Chapter 1

INTRODUCTION TO CONSTRAINT SATISFACTION PROBLEMS

A *constraint satisfaction problem* (*CSP*) is a computational problem where we are given a finite set of variables and a finite set of *constraints* and where the task is to decide whether values can be assigned to the variables so that all the constraints are satisfied. Well-known examples of CSPs are the satisfiability problem for systems of linear equations over the two-element field, the satisfiability problem for systems of linear inequalities over the rational numbers, and the three-colouring problem for undirected graphs, just to name three. CSPs appear in almost every area of theoretical computer science, for instance in artificial intelligence, scheduling, computational linguistics, optimisation, computational biology, and verification. Many computational problems studied in those areas can be modelled by appropriately choosing a set of constraint types, the *constraint language* (or *template*), that are allowed in the input instance of a CSP. In the last decade, huge progress was made to find general criteria for constraint languages that imply that the corresponding CSP can be solved efficiently.

The complexity of CSPs became a topic that vitalises the field of universal algebra as it turned out that questions about the computational complexity of CSPs translate to important universal-algebraic questions about algebras that can be associated to CSPs. This approach is now known as the *algebraic approach* to constraint satisfaction complexity. The algebraic approach has raised questions that are of central importance in universal algebra. The so-called *dichotomy conjecture* of Feder and Vardi [170] stated that every CSP *with a finite domain* is either polynomial-time tractable (i.e., in P) or NP-complete. According to a well-known result by Ladner, it is known that there are NP-*intermediate* computational problems, i.e., problems in NP that are neither tractable nor NP-complete (unless P = NP). But the known NP-intermediate problems are extremely artificial. It would be interesting from a complexity theoretic perspective to discover more natural candidates for NP-intermediate problems. Two positive solutions to the dichotomy conjecture, both using the universal-algebraic approach, have been announced in 2017 by Bulatov and by Zhuk [111, 348].

The problems in the literature that can be described by specifying a constraint language over a finite domain, and that have been studied independently from the CSP framework, are quite limited, and mostly focussed on specialised graph theoretic problems or Boolean satisfiability problems. If we consider the class of all problems that can be formulated by specifying a constraint language over an *infinite* domain, the situation changes drastically. Many problems that have been studied independently in temporal reasoning, spatial reasoning, phylogenetic reconstruction, and computational linguistics can be directly formulated as CSPs over an infinite domain. Also feasibility problems in linear (and also non-linear, or tropical) programming (over the rationals, the integers, or other domains) can be cast as CSPs.

In this book we present a generalisation of the universal-algebraic approach to infinite domains. It turns out that this is possible when the constraint language, viewed as a relational structure $\mathfrak{B}$ with an infinite domain, is ω-*categorical* (or *countably categorical*), i.e., its first-order theory has at most one countable model up to isomorphism. An alternative characterisation of ω-categoricity of $\mathfrak{B}$ which is most useful in our context is in terms of the automorphism group of $\mathfrak{B}$: a structure $\mathfrak{B}$ is ω-categorical if and only if the automorphism group of $\mathfrak{B}$ is *oligomorphic*, i.e., for every $n \in \mathbb{N}$ the automorphism group of $\mathfrak{B}$ has only finitely many orbits in its componentwise action on n-tuples of elements of $\mathfrak{B}$. Many of the CSPs in the mentioned application areas can be formulated with ω-categorical constraint languages—in particular, problems coming from so-called *qualitative calculi* in artificial intelligence tend to have formulations with ω-categorical constraint languages. While ω-categoricity is quite a strong assumption from a model-theoretic point of view (and, for example, constraint languages for linear programming cannot be ω-categorical), the class of computational problems that can be formulated

with ω-categorical constraint languages is still a very large generalisation of the class of finite-domain CSPs. This will be amply demonstrated by examples of ω-categorical constraint languages from many different areas in computer science in Chapter 5.

There are several general results for ω-categorical structures that are important when studying the computational complexity of the respective CSPs. We highlight some of these general results, providing pointers into the text where all the involved technical terms will be gently introduced.

