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_5.

Suppose that a is in the pattern but not d. Then there must be a compatibility edge between a and e, otherwise we could merge a and b. Suppose then that there is a compatibility edge between a and f. Then there cannot be a compatibility edge between b and f, otherwise the pattern Z would occur. Whatever we decide for the compatibility of the remaining edges, the

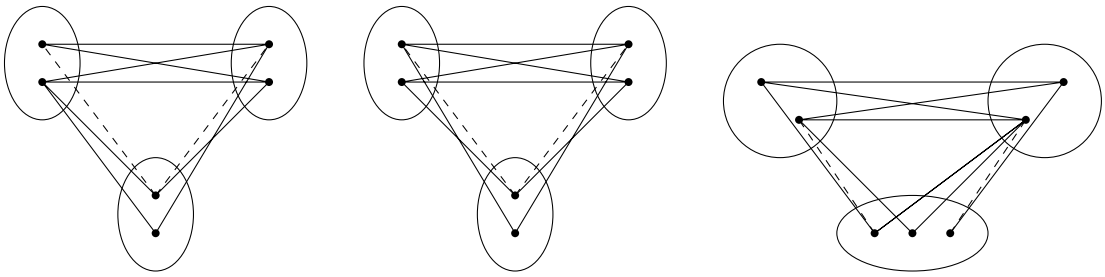
resulting pattern will be a subpattern of U_4. If there is no compatibility edge between a and f, then whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_5.

Suppose that both a and d are in the pattern. There must be a compatibility edge between d and f, otherwise we could merge d and c. Then there cannot be a compatibility edge between d and e, otherwise the pattern Z would occur. There also must be a compatibility edge between a and e, otherwise we could merge a and b. Suppose then that there is a compatibility edge between a and f. Then there cannot be a compatibility edge between b and f, otherwise the pattern Z would occur. Whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_4. If there is no compatibility edge between a and f, then whatever we decide for the compatibility of the remaining edges, the resulting pattern will be a subpattern of U_5, and we have the Lemma. □

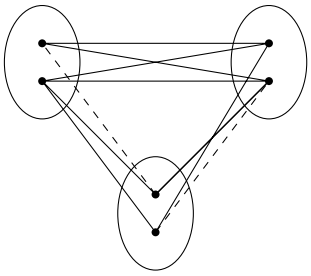
This result shows that the only tractable non-mergeable patterns are the ones which are reducible to a subpattern of a pattern in $\mathcal{U}$. By Lemma 2, we only need to study the subpatterns of the patterns in $\mathcal{U}$. There are 5 patterns in $\mathcal{U}$, four of which having 10 edges, the other one having 11 edges. So the number of patterns to study is bounded by $4 \times 2^{10} + 2^{11} = 6144$. However, most of the possible cases can be discarded by reduction or symmetry arguments. The total number of distinct open patterns on three variables is actually about 100. We give an exhaustive list in the next subsection.

4.3.2 List of All Possible Tractable Patterns on Three Variables

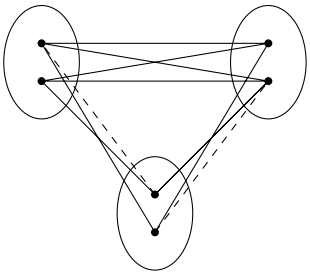
All cases where one of the three constraints does not contain any edge are covered by Theorem 1. We now give the exhaustive list of all 100 non-mergeable patterns on three variables, and with at least one edge in each of the three constraints, which are tractable or whose complexity is still open.



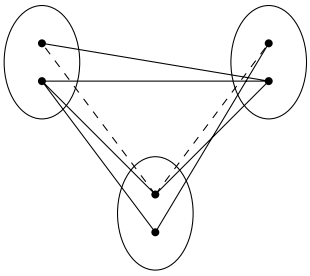
U_1 U_2 U_3



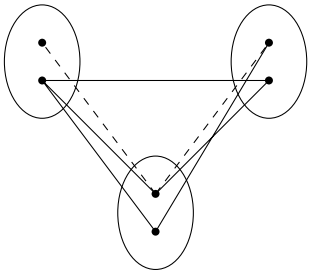
U_4



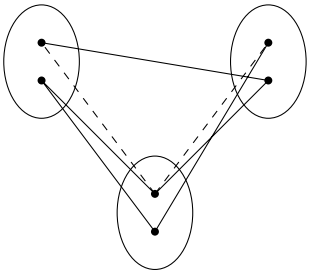
U_5



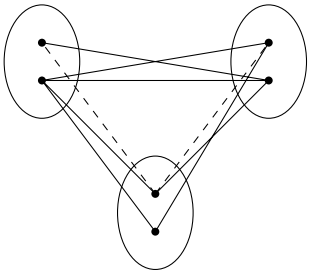
U_6



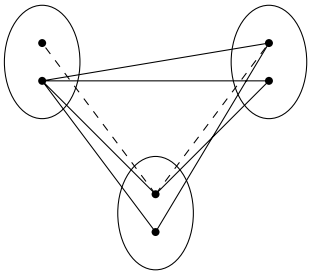
U_7



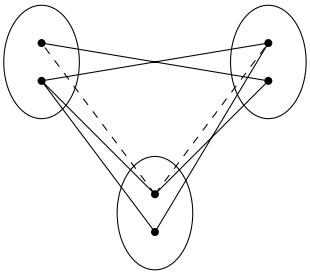
U_8



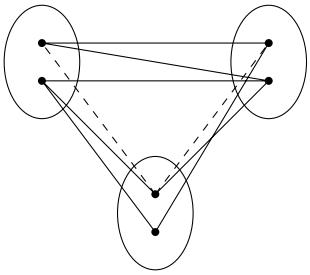
U_9



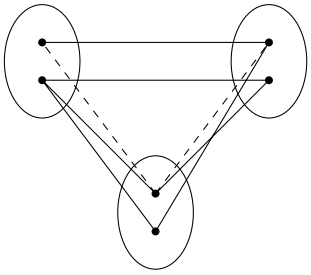
U_{10}



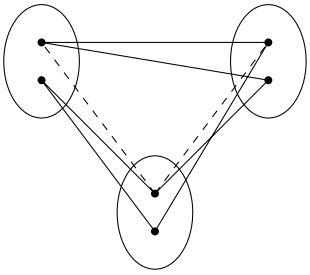
U_{11}



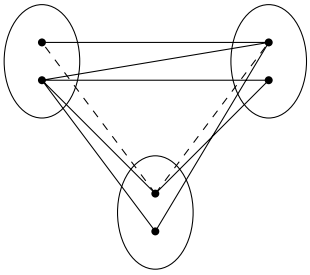
U_{12}



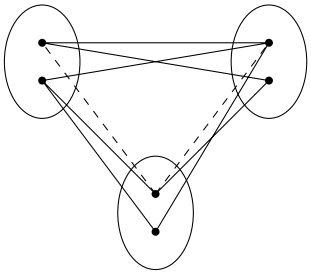
U_{13}



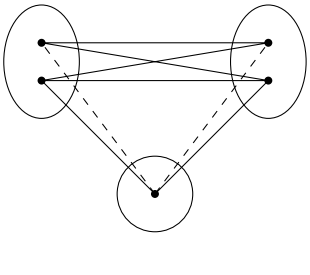
U_{14}



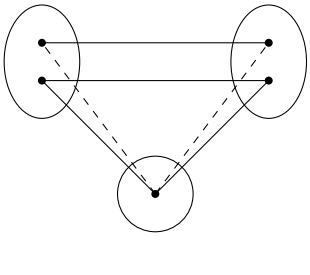
U_{15}



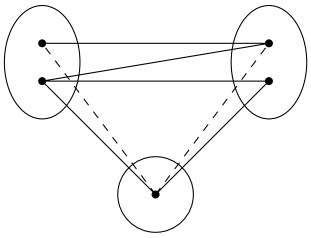
U_{16}



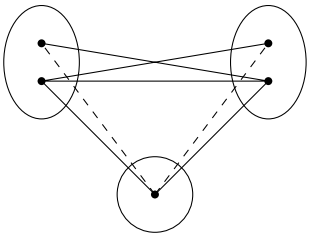
U_{17}



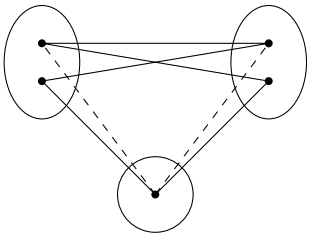
U_{18}



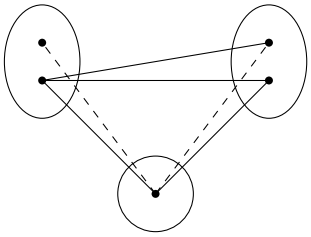
U_{19}



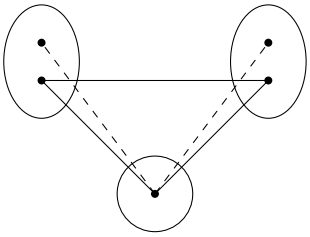
U_{20}



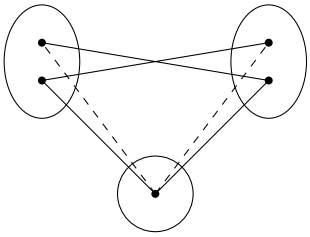
U_{21}



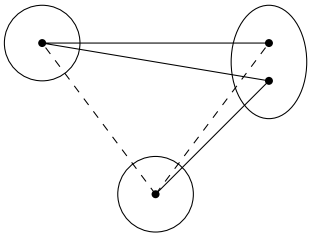
U_{22}



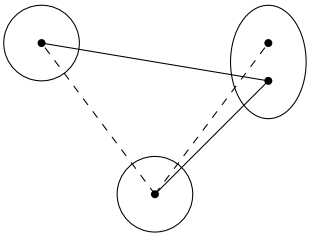
U_{23}



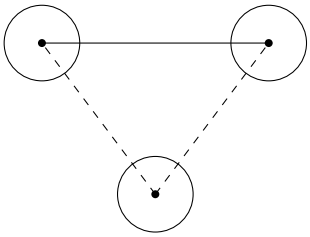
U_{24}



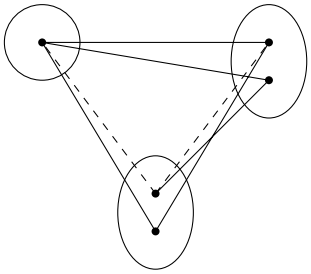
U_{25}



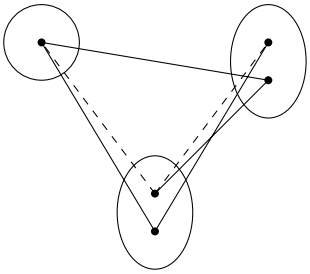
U_{26}



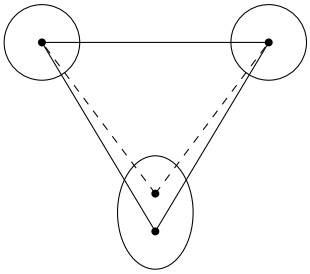
U_{27}



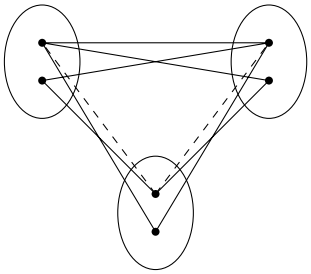
U_{28}



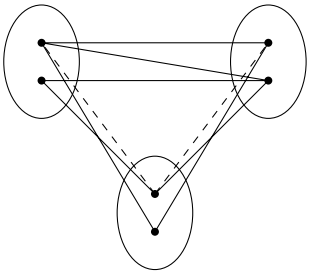
U_{29}



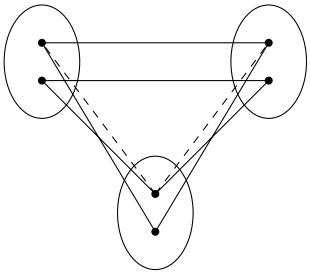
U_{30}



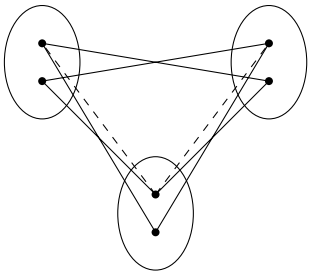
U_{31}



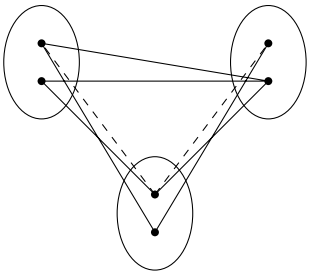
U_{32}



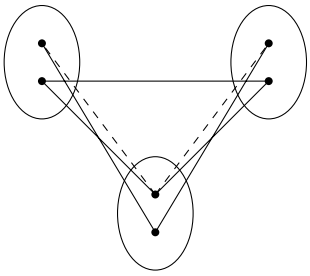
U_{33}



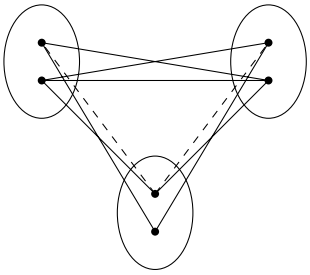
U_{34}



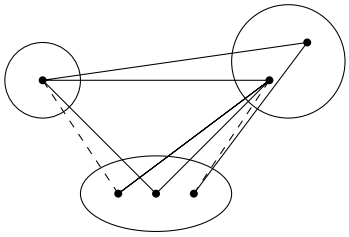
U_{35}



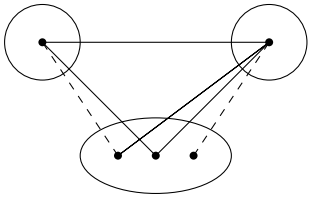
U_{36}



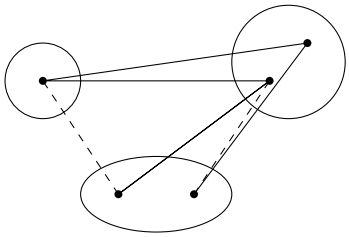
U_{37}



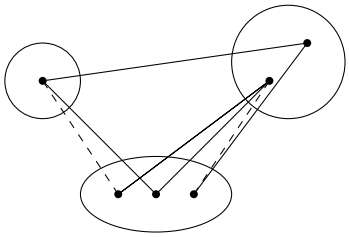
U_{38}



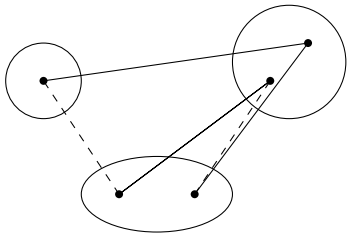
U_{39}



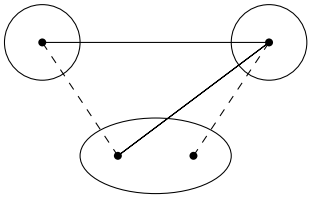
U_{40}



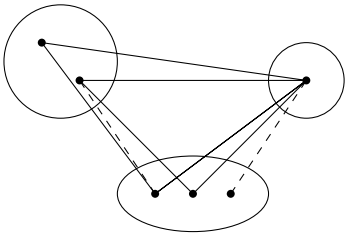
U_{41}



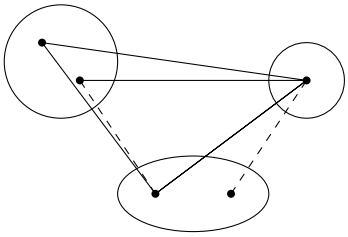
U_{42}



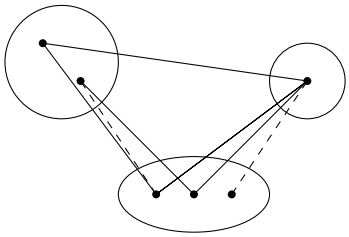
U_{43}



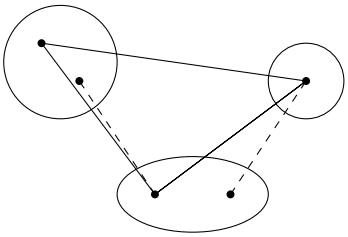
U_{44}



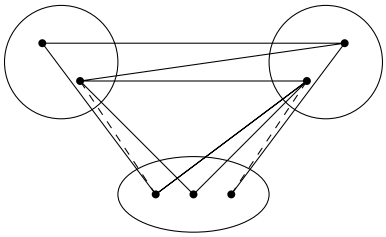
U_{45}



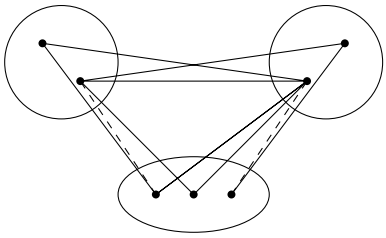
U_{46}



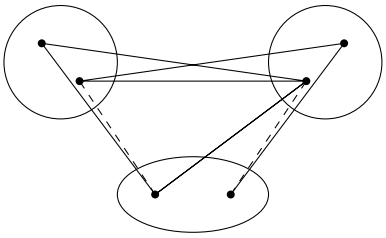
U_{47}



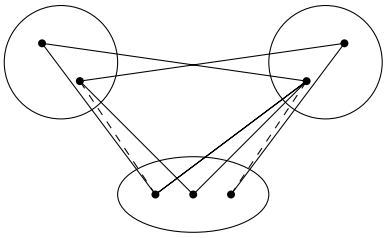
U_{48}



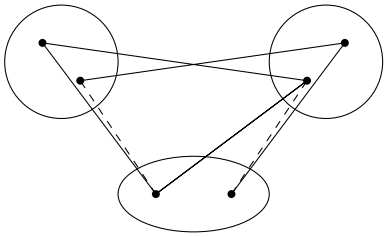
U_{49}



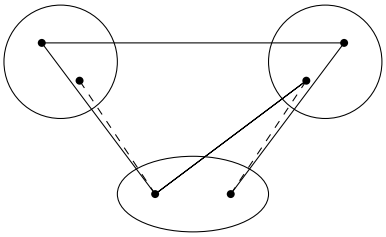
U_{50}



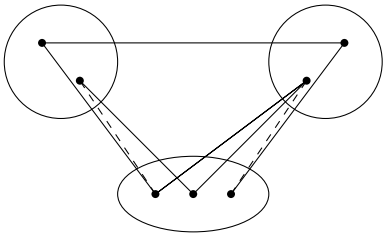
U_{51}



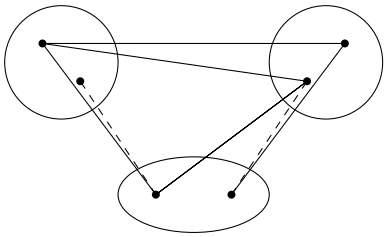
U_{52}



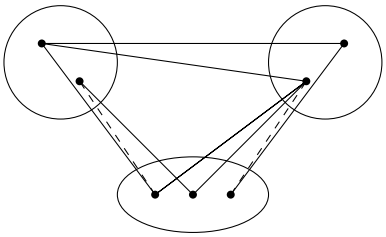
U_{53}



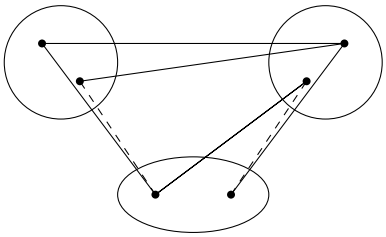
U_{54}



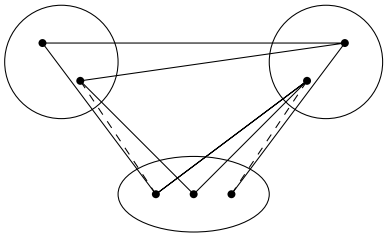
U_{55}



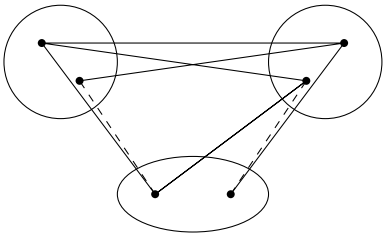
U_{56}



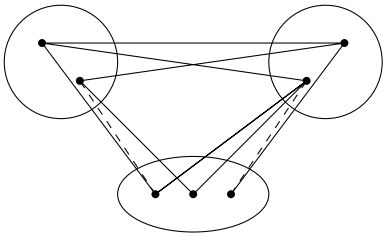
U_{57}



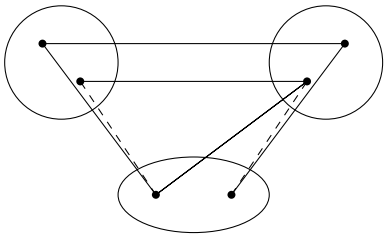
U_{58}



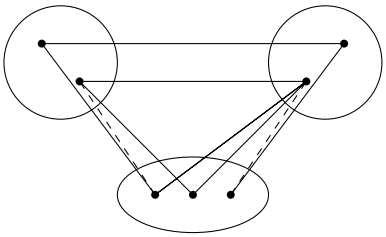
U_{59}



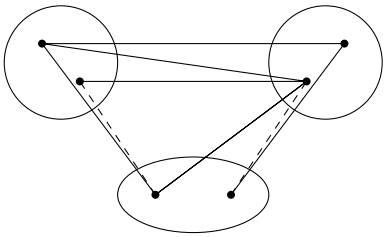
U_{60}



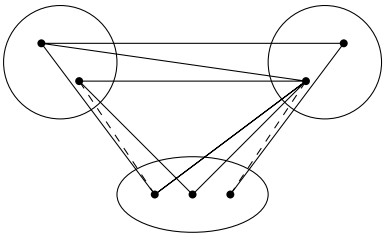
U_{61}



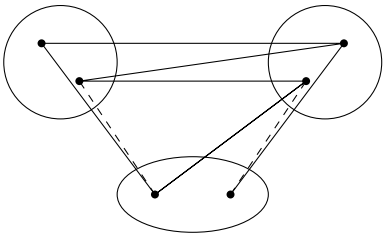
U_{62}



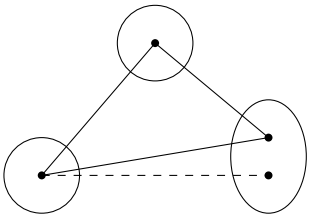
U_{63}



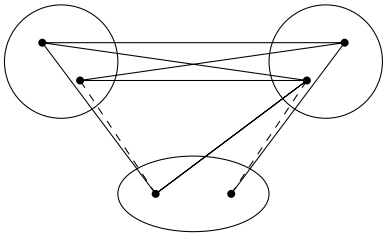
U_{64}



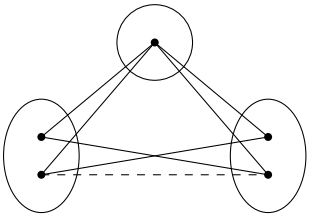
U_{65}



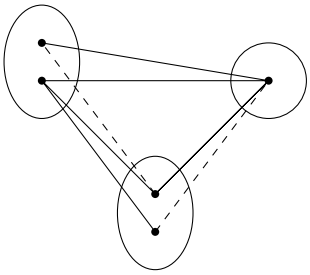
U_{66}



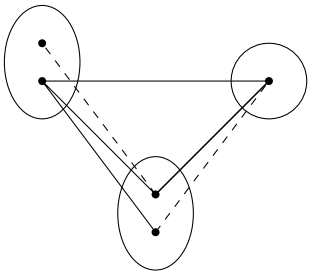
U_{67}



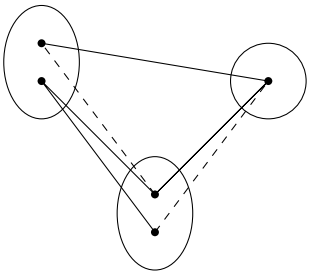
U_{68}



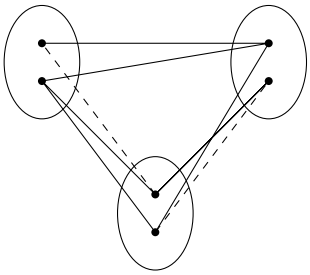
U_{69}



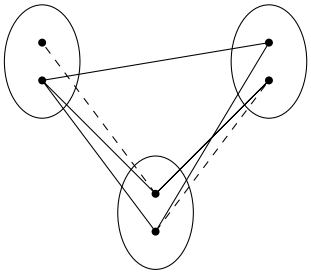
U_{70}



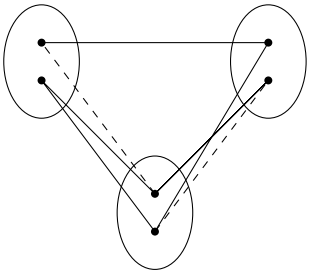
U_{71}



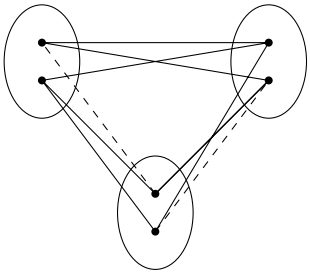
U_{72}



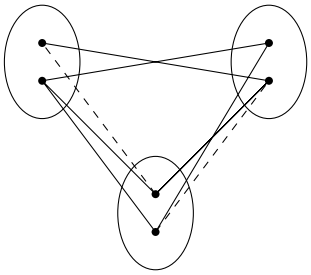
U_{73}



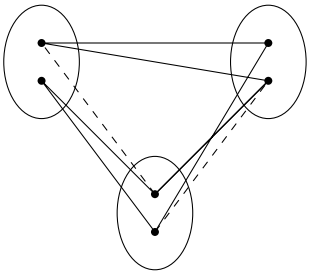
U_{74}



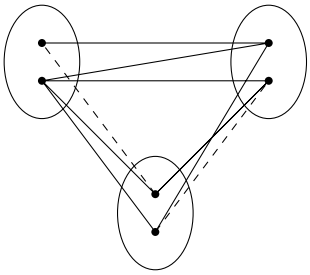
U_{75}



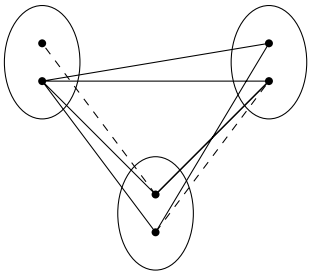
U_{76}



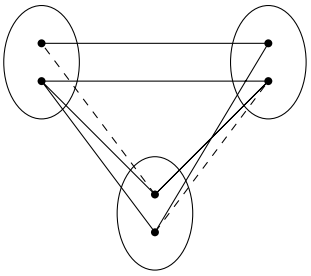
U_{77}



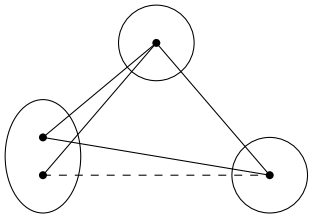
U_{78}



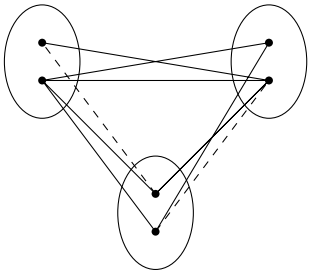
U_{79}



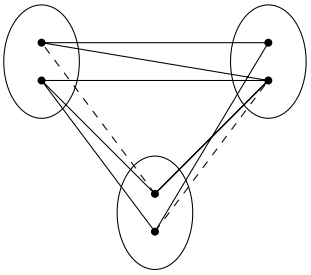
U_{80}



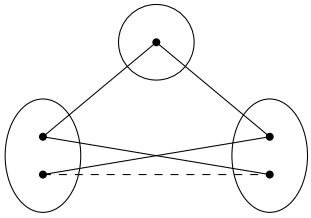
U_{81}



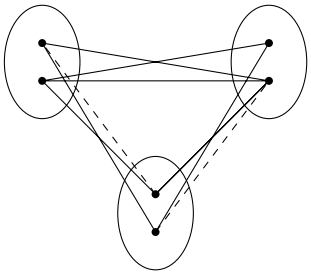
U_{82}



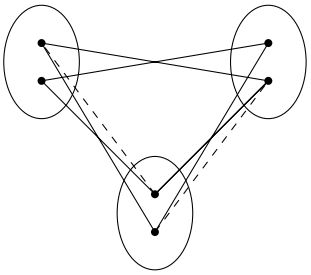
U_{83}



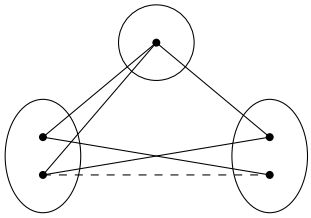
U_{84}



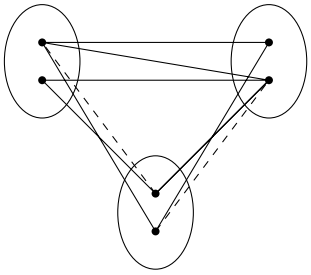
U_{85}



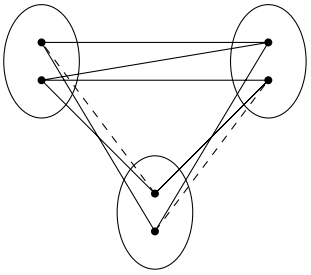
U_{86}



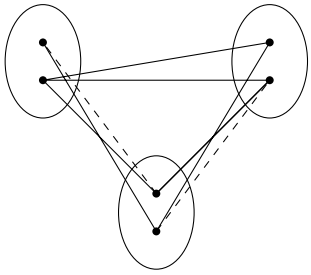
U_{87}



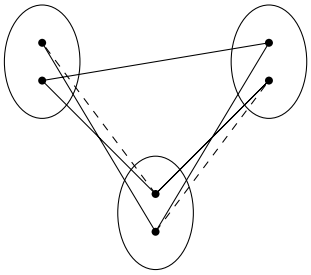
U_{88}



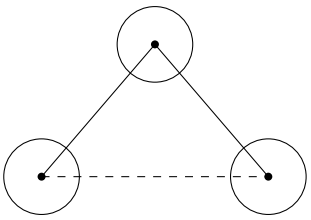
U_{89}



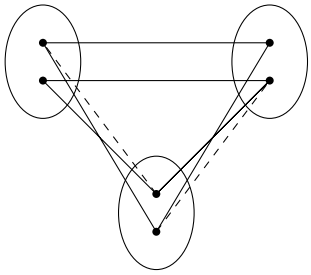
U_{90}



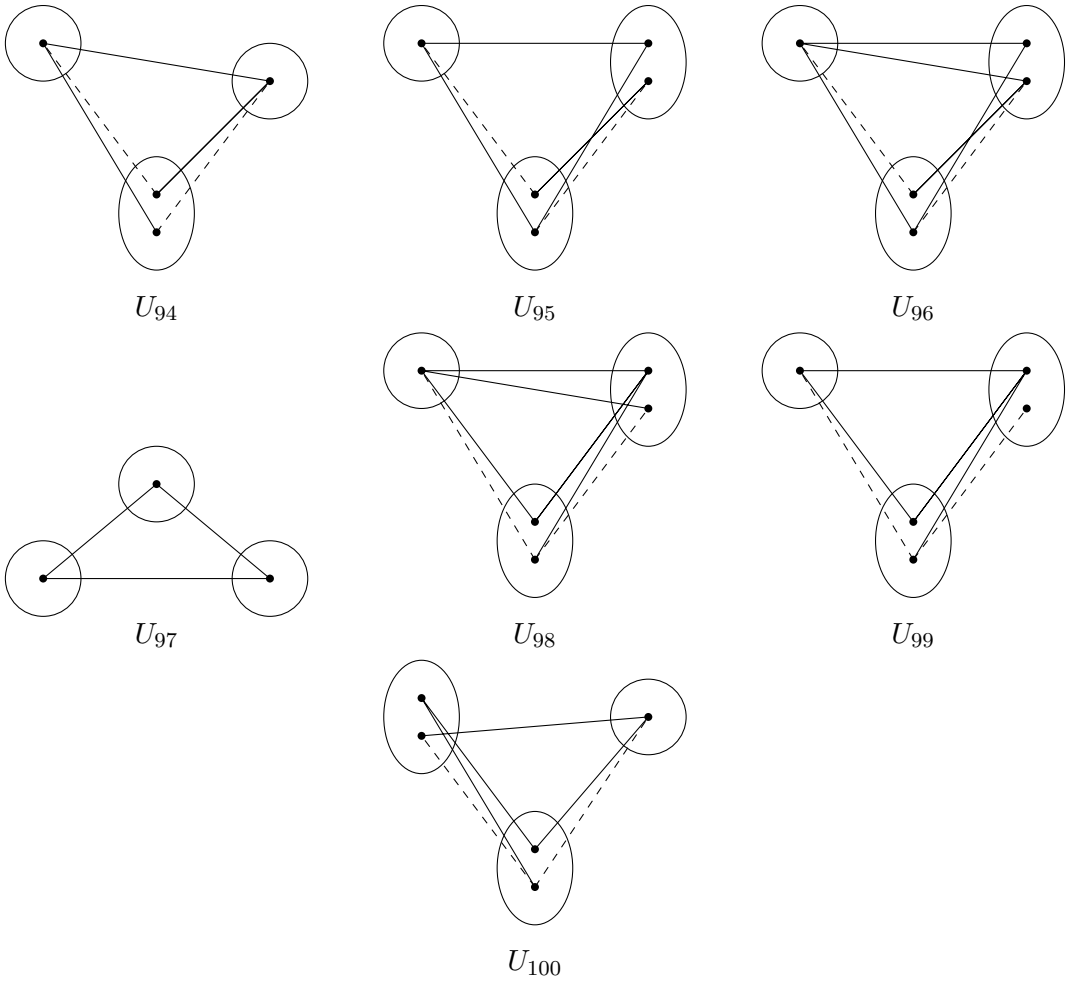
U_{91}



U_{92}



U_{93}



4.3.3 Tractability Proofs

In this section, we are going to prove the tractability of several patterns on three variables, and with at least one edge in each of the three constraints. Some of these patterns are non-mergeable, and thus are listed in Section 4.3.2. We will also show the tractability of two mergeable patterns, U'_{30} and U'_{25}. These two patterns can be reduced by merging to U_{30} and U_{25} respectively. They cannot be reduced to any known tractable pattern, nor can their complexity be inferred from sources other than the proofs that we are about to give. Therefore their tractability is a new and relatively interesting result, even though we have not been able yet to place it within some greater characterization. We first define some basic concepts which are needed for the following proofs.

Definition 42 (weakly incompatible). *Let $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ be a CSP instance, let v and v' be two variables in V_I, and let a be a point in A_v. We say that a and v are* weakly incompatible *with v' if $v = v'$ or if there exists a point $b \in A_{v'}$ such that a and b are incompatible.*

Definition 43 (point connexity set). *Let $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ be a CSP instance, and let a be a point in A_I. We call* point connexity set *of a the set of points $b \in A_I$ such that there exists a path of*

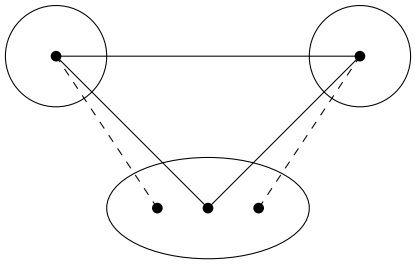


Figure 45: The Pattern U'_{30}

incompability edges between a and b.

Definition 44 (variable connexity set). *Let $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ be a CSP instance, and let a be a point in A_I. We call* variable connexity set of a *the set of variables $v \in V_I$, such that there exists $b \in A_v$, with b belonging to the point connexity set of a.*

Let *CompatibleWithAll* be the following preprocessing operation: if $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ is a CSP instance and there is a point $a \in A_v$, with $v \in V_I$, such that a is compatible with all points of $A_I \backslash A_v$, then we remove the variable v from V_I and only consider the remaining variables of the instance while looking for a solution. It is easy to see that *CompatibleWithAll* is polynomial and does not change the solvability of an instance.

We now give several tractability proofs. The first two patterns studied are mergeable patterns, while the later ones are non-mergeable.

Lemma 46. *The pattern U'_{30}, pictured in Figure 45, is tractable.*

Proof. We suppose that we have a CSP instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ in which the pattern U_{30} does not appear. We apply *CompatibleWithAll* on I. The resulting instance is I'. Suppose that we have a solution S for $I' = \langle V_{I'}, A_{I'}, var_{I'}, E_{I'}, cpt_{I'} \rangle$. Suppose that there are two points $a \neq b$ in S both weakly incompatible with a same variable v. Then by considering the point $c \in A_v$ belonging to S we have the forbidden pattern. So $\forall v \in V_{I'}$, at most one point of S is weakly incompatible with v. But since we applied *CompatibleWithAll* on I, then all points of I', in particular all points of S, are weakly incompatible with at least one variable. Since there are exactly as many variables in $V_{I'}$ as points in S, each point of S is weakly incompatible with exactly one variable. So we can remove from $A_{I'}$ all points which are weakly incompatible with at least two distinct variables. So we have reduced I to a CSP instance not containing the pattern $V-$ consisting of two constraints, one incompability edge in each constraint, and the two incompability edges intersecting on a point in the central variable. The pattern $V-$ is tractable from Theorem 1. So we have reduced U'_{30} to a tractable pattern. Therefore U'_{30} is tractable. □

Lemma 47. *The pattern U'_{25}, pictured in Figure 46, is tractable.*

Proof. Let $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ be a CSP instance such that the pattern U'_{25} does not occur in I. We apply *CompatibleWithAll* on I. Let v_0 and v_1 be two variables in V_I such that there exist $a \in A_{v_0}$ and $b \in A_{v_1}$, with a being incompatible with b. Let $v_2 \in V_I$ be such that a is weakly

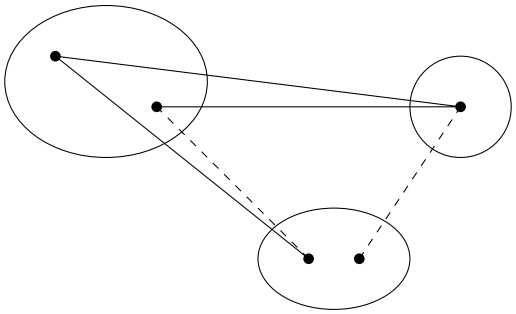


Figure 46: The pattern U'_{25}.

incompatible with v_2. Let $d \in A_{v_2}$ be a point such that a is incompatible with d. By arc consistency, we have $c \in A_{v_2}$ such that a and c are compatible. If b is compatible with both c and d, then we have the forbidden pattern. So b is incompatible with either c or d, and therefore b is weakly incompatible with v_2. So if two points are incompatible, then they are weakly incompatible with the same set of variables.

Let $a \in A_I$ and $v \in V_I$ be such that v is in the variable connexity set of a. Then there is a path of incompatibility edges of length k passing through the points $(a_0 = a, a_1, \ldots, a_k)$ with $a_k \in A_v$. So for all $0 \leq i \leq k-1$ we have a_i and a_{i+1} which are weakly incompatible with the same set of variables. So by the transitive property, all a_i are weakly incompatible with the same set of variables. So a is weakly incompatible with v. So each point of the instance is weakly incompatible with all variables in its variable connexity set.

Let $v_0 \in V_I$ and let a and b be two points in A_{v_0}. Since *CompatibleWithAll* has been applied to I, there is a variable $v_1 \neq v_0$ in V_a, the variable connexity set of a. Since $v_1 \in V_a$, there is a point $c \in v_1$ such that a and c are incompatible. Similarly, since *CompatibleWithAll* has been applied on I, there is a variable $v_2 \neq v_0$ in V_b, the variable connexity set of b. Let $d \in v_2$ be a point such that d is incompatible with b. Let V_d be the variable connexity set of d. If $v_2 = v_1$, then $v_1 \in V_d = V_b$. Otherwise, by arc consistency we have $e \in A_{v_1}$ such that a and e are compatible. The current situation is represented in Figure 47. If d is compatible with both c and e, then we have the forbidden pattern. So d is incompatible with either c or d. So $v_1 \in V_d = V_b$. So if a and b are two points belonging to the same variable, then $\forall v, v \in V_a \Rightarrow v \in V_b$. So if two points a and b are in the same variable, then $V_a = V_b$. So variable connexity sets are not connected to any other variable in the constraint graph. So we can solve variable connexity sets independently.

Suppose that we have a solution S for a given variable connexity set. Let $a \in A_{v_1}$, with $v_1 \in V_I$, be a point of this set such that a is compatible with a point $b \in v_2$ in S, with $v_2 \in V$. If $a \in S$, then a is compatible with all points of S. Otherwise, let a' be the point of S in A_{v_1}. Let $c \in A_{v_3}$, with $v_3 \in V_I$, be a point of S such that v_3 is distinct from v_1 and v_2. Since $c \in S$, c is compatible with a'. Moreover, since we are in a variable connexity set, then b is weakly incompatible with v_3. The current situation is represented in Figure 48. If a is incompatible with c, then we have the forbidden pattern. So a is compatible with c. So a is compatible with all points of S. So a is in a solution S' obtained from S after replacing a' by a. So if a point a is compatible with a point of S, then it is compatible with all points of S, and is itself in a solution S'.

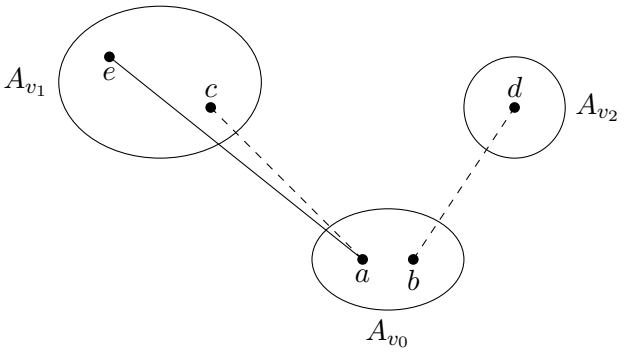


Figure 47: A situation.

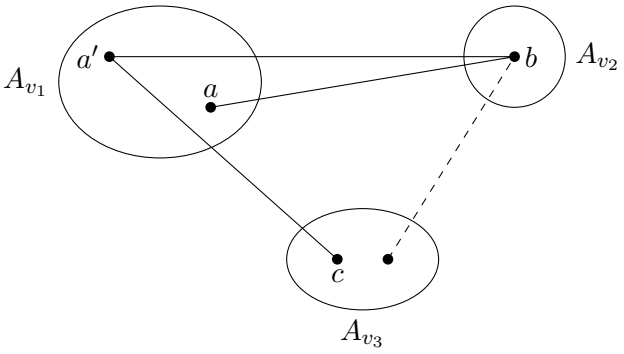


Figure 48: Another situation.

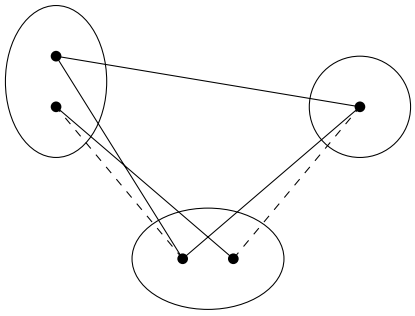


Figure 49: The pattern U_{95}.

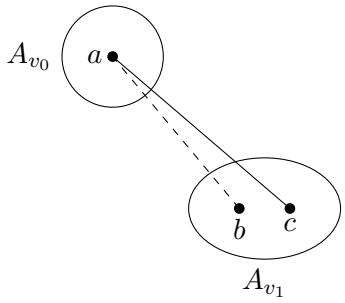


Figure 50: The gadget G.

Let v_0 be a variable in the variable connexity set currently being considered. Suppose that we have a point a in A_{v_0} such that a is in a solution. Let b and c be two points compatible with a such that b and c are assignments to two different variables. So b is in a solution S_1 containing a and c is in a solution S_2 also containing a. Since b is compatible with a point of S_2 (namely a), then b is compatible with all points of S_2. In particular, b is compatible with c. So all points compatible with a are compatible with each other.

So in order to solve the variable connexity set, we take a variable v_0 and a point $a \in A_{v_0}$. By arc consistency, we can take a point compatible with a in each variable of the variable connexity set. If the resulting set is not a solution, then we remove a and we try again with another point in A_{v_0}. If no point of A_{v_0} leads to a solution, then there is no solution for this variable connexity set.

So we have shown that U'_{25} is polynomial. □

Lemma 48. *The pattern U_{95}, pictured in Figure 49, is tractable.*

Proof. Consider an instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ from CSP($\overline{U_{95}}$). Let G be the gadget shown in Figure 50: two variables v_0 and v_1 such that we have $a \in A_{v_0}$, $b, c \in A_{v_1}$, a incompatible with b and a compatible with c. Suppose that we have G in the instance.

Suppose that b is in a solution S. Let d be the the point of S in A_{v_0}. Let $v_2 \in V_I$ be a variable such that $v_2 \neq v_0$ and $v_2 \neq v_1$, and let e be the point of S in A_{v_2}. The current situation is pictured

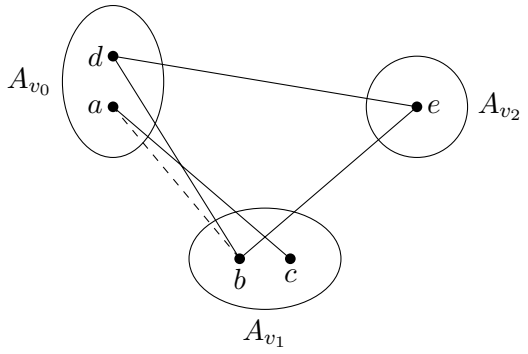


Figure 51: A situation.

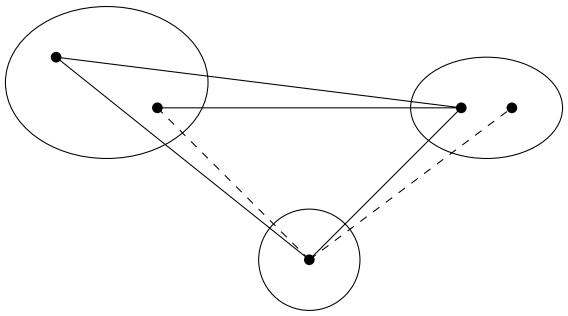


Figure 52: The pattern U_{22}.

in Figure 51. If c and e are incompatible, then we have the forbidden pattern. So c and e are compatible. So c is compatible with all points of S which are neither in A_{v_0} nor in A_{v_1}.

If c and d are compatible, then c is compatible with all points of S which are not in its variable. So c belongs to a solution S' obtained from S after replacing b by c. If c and d are incompatible, then the edges (c, a) and (c, d) form the gadget G. Since $d \in S$, then a is compatible with all points of S which are neither in v_0 nor in v_1. Since a and c are compatible, then a and c belong to a solution S' obtained from S after replacing b by c and d by a.

So if b is in a solution S, then c is also in a solution. So we can remove b. So each time we have the gadget G, we can remove a point. From Theorem 1, we know that the gadget G is a tractable pattern. Therefore, U_{95} is tractable. □

Lemma 49. *The pattern U_{22}, pictured in Figure 52, is tractable.*

Proof. Consider an instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ from CSP($\overline{U_{22}}$). Let v_0 and v_1 be two variables in V_I such that there exist $a \in A_{v_0}$ and $b \in A_{v_1}$, with a being incompatible with b. Let $v_2 \in V_I$ such that a is weakly incompatible with v_2. If there are no two points $c \in A_{v_1}$ and $d \in A_{v_2}$ such that a, c and d are compatible with each other, then a cannot belong to a solution and we can remove a. So we can assume that we have two points $c \in A_{v_1}$ and $d \in A_{v_2}$ such that a, c and d are compatible

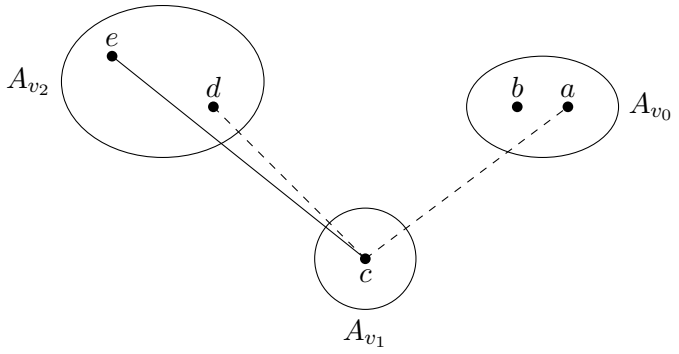


Figure 53: A situation.

with each other. If b is compatible with d, then we have the forbidden pattern. So b is incompatible with d and weakly incompatible with v_2. So if two points are incompatible with each other, then they are weakly incompatible with the same set of variables.

Let $a \in A_I$ and $v \in V_I$ be such that v is in the variable connexity set of a. Then there is a path of incompatibility edges of length k passing through the points $(a_0 = a, a_1, \ldots, a_k)$ with $a_k \in A_v$. So for all $0 \leq i \leq k-1$ we have a_i and a_{i+1} which are weakly incompatible with the same set of variables. So by the transitive property, all a_i are weakly incompatible with the same set of variables. So a is weakly incompatible with v. So each point of the instance is weakly incompatible with all variables in its variable connexity set.

We are now going to show that if a variable $v \in V_I$ belongs to two variable connexity sets of size at least 3, then these two variable connexity sets are actually the same variable connexity set (0). Suppose that we have a and b in A_{v_0}, with $v_0 \in V_I$. Let V_a be the variable connexity set of a and let V_b be the variable connexity set of b. We have to show that if V_a and V_b are of size at least 3, then $V_a = V_b$. Suppose that V_a and V_b are of size at least 3.

We are first going to show that there is at most one variable v_1 such that $v_1 \in V_a$ and $v_1 \notin V_b$. Suppose that we have two different variables v_1 and v_2 in V_a. Since $v_1 \in V_a$, there is a point $c \in A_{v_1}$ such that a and c are incompatible. Let V_c be the variable connexity set of c. Since c is incompatible with a, $V_c = V_a$. Since $v_2 \in V_a = V_c$, there is a point $d \in A_{v_2}$ such that a and d are incompatible. By arc consistency, we also have a point $e \in A_{v_2}$ such that c and e are compatible. The current situation is represented in Figure 53. If b is compatible with c, d and e then we have the forbidden pattern. So b is compatible with c, d or e. So either v_1 or v_2 is in V_b. So there is at most one variable v_1 such that $v_1 \in V_a$ and $v_1 \notin V_b$ (1). By a symmetry argument, there is also at most one variable v_2 such that $v_2 \in V_b$ and $v_2 \notin V_a$ (2).

From (1) and (2), and since V_a and V_b are both of size at least 3, we know that there is another variable $v_0' \in V_a \cap V_b$ such that $v_0' \neq v_0$. Let $v_2 \in V_b$ such that $v_2 \notin V_a$. Since $v_0' \in V_a \cap V_b$, there are two points c and d in $A_{v_0'}$ such that a is incompatible with c and b is incompatible with d. If b is incompatible with c, then a and b are in the same point connexity set and therefore we have (0). So we assume that b is compatible with c. Let V_c be the variable connexity set of c. We have $V_c = V_a$. Since $v_2 \in V_b$, there is a point $e \in A_{v_2}$ such that b and e are incompatible. By arc consistency, we

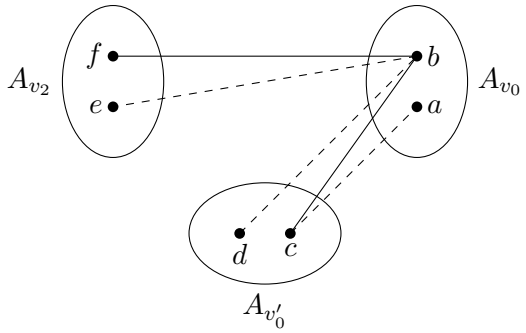


Figure 54: Another situation.

also have a point $f \in A_{v_2}$ such that b and f are compatible. The current situation is represented in Figure 54. If c is compatible with both e and f, then we have the forbidden pattern. So c is incompatible with either e or f. So c is weakly incompatible with v_2. So $v_2 \in V_c = V_a$. So there is no variable $v_2 \in V_I$ such that $v_2 \in V_b$ and $v_2 \notin V_a$ (3). By a symmetry argument, there is no variable $v_1 \in V_I$ such that $v_1 \in V_a$ and $v_1 \notin V_b$ (4).

From (3) and (4) we have $V_a = V_b$. So if a variable $v \in V_I$ belongs to two variable connexity sets of size at least 3, then these two variable connexity sets are actually the same variable connexity set. So variable connexity sets of size at least 3 are not connected to any other variable in the constraint graph. So we can solve variable connexity sets independently.

Suppose we have a solution S for a variable connexity set V of size at least 3. Let $a \in A_{v_0}$ and $b \in A_{v_1}$ be two points of S, with v_0 and v_1 in V_I. Let v_2 be a variable such that $v_2 \neq v_0$ and $v_2 \neq v_1$. Let c be the point of S in v_2. Since V is a variable connexity set, all points of v_0 are weakly incompatible with all variables of $V \backslash v_0$. In particular, a is weakly incompatible with both v_1 and v_2. Let $d \in A_{v_1}$ and $e \in A_{v_2}$ be two points such that a is incompatible with both d and e. The current situation is pictured in Figure 55. If b is compatible with e then we have the forbidden pattern. So b is incompatible with e. So a and b are in the same point connexity set. So all points of the solution S are in the same point connexity set. So we can solve point connexity sets independently.

Let S be a solution for a given point connexity set, and let $a \in A_{v_1}$, with $v_1 \in V_I$, be a point such that a is compatible with a point of S. If a is in S, then a is compatible with all points of S. Otherwise, let $b \in A_{v_2}$, with $v_2 \in V_I$, be the point of S compatible with a and let a' be the point of S in A_{v_1}. Let $c \in A_{v_3}$, with $v_3 \in V_I$, be a point of S such that $v_3 \neq v_1$ and $v_3 \neq v_2$. Since $c \in S$, c is compatible with a'. Moreover, since we are within a variable connexity set, c is weakly incompatible with v_2. The current situation is pictured in Figure 56. If a is incompatible with c, then we have the forbidden pattern. So a is compatible with c. So a is compatible with all points of S. So a is in a solution S' obtained from S after replacing a' by a. So if a point is compatible with a point in S, then it is compatible with all points in S, and is itself in a solution S'.

Let v_0 be a variable in the variable connexity set currently being considered. Suppose that we have a point a in A_{v_0} such that a is in a solution. Let b and c be two points compatible with a such

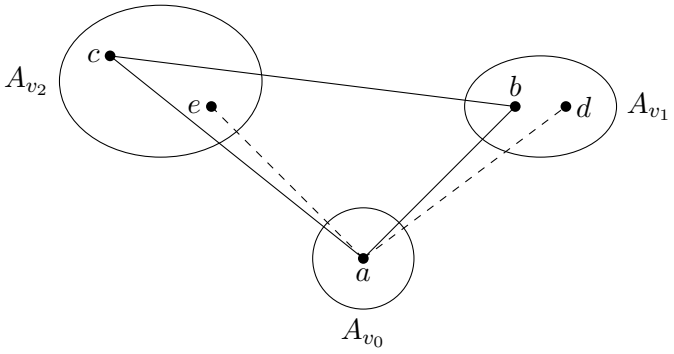


Figure 55: Yet another situation.

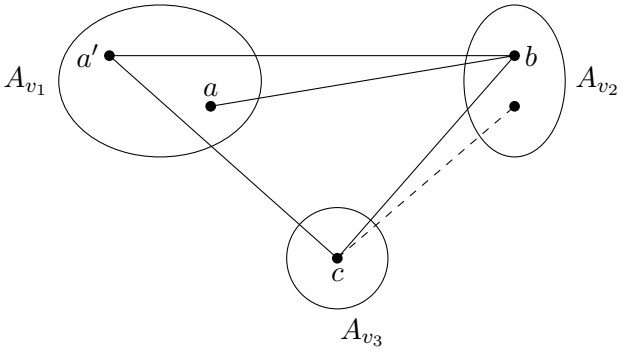


Figure 56: Yet again another situation.

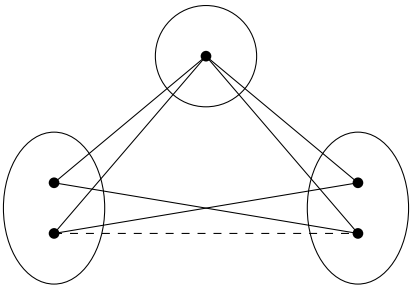


Figure 57: The pattern U_{68}.

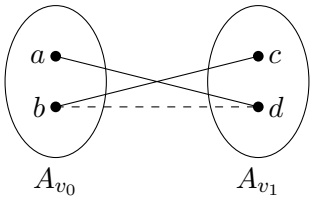


Figure 58: The gadget G.

that b and c belong to two different variables. So b is in a solution S_1 containing a and c is in a solution S_2 also containing a. Since b is compatible with a point of S_2 (namely a), b is compatible with all points of S_2. In particular, b is compatible with c. So all points compatible with a are compatible with each other.

So in order to solve the variable connexity set, we take a variable v_0 and a point $a \in A_{v_0}$. By arc consistency, we can take a point compatible with a in each variable of the variable connexity set. If the resulting set is not a solution, then we remove a and we try again with another point in A_{v_0}. If no point of A_{v_0} leads to a solution, then there is no solution for this variable connexity set.

Once we have a solution for all variable connexity sets of size at least 3, we know that the rest of the instance does not contain any point weakly incompatible with two different variables. So we have reduced I to a CSP instance not containing the pattern $V-$ consisting of two constraints, one incompability edge in each constraint, and the two incompability edges intersecting on a point in the central variable. The pattern $V-$ is tractable from Theorem 1. So we have reduced U_{22} to a tractable pattern. Therefore U_{22} is tractable. □

Lemma 50. *The pattern U_{68}, pictured in Figure 57, is polynomial.*

Proof. Consider an instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ from CSP($\overline{U_{68}}$). Let v be a variable in V_I. Let d be the number of points in $A_v = \{a_1, \ldots, a_d\}$. $\forall 1 \leq j \leq d$, let I_j be the instance obtained from I after removing v and all points $b \in A_I$ such that $cpt_I(a_j, b) = F$. Let G be the gadget shown in Figure 58: two variables v_0 and v_1 such that we have $a, b \in A_{v_0}$, $c, d \in A_{v_1}$, c incompatible with d, a compatible with d and b compatible with c.

Suppose that for some $1 \leq j \leq d$ we have G occuring in I_j. Since a_j is compatible in I with all points of I_j, including a, b, c and d, the pattern U_{68} occurs in I. So $\forall j \in [1, d]$, the gadget G does not

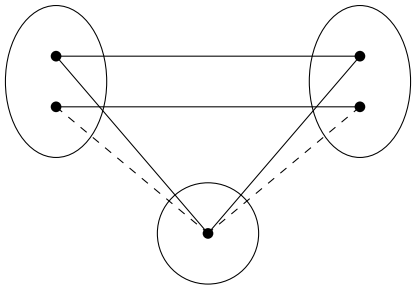


Figure 59: The pattern U_{18}.

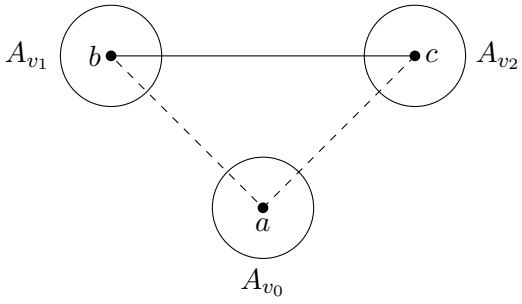


Figure 60: The gadget G.

occur in I_j. So we have reduced I to d instances from CSP($\overline{G}$). The gadget G is reducible by dp-elimination to a pattern composed of a single incompatibility edge. This latter pattern is tractable from Theorem 1. So we have reduced U_{68} to a tractable pattern. Therefore U_{68} is tractable. □

Lemma 51. *The pattern U_{18}, pictured in Figure 59, is polynomial.*

Proof. Consider an instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ from CSP($\overline{U_{18}}$). Let G be the gadget shown in Figure 60: three variables v_0, v_1 and v_2 such that we have $a \in A_{v_0}$, $b \in A_{v_1}$, $c \in A_{v_2}$, a incompatible with both b and c and b compatible with c. Suppose that we have G in the instance.

Suppose that a is in a solution S. Let d be the point of S in A_{v_1} and let e be the point of S in A_{v_2}. Since S is a solution for I, a, d and e are compatible with each other. So we have the forbidden pattern. So if a is in a solution, then we have the forbidden pattern. So a is not in any solution. So we can remove a. So each time we have the gadget G, we can remove a point. The gadget G is actually the pattern NEGTRANS, which is tractable (Cooper & Živný, 2012). So the gadget G is a tractable pattern. Therefore, U_{18} is tractable. □

Lemma 52. *The pattern U_{36}, pictured in Figure 61, is polynomial.*

Proof. Consider an instance $I = \langle V_I, A_I, var_I, E_I, cpt_I \rangle$ from CSP($\overline{U_{36}}$). Let G be the gadget shown in Figure 62: three variables v_0, v_1 and v_2 such that we have $a, b \in A_{v_0}$, $c \in A_{v_1}$, $d \in A_{v_2}$, a incompatible with both c and d and b compatible with both c and d. Suppose that we have G in the instance.

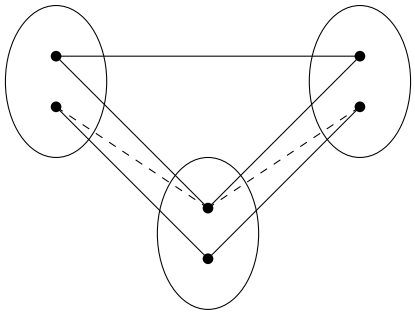


Figure 61: The pattern U_{36}.

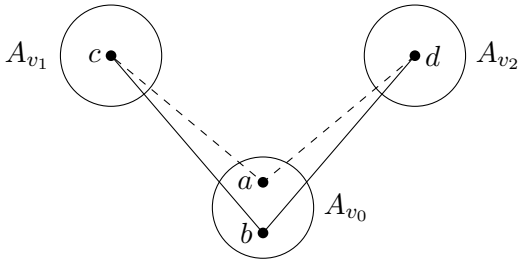


Figure 62: The gadget G.

Suppose that a is in a solution S. Let e be the point of S in A_{v_1} and let f be the point of S in A_{v_2}. Since S is a solution for I, a, e and f are compatible with each other. So we have the forbidden pattern. So if a is in a solution, then we have the forbidden pattern. So a is not in any solution. So we can remove a. So each time we have the gadget G, we can remove a point. The gadget G is actually the pattern T_1, which is tractable from Theorem 1. So the gadget G is a tractable pattern. Therefore, U_{36} is tractable. □

Remarks: Since U_{42} and U_{47} are subpatterns of U_{95}, Lemma 48 implies that U_{42} and U_{47} are tractable. Since U_{23} and U_{26} are subpatterns of U_{22}, Lemma 49 implies that U_{23} and U_{26} are tractable. Since U_{66}, U_{81}, U_{84} and U_{87} are subpatterns of U_{68}, Lemma 50 implies that U_{66}, U_{81}, U_{84} and U_{87} are tractable.

4.3.4 Summary of the Section

We now summarize the complexity results from Section 4.3.

Pattern	Tractable?	
What was already known:		Proof
U_{27}	Yes	NEGTRANS (Cooper & Živný, 2012)
U_{94}	Yes	BTP (Cooper et al., 2010)
U_{43}	Yes	Subpattern of BTP.
U_{92}	Yes	(Cooper & Živný, 2012)
U_{97}	Yes	(Cooper & Živný, 2012)
New results in the thesis:		Method
U_{22}	Yes	Variable Connexity Sets.
U_{23}	Yes	Subpattern of U_{22}.
U_{26}	Yes	Subpattern of U_{22}.
U_{95}	Yes	Gadget + Removal of a point.[1]
U_{42}	Yes	Subpattern of U_{95}.
U_{47}	Yes	Subpattern of U_{95}.
U_{18}	Yes	Gadget + Removal of a point.
U_{36}	Yes	Gadget + Removal of a point.
U_{68}	Yes	Reduction to only weakly incompatibility variables.[2]
U_{66}	Yes	Subpattern of U_{68}.
U_{81}	Yes	Subpattern of U_{68}.
U_{84}	Yes	Subpattern of U_{68}.
U_{87}	Yes	Subpattern of U_{68}.
Non-mergeable patterns on three variables not subpatterns of one of the patterns in $\mathcal{U}$.	No	Number maximum of points + exhaustive search.
U'_{30}	Yes	Property shared by points in a solution.
U'_{25}	Yes	Variable Connexity Sets.
Open cases:		Method[3]
All 82 other patterns from Section 4.3.2.	?	Exhaustive search.

[1] "Gadget + Removal of a point" refers to the type of proof consisting of removing a point from an instance I if a given tractable gadget appears in I. We previously used this method in

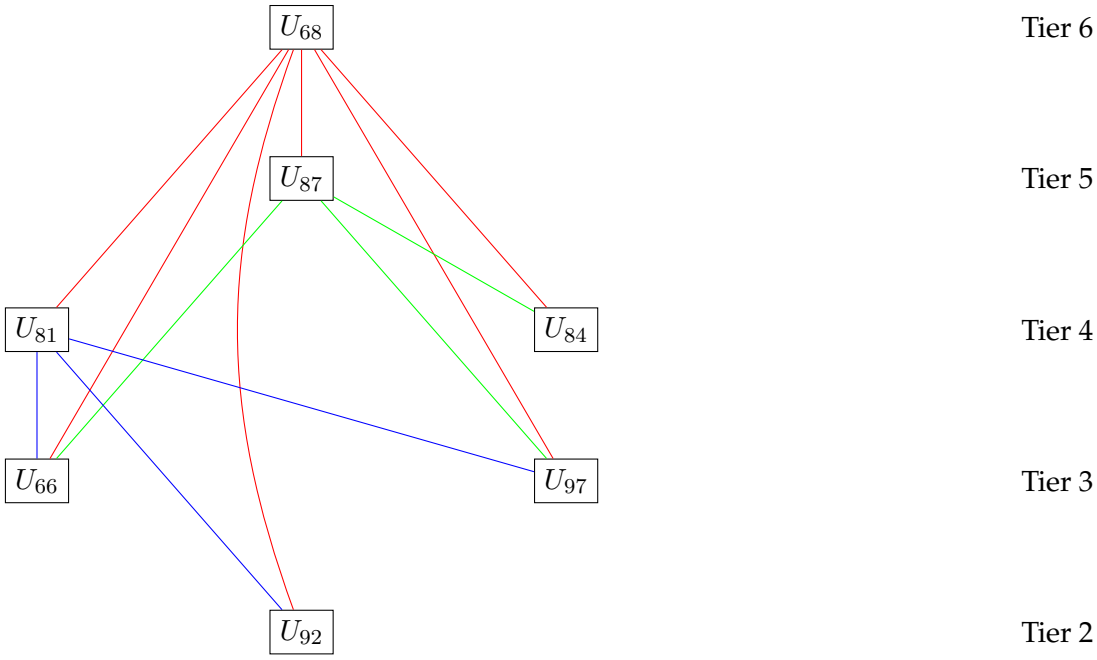


Figure 63: Graph of patterns including one or none incompatibility edges.

the tractability proofs for patterns T_3 and T_4 in Theorem 1. It was also used in the first part of the tractability proof for T_2, from the same Theorem.

[2]"Reduction to only weakly incompatibility variables" refers to the removal of a variable v from a pattern P when there is no incompatibility edge between v and any other variable from P.

[3]"Method" does not refer here to the proof of why the cases are open, but to the proof of why they are the only open cases remaining.

We give in Figures 63, 64 and 65 three graphs presenting the extension relations between all patterns from Section 4.3.2. In each graph, patterns are sorted by tiers, with each tier corresponding to the number of compatibility edges in the patterns. A line between two patterns in the graph represents a relation of extension, that is the pattern with the less compatibility edges is a subpattern of the pattern it is linked to. A rectangle around the label of a pattern denotes tractability.

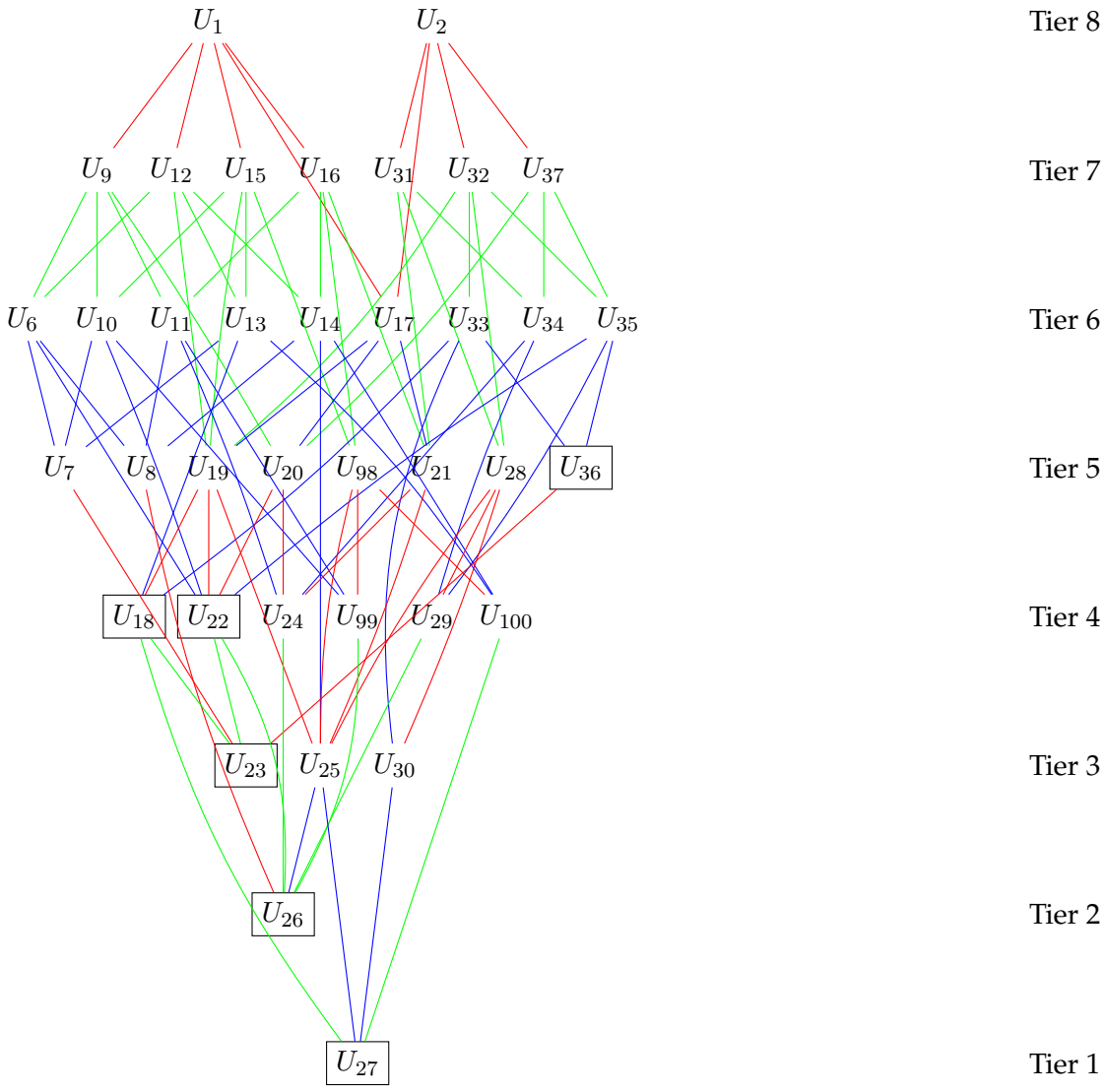


Figure 64: Graph of patterns including two intersecting incompatibility edges.

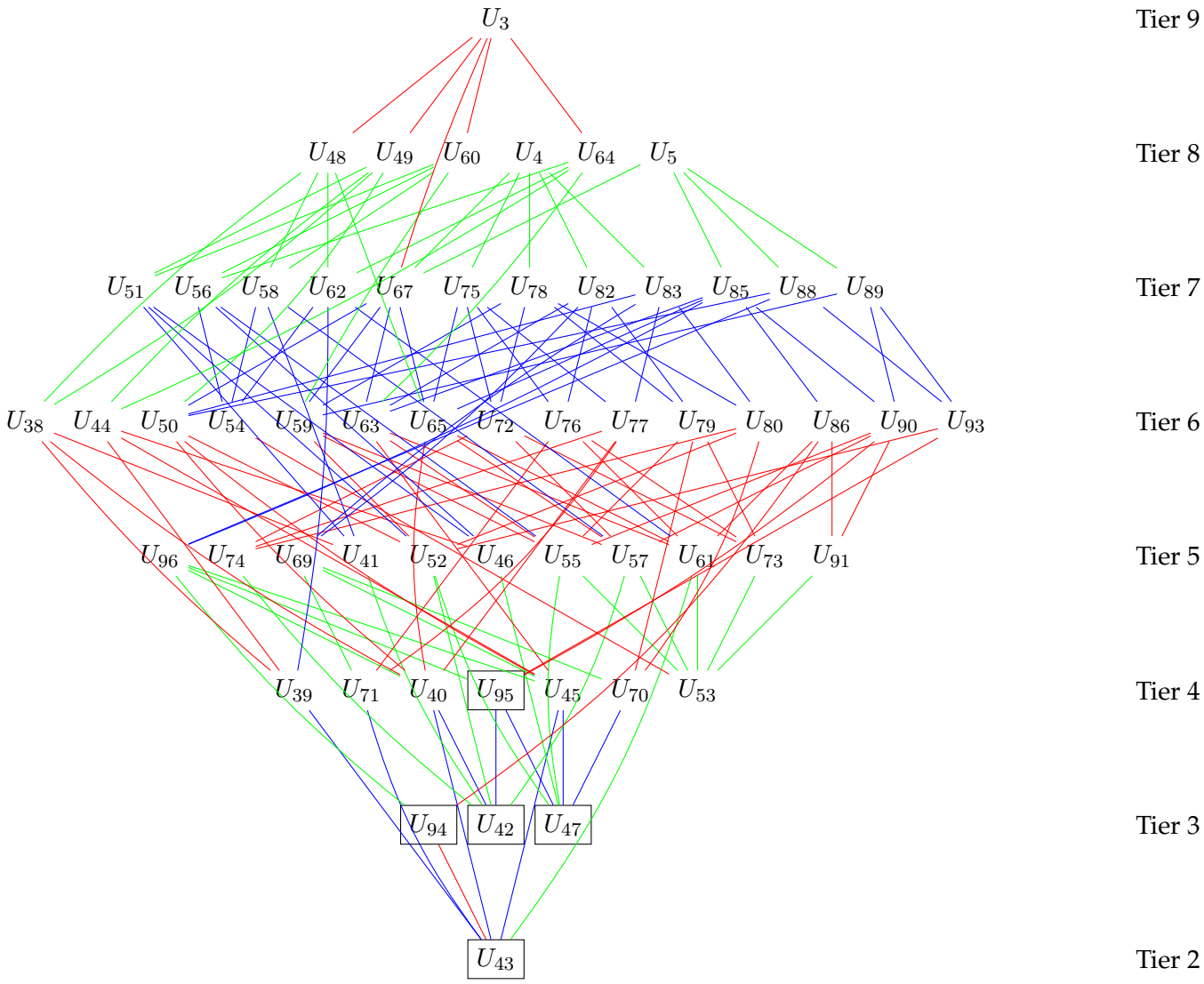


Figure 65: Graph of patterns including two non-intersecting incompatibility edges.

5 Simplification Operations

This chapter offers the rest of the results we found. It is divided in two parts. The first one uses one of the versions of forbidden patterns we defined, leading to the characterization of when they allow the elimination of a specific variable. The other one deals with other forms of instance simplification, like value elimination or subdomains fusion. It is not centered on forbidden patterns, but on other types of local conditions.

5.1 Variable Elimination

In this subsection we are concerned with variable elimination characterized by forbidden patterns. We now define what this means.

Definition 45 (variable eligible for elimination). *If $I = \langle V, A, var, E, cpt \rangle$ is a CSP instance, we say that a variable $x \in V$ can be* eliminated *in I 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.
The following Definition expands on Definition 21.

Definition 46 (arc consistent). *Let $I = \langle V, A, var, E, cpt \rangle$ be a CSP instance. A point $a \in A_v$, with $v \in V$, is called* arc consistent *if, for all variables $w \neq v$ there is some point b compatible with a.*

The CSP instance $I = \langle V, A, var, E, cpt \rangle$ 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 section we will assume that all CSP instances are arc-consistent.

In order to use patterns for variable elimination we need to define what we mean when we say that a flat pattern or an existential pattern occurs at variable x of a CSP instance.

Definition 47 (occurs at a variable, for an existential pattern). *Let $I = \langle V, A, var, E, cpt \rangle$ be a CSP instance. Let v be a variable in V. Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, e_P \rangle$ be an existential pattern. If $\forall S \subseteq A_v$ such that $|S| = |e_P|$, P occurs on S, then we say that P* occurs at v.

The concept of an existential pattern occuring on a set of points was previously defined in Definition 17.

Definition 48 (occurs at a variable, for a flat pattern). *Let $I = \langle V, A, var, E, cpt \rangle$ be a CSP instance. Let v be a variable in V. Let $s \in A_v$. Let $P = \langle V_P, A_P, var_P, E_P, cpt_P \rangle$ be a flat pattern. Let $a \in A_P$. If the existential pattern $P' = \langle V_P, A_P, var_P, E_P, cpt_P, a \rangle$ occurs on s, then we say that P* occurs at v.

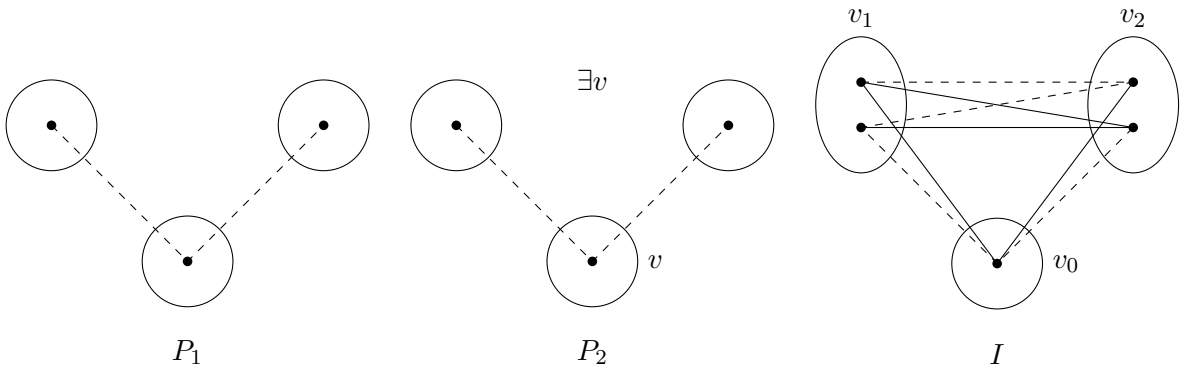


Figure 66: Example of a Quantified Pattern.

Definition 49 (occurs at a variable, for a quantified pattern). *Let $I = \langle V, A, var, E, cpt\rangle$ be a CSP instance. Let v be a variable in V. Let $s \in A_v$. Let $P = \langle V_P, A_P, var_P, E_P, cpt_P, v_P\rangle$ be a quantified pattern. Let $a \in A_{v_P}$. If the existential pattern $P' = \langle V_P, A_P, var_P, E_P, cpt_P, a\rangle$ occurs on s, then we say that P* occurs at *v.*

For example, in Figure 66, the flat pattern P_1 occurs at all three variables of the instance I. The quantified pattern P_2 occurs at variables v_0 and v_1, but does not occur at variable v_2.

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

Definition 50 (VE pattern). *A flat pattern is a* VE pattern *if, whenever the pattern does not occur at a variable x in an arc-consistent CSP instance, then x can be eliminated.*

Likewise, a quantified pattern is a VE pattern *if, whenever the pattern does not occur at a variable x in an arc-consistent CSP instance, then x can be eliminated.*

An existential pattern is a VE pattern *if, whenever the pattern does not occur at a variable x, for at least one value mapping, then x can be eliminated.*

Clearly existential patterns will allow more variables to be eliminated than quantified patterns. It follows from Definitions 17, 47, 48 and 49 that we only need to identify reduced VE patterns. We now give the characterization of all reduced VE patterns. There are precisely six, shown in Figure 67.

We begin by showing that each of these patterns allows variable elimination. We need a technical lemma which shortens several proofs.

Lemma 53. *If P is an existential VE pattern then its quantified simplified pattern is also a VE pattern. If P' is a quantified VE pattern then its flat simplified pattern is also a VE pattern.*

Proof. This is clear since every variable that can be eliminated by the quantified pattern could also be eliminated by the existential pattern. The same argument can be used for the second part of the Lemma. □

Proposition 3. *The patterns ∃subBTP, ∃invsubBTP, ∃snake(2), BTP, invsubBTP and snake(2) are VE patterns.*

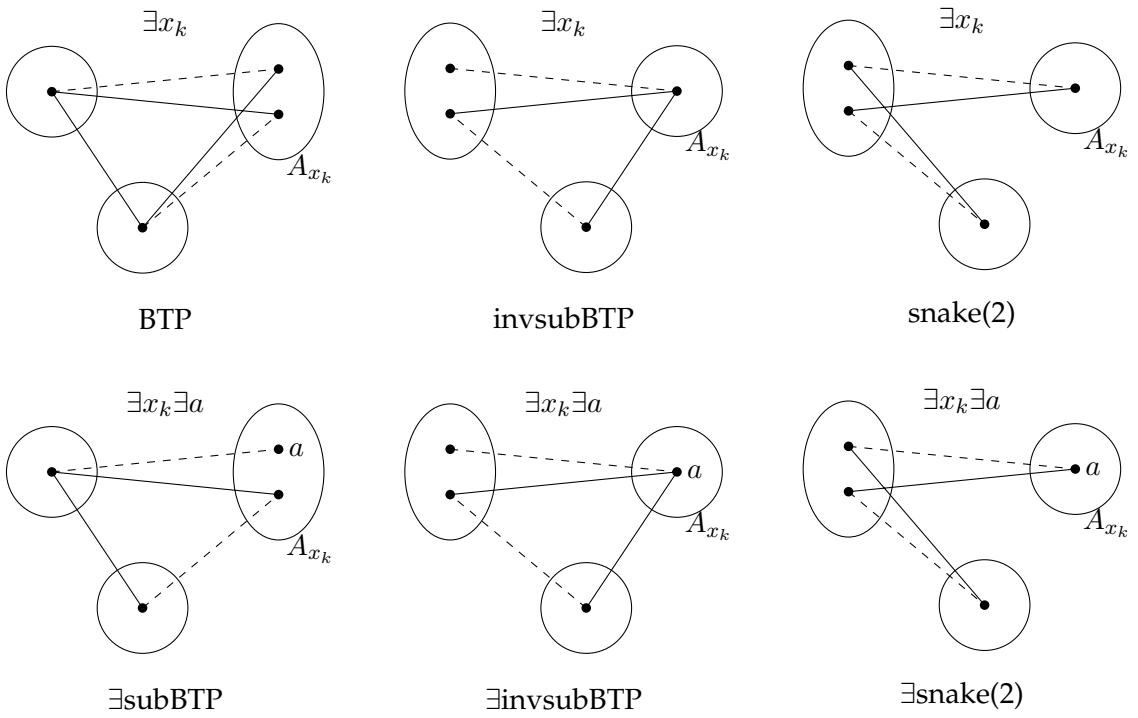


Figure 67: Variable elimination patterns.

Proof. Since it is known that BTP is a VE pattern (Cooper et al., 2010), by Lemma 53 we only need to prove the result for the three existential patterns: ∃subBTP, ∃invsubBTP and ∃snake(2).

Every two variable arc-consistent CSP instance allows either variable to be eliminated. So we only have to prove that the existential 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 $I = \langle X, A, var, E, cpt \rangle$ and let s be a partial solution on $X \setminus \{x\}$. For any $v \in X$, let $s(v)$ be the point of A_v in s.

Fix some assignment $d \in A_v$, and let:

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

If $y, z \in Y$, since s is a partial solution, we know $\mathrm{cpt}(s(y), s(z)) = T$.

What is more, by arc consistency, for all $z \in \overline{Y}$, there is some $t(z) \in A_z$ such that $\mathrm{cpt}(t(z), d) = T$.

We now prove the result for each pattern in turn.

Suppose that ∃subBTP does not occur at x with mapping $a \mapsto d$ in I.

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.

On the other hand, suppose that there is some $y \in \overline{Y}$. By arc consistency, $\exists b \in A_x$ such that $\mathrm{cpt}(s(y), b) = T$. Since the pattern ∃subBTP does not occur, we can deduce that, for every variable $z \in X$ different from both x and y, $\mathrm{cpt}(s(z), b) = T$. Hence, we can extend s to a solution to I by choosing $s(x) = b$. So, in any case x can be eliminated and ∃subBTP is indeed an existential VE pattern.

Now instead, suppose ∃invsubBTP does not occur at x with mapping $a \mapsto d$ in I.

Since the pattern ∃invsubBTP does not occur, if both y and z belongs to $\overline{Y}$ then $\mathrm{cpt}(t(y), t(z)) = T$.

Also, if $y \in Y$, $z \in \overline{Y}$, then $\mathrm{cpt}(s(y), t(z)) = T$.

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

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

So ∃invsubBTP is indeed an existential VE pattern.

For the third pattern, suppose that ∃snake(2) does not occur at x with mapping $a \mapsto d$ in I.

If $z \in \overline{Y}, y \in Y$, since the pattern ∃snake(2) does not occur, we can deduce that $\mathrm{cpt}(t(z), s(y)) = T$.

If both y and z belong to $\overline{Y}$, then we can deduce that $cpt(t(y), t(z)) = T$.

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

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

So ∃snake(2) is also an existential VE pattern. □

As a side note, ∃snake(2) is the same pattern as X_2 from Figure 37.

Our aim is to precisely characterize all reduced patterns which allow variable elimination in an arc-consistent binary CSP instance. We begin by identifying many patterns, shown in Figure 68, which are not variable elimination patterns.

Lemma 54. *None of the following patterns allow variable elimination in arc-consistent binary CSP instances: any pattern on four variables, any pattern with three distinct values for the same variable, Triangle, V, XL, 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 (existential) pattern P at a variable x (for some mapping);
- has a partial solution on all the variables except x;
- has no solution.

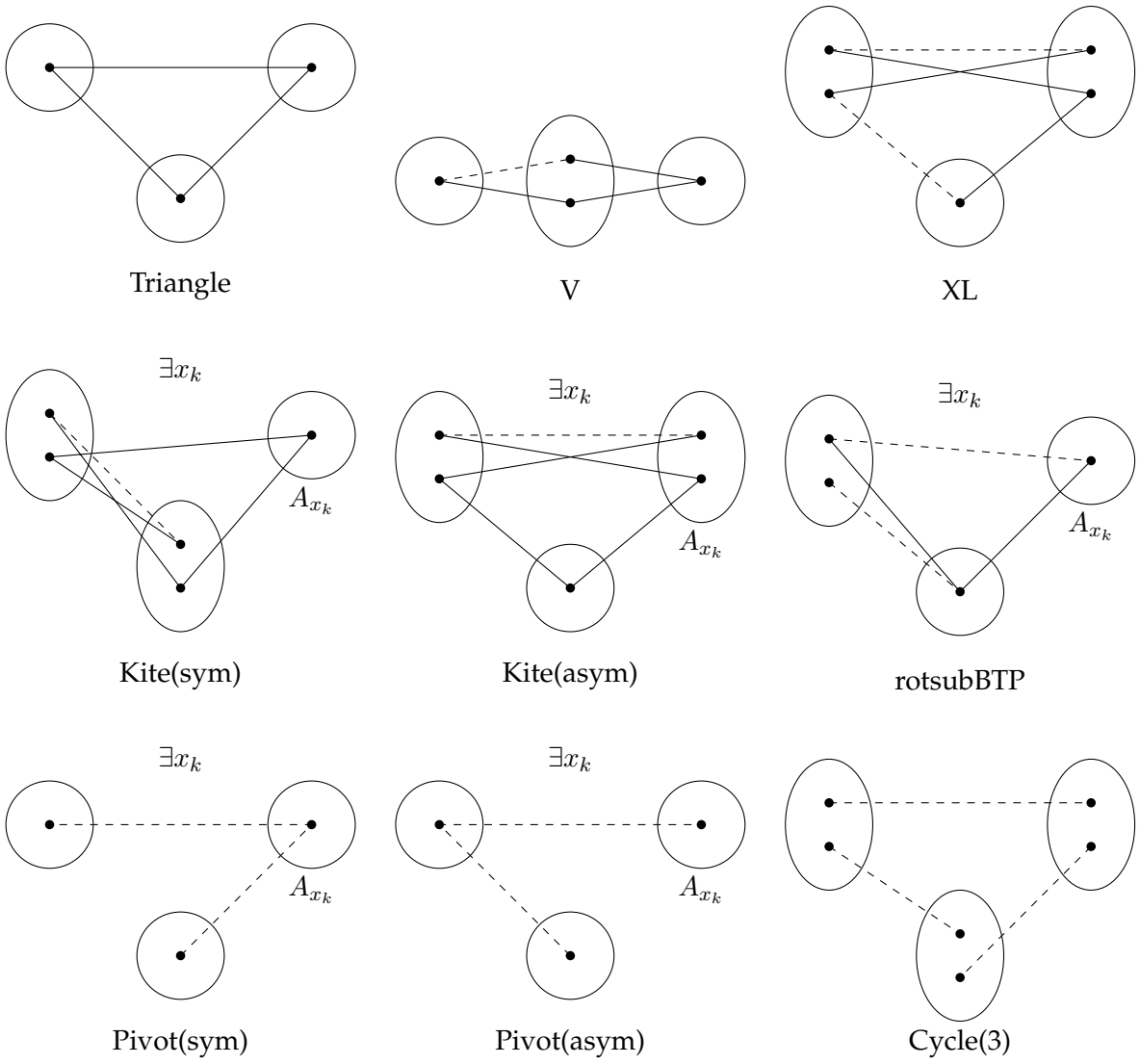


Figure 68: Patterns which do not allow variable elimination.

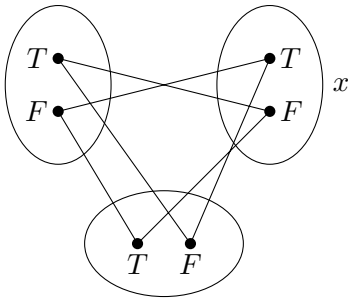


Figure 69: The instance I_3^{2COL}.

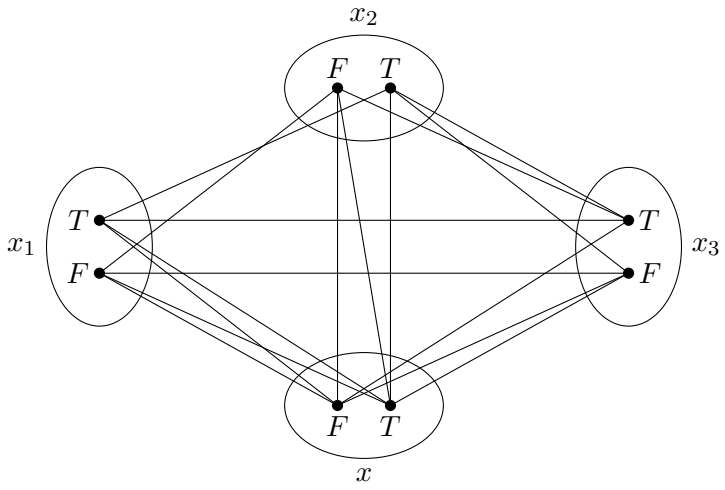


Figure 70: The instance I_4^{SAT}.

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

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

  Let I_3^{2COL} be the CSP instance (corresponding to 2-coloring on 3 variables) with three Boolean variables (domain is $\{T, F\}$ with meaning True and False), 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. I_3^{2COL} is represented in Figure 69. Only compatibility edges are drawn in the figure.

- 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: $x_1 \equiv x_2$, $x_1 \equiv x_3$, $x_2 \vee x_3$, $\overline{x_2} \vee x$, $\overline{x_3} \vee \overline{x}$.$I_4^{SAT}$ is represented in Figure 70. Only compatibility edges are drawn in the figure.

- For Cycle(3), or Pivot(asym).

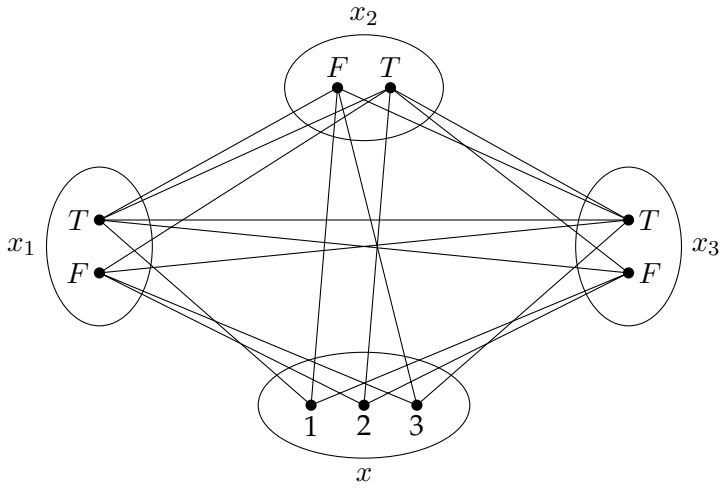


Figure 71: The instance I_4.

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: $\overline{x_1} \vee \overline{x_2}$, $\overline{x_1} \vee \overline{x_4}$, $x_1 \vee \overline{x_3}$, $x_1 \vee \overline{x_5}$, $x_2 \vee \overline{x}$, $x_4 \vee x$, $x_3 \vee \overline{x}$, $x_5 \vee x$.

- 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 the domain of x is $\{1, 2, 3\}$, with the following constraints: $x_1 \vee x_2$, $x_1 \vee x_3$, $x_2 \vee x_3$, $\overline{x_i} \Leftrightarrow (x \neq i)$ $(i = 1, 2, 3)$. I_4 is represented in Figure 71. Only compatibility edges are drawn in the figure.

- 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, x_3 and x take their values in $\{1, 2, 3\}$, with the following constraints: $x_1 = x_2$, $x_1 = x_3$, $x_2 = x_3$, $(x_1 = 1) \vee (x = 1)$, $(x_2 = 2) \vee (x = 2)$, $(x_3 = 3) \vee (x = 3)$. I_4^{ZOA} is represented in Figure 72. Only compatibility edges are drawn in the figure.

- 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 the domain of x_i is $\{0, 1, 2\}, i = 1, \ldots, 6$ and the domain of x is $\{0, 1\}$, with the following constraints:

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

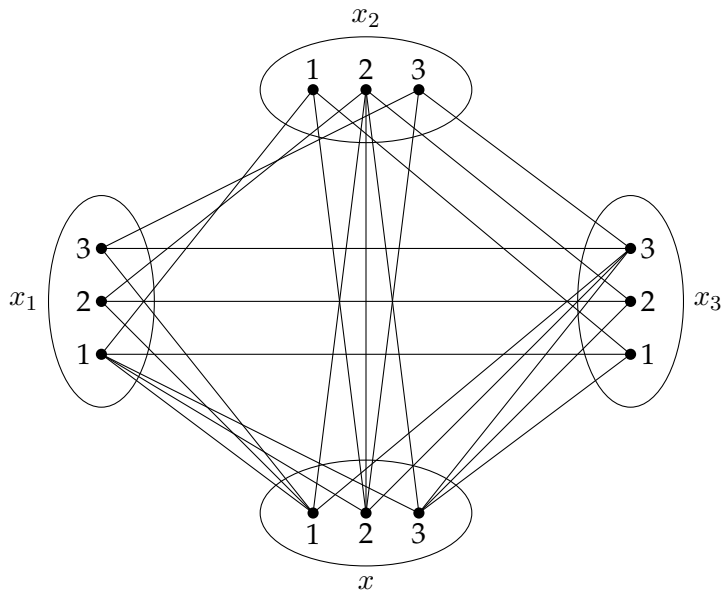


Figure 72: The instance I_4^{ZOA}.

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

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

□

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

Lemma 55. *The only flat or quantified reduced patterns with at least three variables that do not contain any of the patterns listed in Lemma 54 are contained in BTP, invsubBTP or snake(2).*

Proof. Consider a quantified reduced VE pattern $P = \langle X, A, var, E, cpt \rangle$ that does not contain any of the patterns listed in Lemma 54.

By Lemma 54 we know that P has at most three variables each with domain size strictly less than three.

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

Any reduced pattern that does not contain an incompatible pair of assignments must contain Triangle. What is more if any assignment is incompatible with two other assignments then it 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 73.

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.

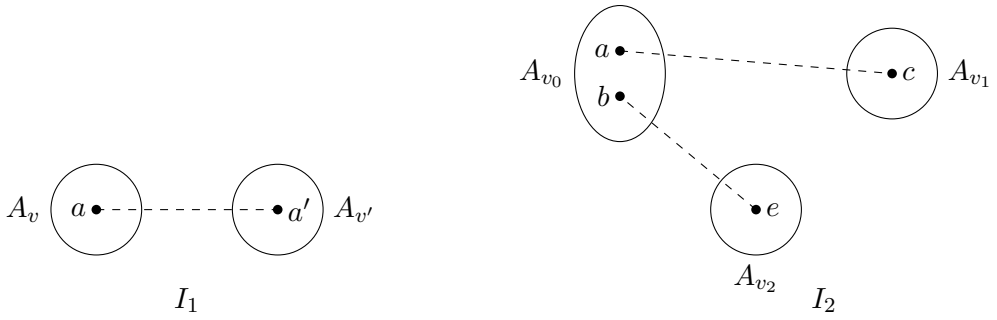


Figure 73: The possible negative skeletons of VE patterns.

Since the domains have at most two elements we begin by assuming that the domain of v_1 is $\{c, d\}$ and the domain of v_2 is $\{e, f\}$. 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 we can embed XL in P. Furthermore, d and f cannot be compatible since, whichever variable is chosen for $e(P)$ we can embed either Kite(sym) or Kyte(asym). It follows that d can be removed as it is a dangling assignment.

Now we begin again. As before to avoid merging a and b or embedding V we have that f is compatible with b and not compatible with a. So, f must be compatible with c to avoid a dangling point. In this case P contains Triangle.

Finally, we have the domain of v_1 equal to $\{c\}$ and the domain of v_1 equal to $\{e\}$. Suppose that there is a compatibility edge between c and e. If we put an existential quantifier on the variable v_0 then, whether or not there is a compatibility edge between a and e, the pattern is contained in BTP. If we put an existential quantifier on the variable v_1 and there is no compatibility edge between a and e, then the pattern is contained in invsubBTP. If we put an existential quantifier on the variable v_1 and there is a compatibility edge between a and e, then the pattern contains rotsubBTP. If we put an existential quantifier on the variable v_2, then the pattern contains rotsubBTP. So we have covered all cases in which there is a compatibility edge between c and e.

Whether or not the last edge between a and e is an incompatibility one, the pattern is contained in BTP if we put an existential quantifier on the variable v_0, and the pattern is contained in snake(2) if we put an existential quantifier on either v_1 or v_2.

The last case to consider is when P is a 3-variable pattern with $P^- = I_1$. Any two assignments for the third variable could be merged so we can assume it has domain size one.

Since P does not contain V, 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 we put an existential quantifier on either v or v', and contained in invsubBTP if we put an existential quantifier on v''. □

The following proposition is a direct consequence of Proposition 3 together with Lemma 54 and Lemma 55.

Proposition 4. *The irreducible flat or quantified patterns allowing variable elimination in arc-consistent binary CSP instances are BTP, invsubBTP or snake(2) (and their irreducible subpatterns).*

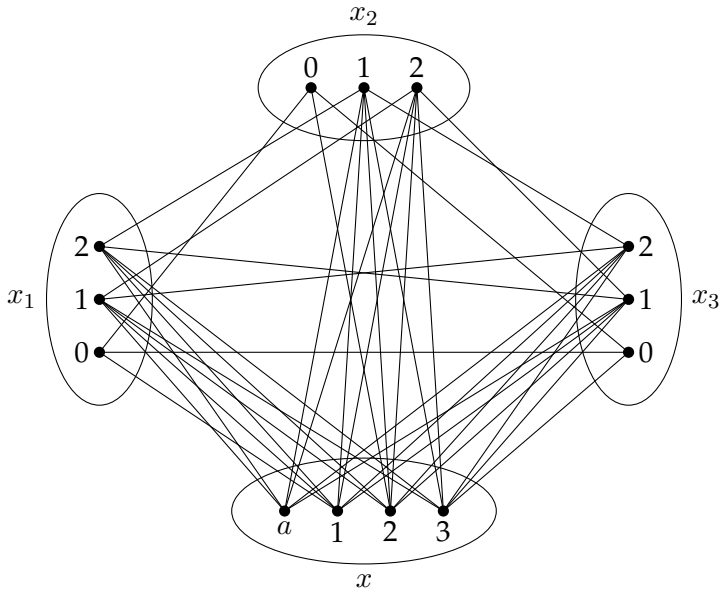


Figure 74: The instance I_4'.

We are now able to complete the characterisation with little extra work. We first demonstrate which existential patterns are not VE patterns.

Proposition 5. *The only irreducible existential patterns which allow variable elimination in arc-consistent binary CSP instances are ∃subBTP, ∃invsubBTP, ∃snake(2) (and their subpatterns).*

Proof. Consider the instance I_4', pictured in Figure 74 with four variables x_1, x_2, x_3 and x where the domain of x_1, x_2 and x_3 are all $\{0, 1, 2\}$ and the domain of x is $\{a, 1, 2, 3\}$. Only compatibility edges are drawn in the figure.

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 must have that $(x_i > 0) \vee (x = i)$.

The instance I_4' is arc-consistent, has a partial solution on variables $\{x_1, x_2, x_3\}$ but has no solution.

Let P be an existential pattern where $|e(P)| > 1$. Either P is not contained in I_3^{2COL} or does not occur at x in I_4', and hence is not a VE pattern.

So all existential VE patterns have precisely one existential value.

We know from Proposition 3 that ∃subBTP, ∃invsubBTP, ∃snake(2) are existential VE patterns.

Proposition 3 and Lemma 53 show that when we flatten an existential VE pattern then the resulting flat pattern is contained in BTP, invsubBTP or snake(2).

In the case of snake(2) and invsubBTP the maximal existential patterns contained in these patterns are VE patterns and so there is nothing left to prove.

We still have to consider subpatterns of the existential version of BTP shown in Figure 75. First observe that if we have only one pair of assignments in the domain of the compatibility function then it is a subpattern of ∃snake(2) and is a VE pattern.

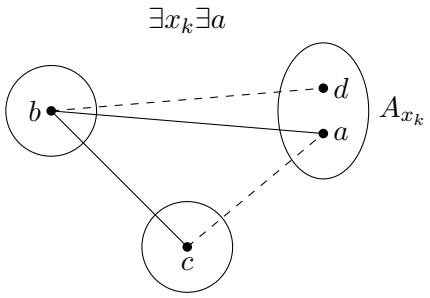


Figure 75: The final case: are any subpatterns of this pattern VE patterns?

The CSP I_4' has the property that, for each variable x_i, where $i = 1, 2, 3$, all values for x are compatible with the assignment $x_i = 1$. Hence, if we remove the assignment c from the pattern in Figure 75, the resulting existential pattern would allow the elimination of x from I_4' so is not VE.

It remains to consider subpatterns which do not include the incompatibility between d and b and hence have three domains of size one. This whole subpattern would allow variable elimination in I_3^{2COL} so is not VE.

All the non-trivial subpatterns now consist of two compatibility edges. Two have dangling reductions to trivial patterns. Hence, after reduction, all three are contained in ∃snake(2), and so are VE. □

Theorem 6. *The only irreducible patterns which allow variable elimination in arc-consistent binary CSP instances are BTP, snake(2), invsubBTP, ∃subBTP, ∃invsubBTP, ∃snake(2) (and their subpatterns).*

We have answered the question of which local obstructions allow variable elimination in binary CSPs.

5.2 Fusion of Subdomains

The aim of this section is to demonstrate that other techniques exist for reducing search-space size via the reduction of the total number of points. We give examples of such techniques based on the (partial) fusion of variable domains. We concentrate our attention on reduction operations whose complexity is comparable with known techniques, such as arc consistency and neighborhood substitution. In other words, we require that the complexity for detecting such reduction operations should be linear in the number of non-trivial constraints in the instance.

In a CSP instance, it is always possible to combine two variables v_1, v_2 so that the resulting variable v has a domain which is the cartesian product of the domains of v_1 and v_2. We are interested in special cases of this fusion operation in which the total number of points decreases. We first define a very general fusion operation for two variables. We then describe weaker versions which can be detected in complexity which is linear in the number of constraints.

Definition 51 (fusible points). *Consider a CSP instance $I = \langle V, A, var, E, cpt \rangle$ with $v_1, v_2 \in V$ and two points $a \in A_{v_1}$ and $b \in A_{v_2}$. Suppose that $\forall d \in A_{v_1} \forall e \in A_{v_2}$ such that d and e are compatible, either*

(1) $\forall u \in A \setminus (A_{v_1} \cup A_{v_2})$ such that u is compatible with both d and e, we have a compatible with both u and e, or (2) $\forall u \in A \setminus (A_{v_1} \cup A_{v_2})$ such that u is compatible with both d and e, we have b compatible with both u and d. We say that the points a, b are fusible.

Definition 52 (two-point fusion). *Consider a CSP instance $I = \langle V, A, var, E, cpt \rangle$ with $v_1, v_2 \in V$ and two points $a \in A_{v_1}$ and $b \in A_{v_2}$ such that a and b are fusible. Then we can perform the* two-point fusion *of v_1 and v_2 to create a new* fused *variable v. The resulting instance is $I' = \langle V', A', var', E', cpt' \rangle$ defined by:*

- $V' = (V \setminus \{v_1, v_2\}) \cup \{v\}$.
- $A' = (A \setminus (A_{v_1} \cup A_{v_2})) \cup \{p \in A_{v_1} \mid p \text{ is compatible with } b\} \cup \{q \in A_{v_2} \setminus \{b\} \mid q \text{ is compatible with } a\}$.
- $var'(u) = var(u)$ *for all* $u \in A' \setminus (A_{v_1} \cup A_{v_2})$ *and* $var'(u) = v$ *for all* $u \in A' \cap (A_{v_1} \cup A_{v_2})$.
- $E' = \{\{u, u'\} \mid (u, u') \subseteq A' \times A', var(u) \neq var(u')\}$.
- – $cpt'(p, q) = cpt(p, q)$ *if* $p, q \in A' \setminus A'_v$.
  – $cpt'(p, u) = T$ *if* $p \in A_{v_1} \wedge u \in A \setminus (A_{v_1} \cup A_{v_2}) \wedge cpt(p, b) = cpt(p, u) = cpt(b, u) = T$.
  – $cpt'(q, u) = T$ *if* $q \in A_{v_2} \wedge u \in A \setminus (A_{v_1} \cup A_{v_2}) \wedge cpt(a, q) = cpt(q, u) = cpt(u, a) = T$.
  – $cpt'(p, q) = F$ *otherwise.*

Proposition 6. *If $I = \langle V, A, var, E, cpt \rangle$ is a binary CSP instance and I' the result of applying a two-point fusion operation to I, then I' has a solution if and only if I has a solution.*

Proof. The two-point fusion operation effectively has the same effect as only allowing pairs of assignments of the form $(a, \cdot)$ or $(\cdot, b)$ to the variables (v_1, v_2) of I. It therefore suffices to show that any solution S to I can be transformed into another solution S' with either $a \in S'$ or $b \in S'$. Suppose that in S, variables (v_1, v_2) are assigned values (d, e). Then $de \in E$ and by Definition 51, either (1) $\forall u \in A \setminus (A_{v_1} \cup A_{v_2})$ such that u is compatible with both d and e, we have a compatible with both u and e, or (2) $\forall u \in A \setminus (A_{v_1} \cup A_{v_2})$ such that u is compatible with both d and e, we have b compatible with both u and d. In case (1), we can replace d by a in S to produce another solution S' containing the assignment a. In Case (2) we can replace e by b in S to produce another solution S' containing the assignment b. □

Remark 1. *If $D_1 = |A_{v_1}|$, $D_2 = |A_{v_2}|$ and $D = |A_v|$, we have $D \leq D_1 + D_2 - 1$. So after a two-point fusion, the total number of variables decreases by 1 and the total number of points decreases by at least 1.*

The complexity of applying a two-point fusion operation to particular points a, b which are already known to be fusible is easily seen to be $O(nd^2)$, where n is the number of variables and d the maximum domain size of the CSP instance. The detection of pairs of fusible points for all pairs of assignments $a, b \in A$ requires exhausting over five assignments to three variables. Therefore, it is interesting to identify weaker versions that can be tested faster. We omit the proofs of the following two propositions since they follow directly from Definition 51.

Proposition 7. *If $a \in A$ is incompatible with only one other point $b \in A$, then a, b are fusible.*

Proposition 8. *Suppose that $a \in A_{v_1}$ is compatible with all points in $A_{v_2} \setminus \{b\}$, where $var(b) = v_2$, and $\forall d \in A_{v_1} \setminus \{a\}$, $\forall e \in A_{v_2} \setminus \{b\}$ such that d and e are compatible, we have $\forall u \in A \setminus (A_{v_1} \cup A_{v_2})$ if d is compatible with u then a is compatible with u. Then a, b are fusible.*

The premises of Propositions 7, 8 can be checked for all pairs $a, b \in A$ in time $O(cd^2)$and $O(cd^4)$ respectively, where c is the number of non-trivial binary constraints in the CSP instance, using suitable data structures. For example, it is clearly $O(cd^2)$ in time and space to construct the data structure inc, where $\forall a \in A$, $inc(a)$ is the set of variables $v \neq var(a)$ with a point $p \in A_v$ which is incompatible with a. It also requires only $O(cd^3)$ time and $O(cd^2)$ space to construct the data structure $notSub$, where $\forall a, d \in A$ with $var(a) = var(d)$, $not - sub(a, d)$ is the set of variables v such that $\exists p \in A_v$ with $du \in E$ and $au \notin E$. Using these data structures, testing the premises of Propositions 8 can be achieved by $O(1)$ operations for each quadruple (a, b, d, e) such that $a, d \in A_{v_1}$, $b, e \in A_{v_2}$ and there is a non-trivial constraint between variables v_1,v_2.

When we perform a two-point fusion operation, the number of variables and the number of assignments both decrease. In this section, we consider a different form of fusion, which we call subdomain fusion, in which the number of variables remains constant but the number of assignments decreases. Whereas two-point fusion can be applied to points with few incompatibilities, subdomain fusion can be applied to points with few compatibilities. As in the two-point fusion operation, two assignments to a pair of variables can be fused to form a single assignment. However, this can now happen on subdomains of two variables instead of the whole domains. As a result, we need to keep both original variables. The fused assignments are effectively placed in the domain of one of the variables, all of these new assignments are compatible with a new dummy assignment to the other variable.

Definition 53 (fusible subdomains). *Consider a CSP instance $I = \langle V, A, var, E, cpt \rangle$ with $v_1, v_2 \in V$. For $B_1 \subset A_{v_1}$, $B_2 \subset A_{v_2}$ let $C(B_1, B_2)$ denote $\{(p, q) \mid (p \in B_1) \wedge (q \in B_2) \wedge p$ and q are compatible$\}$. If $\forall p \in B_1, \forall q \in A_{v_2} \setminus B_2$ we have p is incompatible with q and if $\forall p \in A_{v_1} \setminus B_1, \forall q \in B_2$ we have p is incompatible with q and if $|E(B_1, B_2)|+1 < |B_1|+|B_2|$, then we say that B_1, B_2 are* fusible subdomains.

Definition 54 (subdomain fusion). *Consider a CSP instance $I = \langle V, A, var, E, cpt \rangle$ with $v_1, v_2 \in V$. Let $B_1 \subset A_{v_1}$ and let $B_2 \subset A_{v_2}$. We can perform the* subdomain fusion *of B_1 and B_2 by transforming I into the instance $I' = \langle V', A', var', E', cpt' \rangle$ defined as follows. All elements of $C(B_1, B_2)$ (pairs of compatible assignments to variables v_1, v_2 in $B_1 \times B_2$) are now considered as points (assignments to the single variable v_1) and a new dummy assignment d is created for variable v_2. Thus:*

- $V' = V$.
- $A' = (A \setminus (B_1 \cup B_2)) \cup C(B_1, B_2) \cup \{d\}$.
- $var'(u) = var(u)$ *for all* $u \in A' \setminus (C(B_1, B_2) \cup \{d\})$, $var'(u) = v_1$ *for all* $u \in C(B_1, B_2)$ *and* $var(d) = v_2$.
- $E' = \{\{u, u'\} \mid (u, u') \subseteq A' \times A', var(u) \neq var(u')\}$.

- – $cpt'(p,q) = cpt(p,q)$ *if* $p,q \in A' \setminus (C(B_1,B_2) \cup \{d\})$.
  – $cpt'((p,q),u) = T$ *if* $(p,q) \in C(B_1,B_2) \wedge u \in A \setminus (A_{v_1} \cup A_{v_2}) \wedge cpt(p,u) = cpt(q,u) = T$.
  – $cpt'(u,d) = T$ *if* $u \in C(B_1,B_2) \cup (A' \setminus (A_{v_1} \cup A_{v_2}))$.
  – $cpt'(p,q) = F$ *otherwise.*

Remark 2. *A subdomain fusion operation reduces the total number of points (variable-value assignments) since* $|A'| = |A| - (|B_1| + |B_2|) + |C(B_1,B_2)| + 1 < |A|$*) by definition of fusible subdomains.*

Proposition 9. *If* $I = \langle V, A, var, E, cpt\rangle$ *is a binary CSP instance and* $I' = \langle V', A', var', E', cpt'\rangle$ *the result of applying a subdomain fusion operation to* I*, then* I' *has a solution if and only if* I *has a solution.*

Proof. Suppose first that the operations has been applied to two variables v_1 and v_2, with $v_1, v_2 \in V$. Suppose that the fused subdomains are $B_1 \subset A_{v_1}$ and $B_2 \subset A_{v_2}$. Suppose also that d is the new dummy assignment for variable v_2.
Suppose that I has a solution S. Let a_1 be the point of S in A_{v_1} and let a_2 be the point of S in A_{v_2}. Since B_1 and B_2 are fusible subdomains, there is no compatibility edge between B_1 and $A_{v_2} \setminus B_2$, nor between $A_{v_1} \setminus B_1$ and B_2. Therefore, either $a_1 \in B_1$ and $a_2 \in B_2$, or $a_1 \in A_{v_1} \setminus B_1$ and $a_2 \in A_{v_2} \setminus B_2$. In the latter case, S is also a solution for I'. In the former case, let $c = (a_1, a_2)$. c is an edge between v_1 and v_2 in I, but it is also a point in A_{v_1} in I'. From the third case of the construction of cpt' in the definition of subdomain fusion, we know that c is compatible with d in I'. From the second case of the construction of cpt' in the definition of subdomain fusion, we know that c is compatible with all points of $S \subset \{a_1, a_2\}$ in I'. From the third case of the construction of cpt' in the definition of subdomain fusion, we know that d is compatible with all points of $S \subset \{a_1, a_2\}$ in I'. So after replacing a_1 by c and a_2 by d in S, we have a solution S' for I'. Therefore, if there is a solution for I, then there is also a solution for I'.
Suppose now that there is a solution S' for I'. Let c_1 be the point of S' in A_{v_1} and let c_2 be the point of S' in A_{v_2}. From the fourth case of the construction of cpt' in the definition of subdomain fusion, there is no compatibility edge between $C(B_1,B_2)$ and $A_{v_2} \setminus d$, nor between $A_{v_1} \setminus (C(B_1,B_2))$ and d. Therefore, either $c_1 \in C(B_1,B_2)$ and $c_2 = d$, or $c_1 \in A_{v_1} \setminus (C(B_1,B_2))$ and $c_2 \in A_{v_2} \setminus d$. In the latter case, S' is also a solution for I. In the former case, we know that $c_1 = (a_1, a_2)$, with $a_1 \in B_1$ and $A_2 \in B_2$. Since $(a_1, a_2) \in C(B_1,B_2)$, a_1 and a_2 are compatible in I. Since c_1 is compatible with all points of $S' \setminus \{c_1, d\}$, we know from the second case of the construction of cpt' in the definition of subdomain fusion that both a_1 and a_2 are compatible with all points of $S' \setminus \{c_1, d\}$ in I. So after replacing c_1 by a_1 and c_2 by a_2 in S', we have a solution S for I. Therefore, if there is a solution for I', then there is also a solution for I and we have the proposition.

□

Consider the graph $G = \langle G_1, G_2\rangle$, where $G_1 = A_{v_1} \cup A_{v_2}$ is the set of vertices of G and $G_2 = \{(p,q) \mid p \in A_{v_1} \wedge q \in A_{v_2} \wedge cpt(p,q) = T\}$ is the set of edges of G. Let $C_1, \ldots, C_r$ be the connected components of G. Subdomains B_1, B_2 are fusible only if $B_1 \cup B_2$ is the set of vertices in some connected components $C_{i_1}, \ldots, C_{i_t}$ of G since $B_1 \cup B_2$ must not be connected to the other vertices of G. In order to find the subdomains B_1, B_2 of A_{v_1}, A_{v_2} which maximize $(|B_1|+|B_2|)-(|C(B_1,B_2)|+1)$ (the reduction in the number of points during subdomain fusion) we simply need to find the

union of those connected components which are trees. Indeed, if C_i with vertices $B'_1 \subseteq A_{v_1}$, $B'_2 \subseteq A_{v_2}$ is a tree, then $|B'_1| + |B'_2| = |C(B'_1, B'_2)| + 1$, otherwise $|B'_1| + |B'_2| < |C(B'_1, B'_2)| + 1$. It follows that we can perform a subdomain fusion if and only if there are at least two trees among the connected components of G. Thus, we can detect whether it is possible to perform a subdomain fusion somewhere in the instance in time $O(cd^2)$.

A special case of subdomain fusion occurs when the subdomains are the complete domains. In this case, we do not need to keep the variable v_2 after the fusion operation since it has a singleton domain containing only the dummy point d. Thus, in this case, we can reduce the total number of points in the instance (and the number of variables) if $|B'_1| + |B'_2| = |C(B'_1, B'_2)| + 1$. In other words, we only require that the the graph G, representing of the binary constraint between the two variables v_1, v_2, is a tree (or a forest). Common examples of binary constraints whose graph is a forest include the disjunctive constraint $(v_1 = a) \vee (v_2 = b)$ and bijective constraints such as $v_1 = v_2$.

Remark: Two-point fusion and subdomain fusion are not comparable with the fusion operations from Section 3.3.2.

We have demonstrated the existence of novel low-order polynomial time simplification operations for binary CSPs which reduce the size of an instance, in terms of the number of microstructure vertices.

6 Conclusion

6.1 What we have done

We now summarize the major contributions of this thesis.

We have formally defined several versions of forbidden patterns in the context of the binary Constraint Satisfaction Problem, each one tailored to be the most practical for the use we intended it for. We then presented a new tool for forbidden patterns, which we called "reduction between patterns", by analogy with the similarly named operation which exposes the complexity relations between different problems. We went on to show that from a small number of complexity results, this new operation allows us to build tractability dichotomies on a large scope of CSP problems. We then proceeded to use all these notions to reveal several tractable classes. We finally provided various simplification operations based on forbidden pattern or similar concepts, each operation decreasing the size of the CSP instances it is applied to.

What this thesis is about in a nutshell:

- Formal definition of a forbidden pattern in the binary CSP context. Different versions for different uses:
  - Flat patterns.
  - Quantified patterns.
  - Existential patterns.
- New tool: reduction between patterns.
- Several dichotomies:
  - A dichotomy for the complexity of classes of CSP instances defined by a flat pattern on two constraints.
  - A dichotomy for the complexity of classes of CSP instances defined by an existential pattern on two constraints.
  - A dichotomy for the complexity of classes of CSP instances defined by a Max-CSP subproblem.
  - A dichotomy for the patterns allowing variable elimination.
- Several miscellaneous results:
  - A set of necessary conditions for the tractability of classes of CSP instances defined by several Max-CSP subproblems.
  - A set of necessary conditions for the tractability of classes of CSP instances defined by a flat pattern on three variables $\Leftrightarrow$ a list of all tractable and open flat patterns on three variables.
  - Some tractability proofs for flat patterns on three variables.

- Some simplification operations decreasing the size of CSP instances:
  - Fusion of two domains.
  - Fusion of two subdomains.

6.2 What we can do next

The field of forbidden patterns has only been entered quite recently, and as a result we are among the first to try to map it. The steps we have taken represent a considerable progress; nonetheless there is much left to do. The first avenue of research that merits further investigation is the completion of the characterization of the complexity of forbidden patterns on three variables. Although there are still many cases left open, the reduction tool we introduced means that we might finish the result with only a few more tractability (or NP-Completeness) proofs. The main avenues that we'd like to explore afterwards are:

1. Generalizing our results for the CSP to Max-CSP and Valued-CSP.
2. Generalizing our results on binary patterns to patterns of any arity.
3. Generalizing our results on fobidding one pattern to forbidding several patterns at once.
4. Generalizing our results on three-variable flat patterns to existential patterns.
5. Generalizing our results on three-variable patterns to patterns with more variables.

The main issue one might encounter when trying to deal with point 1 or point 2 is the proper convention to adopt for the definition of a generalized pattern. The generalization is very straightforward in the case of Max-CSP patterns, which is one of the reasons we chose to study that kind of patterns, but for the other ones there are several distinct possibilities, all equally valid, for how to define a forbidden pattern. However, there is no reason to restrict oneself to only one path. Therefore what seems to be an obstacle at first, might reveal itself in the end to be just different trails leading to a large number of distinct and diverse tractable classes.

Point 3 is not only interesting in itself, but also because it is actually an open question whether classes which can only be defined by forbidding two or more patterns actually exist.

There are also some interesting follow-up questions to ponder:

6. What happens when we place an ordering structure on the variables of our tractable patterns?
7. What can we find if we look only for value elimination instead of variable elimination?
8. How to find a solution for an instance belonging to a tractable class defined by a forbidden pattern?

The inspiration for point 6 is that it has already be done for one of the patterns: BTP (Cooper et al., 2010). Finding tractable classes defined in this way would be a very valuable enrichment of our results. Point 7 is a work in progress which will be completed in the near future. As for point 8, it is worth noting that most of our tractability proofs are constructive, hence it is possible to infer a solution for such instances. Still, extracting explicit solving algorithms would be a small, but useful, step forward.

List of Definitions

List of Figures

References

Bertelé, U., & Brioshi, F. (1972). *Nonserial dynamic programming*. Academic Press.

Bessière, C., & Debruyne, R. (2001). Domain filtering consistencies. *J. Artif. Intell. Res. (JAIR)*, *14*, 205–230.

Bessière, C., Martinez, D., & Verfaillie, G. (1999). A generic customizable framework for inverse local consistency. In *AAAI*, pp. 169–174.

Bessière, C., Régin, J.-C., Yap, R. H. C., & Zhang, Y. (2005). An optimal coarse-grained arc consistency algorithm. *Artificial Intelligence*, *165*(2), 165–185.

Bulatov, A., Jeavons, P., & Krokhin, A. (2005). Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, *34*(3), 720–742.

Bulatov, A. A. (2003). Tractable conservative constraint satisfaction problems. In *LICS 2003: Proceedings of 18th IEEE Symposium on Logic in Computer Science*, pp. 321–330.

Bulatov, A. A. (2006). A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, *53*(1), 66–120.

Chen, X., & Van Beek, P. (2001). Conflict-directed backjumping revisited. *J. Artif. Intell. Res. (JAIR)*, *14*, 53–81.

Cohen, D. A. (2003). A New Class of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In *CP 2003*, pp. 807–811. LNCS 2833, Springer.

Cohen, D. A., Cooper, M. C., Creed, P., Marx, D., & Salamon, A. Z. (2012). The tractability of csp classes defined by forbidden patterns. *J. Artif. Intell. Res. (JAIR)*, *45*, 47–78.

Cohen, D. A., Cooper, M. C., Escamocher, G., & Živný, S. (2013). Variable elimination in binary csp via forbidden patterns. In *IJCAI*, pp. 517–523.

Cohen, D. A., Cooper, M. C., & Jeavons, P. (1994). Characterising tractable constraints. *Artificial Intelligence*, *65*(2), 347–361.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In Harrison, M. A., Banerji, R. B., & Ullman, J. D. (Eds.), *STOC*, pp. 151–158. ACM.

Cooper, M. C., De Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., & Werner, T. (2010). Soft arc consistency revisited. *Artificial Intelligence*, *174*, 449–478.

Cooper, M. C. (1997). Fundamental properties of neighbourhood substitution in constraint satisfaction problems. *Artificial Intelligence*, *90*(1-2), 1–24.

Cooper, M. C., De Roquemaurel, M., & Régnier, P. (2001). A weighted CSP approach to cost-optimal planning. *Artificial Intelligence Communications*, *24*(1), 1–29.

Cooper, M. C., & Escamocher, G. (2012). A dichotomy for 2-constraint forbidden csp patterns. In *AAAI*, pp. 464–470.

Cooper, M. C., Escamocher, G., & Živný, S. (2012). A characteristion of the complexity of forbidding subproblems in binary max-csp. In *CP*, pp. 265–273.

Cooper, M. C., Jeavons, P. G., & Salamon, A. Z. (2010). Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artificial Intelligence*, *174*(9–10), 570–584.

Cooper, M. C., & Živný, S. (2011a). Hierarchically nested convex VCSP. In *CP 2011*, pp. 187–194. LNCS 6876, Springer.

Cooper, M. C., & Živný, S. (2011b). Hybrid tractability of valued constraint problems. *Artificial Intelligence*, *175*(9-10), 1555–1569.

Cooper, M. C., & Živný, S. (2011c). Tractable triangles. In *CP 2011*, pp. 195–209.

Cooper, M. C., & Živný, S. (2012). Tractable triangles and cross-free convexity in discrete optimisation. *J. Artif. Intell. Res. (JAIR)*, *44*, 455–490.

Creignou, N., Khanna, S., & Sudan, M. (2001). Complexity classification of boolean constraint satisfaction problems. *SIAM Monographs on Discrete Mathematics and Applications*, *7*.

Dalmau, V., Kolaitis, P., & Vardi, M. (2002). Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP 2002*, pp. 310–326. LNCS 2470, Springer.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Deineko, V., Jonsson, P., Klasson, M., & Krokhin, A. (2008). The approximability of Max CSP with fixed-value constraints. *Journal of the ACM*, *55*(4).

Elfe, C., & Freuder, E. (1996). Neighborhood inverse consistency preprocessing. In *AAAI*, pp. 202–208.

Freuder, E. C. (1991). Eliminating interchangeable values in constraint satisfaction problems. In Dean, T. L., & McKeown, K. (Eds.), *AAAI*, pp. 227–233. AAAI Press / The MIT Press.

Gao, J., Yin, M., & Zhou, J. (2011). Hybrid tractable classes of binary quantified constraint satisfaction problems. In *AAAI*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

Gent, I., Jefferson, C., & Miguel, I. (2006). Watched literals for constraint propagation in minion. In *CP 2006*, pp. 182–197. LNCS 4204, Springer.

Grohe, M. (2007). The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, *54*(1), 1–24.

Henderson, T. C., & Mohr, R. (1986). Arc and path consistency revisited. *Artificial Intelligence*, *28*, 225–233.

Jégou, P. (1993). Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems. In *AAAI '93*, pp. 731–736.

Jonsson, P., Klasson, M., & Krokhin, A. (2006). The approximability of three-valued max csp. *SIAM J. Comput.*, *35*(6), 1329–1349.

Jonsson, P., Kuivinen, F., & Thapper, J. (2011). Min csp on four elements: Moving beyond submodularity. In *CP 2011*, pp. 438–453. LNCS 6876, Springer.

Kamiński, M. (2010). Max-cut and containment relations in graphs. In *WG 2010 - 36th International Workshop on Graph Theoretic Concepts in Computer Science*, pp. 15–26.

Kolmogorov, V. (2013). The power of linear programming for finite-valued CSPs: a constructive characterization. In *ICALP*, pp. 625–636. LNCS 7965, Springer.

Larrosa, J., & Dechter, R. (2003). Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, *8*(3), 303–326.

Lecoutre, C. (2009). *Constraint Networks: Techniques and Algorithms*. ISTE/Wiley.

Lewis, J., & Yannakakis, M. (1980). The node-deletion problem for hereditary properties is np-complete. *Journal of Computer System Sciences*, *20*(2), 219–230.

Likitvivatanavong, C., & Yap, R. (2013). Eliminating Redundancy in CSPs Through Merging and Subsumption of Domain Values. *ACM SIGAPP Applied Computing Review*, *13*(2).

Mackworth, A. K. (1977). Consistency in network of relations. *Artificial Intelligence*, *8*, 99–118.

Marx, D. (2010a). Can you beat treewidth?. *Theory of Computing*, *6*(1), 85–112.

Marx, D. (2010b). Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pp. 735–744. ACM.

Meyer, A. R., & Stockmeyer, L. (1972). The equivalence problem for regular expressions with squaring requires exponential space. In *13th IEEE Symposium on Switching and Automata Theory*, pp. 125–129.

Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming Ser. A*, *118*(2), 237–251.

Prosser, P. (1993). Hybrid algorithms for the constraint satisfaction problem. *Computationnal Intelligence*, *9*(3), 268–299.

Raghavendra, P. (2008). Optimal algorithms and inapproximability results for every CSP?. In *STOC 2008*, pp. 245–254.

Rossi, F., Van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. Elsevier.

Salamon, A. Z., & Jeavons, P. G. (2008). Perfect constraints are tractable. In *CP 2008*, Vol. 5202 of *Lecture Notes in Computer Science*, pp. 524–528. Springer.

Shamir, R., Sharan, R., & Tsur, D. (2004). Cluster graph modification problems. *Discrete Applied Mathematics*, *144*, 173–182.

Thapper, J., & Živný, S. (2012). The power of linear programming for valued CSPs. In *FOCS 2012*, pp. 669–678.

Thapper, J., & Živný, S. (2013). The complexity of finite-valued CSPs. In *STOC 2013*, pp. 695–704.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, *42*(2), 230–265.

Waltz, D. (1975). Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, pp. 19–91. Winston, P.H.

Résumé

Le problème de satisfaction de contraintes (CSP) est NP-complet, même dans le cas où toutes les contraintes sont binaires. Cependant, certaines classes d'instances CSP sont traitables. Récemment, une nouvelle méthode pour définir de telles classes a émergée. Cette approche est centrée autour des motifs interdits, ou l'absence locale de certaines conditions. Elle est l'objet de ma thèse.

Nous définissons formellement ce que sont les motifs interdits, présentons les propriétés qu'ils détiennent, et finalement les utilisons afin d'établir plusieurs résultats de complexité importants. En utilisant différentes versions de motifs, toutes basées sur le même concept de base, nous énumérons un nombre important de nouvelles classes traitables, ainsi que certaines NP-completes. Nous combinons ces résultats pour révéler plusieurs dichotomies, chacune englobant une large gamme de classes d'instances CSP.

Nous montrons aussi que les motifs interdits représentent un outil intéressant pour la simplification d'instances CSPs. Nous donnons plusieurs nouveaux moyens de réduire la taille des instances CSP, que ce soit en éliminant des variables ou en fusionnant les domaines, et montrons comment ces méthodes sont activées par l'absence locale de certains modèles. Comme les conditions de leur utilisation sont entièrement locales, nos opérations peuvent être utilisés sur un large éventail de problèmes.

Mots clés:
problème de satisfaction de contraintes, motif interdit, classe traitable, élimination de variables

Abstract

The Constraint Satisfaction Problem (CSP) is NP-Complete, even in the case where all constraints are binary. However, some classes of CSP instances are tractable. Recently, a new method for defining such classes has emerged. This approach is centered around forbidden patterns, or the local absence of some conditions. It is the focus of my thesis.

We formally define what forbidden patterns are, exhibit the properties they hold, and eventually put them to use in order to establish several important tractability results. Using different versions of patterns, all based on the same core concept, we list a significant number of new tractable classes, as well as some NP-Complete ones. We combine these results to reveal several dichotomies, each one encompassing a large range of classes of CSP instances.

We also show how useful a tool forbidden patterns can be in the field of CSP instance simplification. We give multiple new ways of decreasing the size of CSP instances, whether by eliminating variables or fusioning domains, and prove how all these methods are enabled by the local absence of some patterns. Since the conditions for their use are entirely local, our operations can be used on a wide array of problems.

Keywords:
constraint satisfaction problem, forbidden pattern, tractable class, variable elimination