- Every finite or countably infinite ω-categorical structure is homomorphically equivalent to a finite or countably infinite ω-categorical structure which is *model complete* and a *core* (Section 2.6). Model-complete cores have many good properties: for example, they have quantifier elimination once expanded by all primitively positively definable relations. Moreover, they can be expanded by finitely many singleton relations $\{a\}$ without changing the complexity of the CSP. Since homomorphically equivalent structures have the same CSP, we can therefore focus on structures having these properties.
- The so-called *polymorphism clone* of an ω-categorical structure $\mathfrak{B}$ fully captures the computational complexity of the corresponding CSP (Chapter 6). Every polymorphism clone gives rise to a *topological clone* with respect to the pointwise convergence topology. Topological clones are defined analogously to topological groups and are a promising new subject in mathematics. We will see that the complexity of the CSP is already captured by the polymorphism clone viewed as a topological clone (Section 9.4).
- Finally, if the expansion of an ω-categorical model-complete core by finitely many singleton relations does not interpret primitively positively all finite structures, then it must have a pseudo-Siggers polymorphism (see Section 10.2). For finite domains, the existence of such polymorphisms turned out to be the dividing line between the polynomial-time tractable and the NP-hard CSPs.

Despite all these general results for ω-categorical structures, the class of all ω-categorical structures is still too large to hope for a complete complexity classification. It is easy to see that there are ω-categorical structures $\mathfrak{B}$ whose CSP is undecidable, and, as we will see in Chapter 13, there are CSPs of various other complexities: for example there are ω-categorical structures with a coNP-complete CSP, or with a CSP that is contained in coNP but neither coNP-hard nor in P.

However, there is a natural subclass of the class of all ω-categorical structures that might exhibit a complexity dichotomy as well. Formally, we consider the CSPs for *reducts of finitely bounded homogeneous structures*; these notions will be introduced in the text. Such structures have a finite representation

and their CSPs are always in NP. The class of CSPs of reducts of finitely bounded homogeneous structures contains all CSPs over finite domains, but also contains many additional CSPs from the mentioned application areas. Moreover, every CSP in the complexity class MMSNP (for Monotone Monadic Strict NP, treated in Section 5.6.2) can be formulated in this way. It follows from general principles that if the model-complete core template for such a CSP does not have a pseudo-Siggers polymorphism, then the CSP is NP-hard. We conjecture that otherwise, the CSP ought to be in P, and refer to this as the *infinite-domain tractability conjecture*. This conjecture has been confirmed in numerous special cases:

- For all finite-domain CSPs [348, 111];
- for all first-order reducts of
  - $(\mathbb{Q}; <)$ [69] (see Chapter 12),
  - the countable random graph [90],
  - the model companion of the class of all C-relations [64],
  - the universal homogeneous poset [241],
  - all homogeneous undirected graphs [80],
  - all unary structures [81];
- for all CSPs in the complexity class MMSNP [75].

Any outcome of the infinite-domain dichotomy conjecture for reducts of finitely bounded homogeneous structures is significant: a negative answer might provide relatively natural NP-intermediate problems, which would be of interest for complexity theorists. A positive answer probably comes with a criterion which describes the NP-hard CSPs, and it would probably provide algorithms for the polynomial-time tractable CSPs. But then we would have a fascinatingly rich catalogue of computational problems where the computational complexity is known. Such a catalogue would be a valuable tool for deciding the complexity of computational problems: since CSPs for finitely bounded homogeneous structures are abundant, one may derive algorithmic results by reducing the problem of interest to a known tractable CSP, and one may derive hardness results by reducing a known NP-hard CSP to the problem of interest.

An important feature of the universal-algebraic approach is that the tractability of a CSP can be linked to the existence of polymorphisms of the constraint language. This link can be exploited in several directions: first, if we already know that a constraint language of interest has a polymorphism satisfying good properties, then this polymorphism can guide the search for an efficient algorithm for the corresponding CSP. Another direction is that we already have an algorithm (or an algorithmic technique), and that we want to know for which CSPs the algorithm is a correct decision procedure: again, polymorphisms are the key tool for this task. Finally, we may use the absence of polymorphisms with good properties to prove that a CSP is

NP-hard. There are several instances where these three directions of the algebraic approach have been used very successfully for CSPs with finite domain constraint languages [113, 126, 212, 21, 111, 348] or ω-categorical constraint languages [69, 90, 64, 80].

Another tool that becomes useful specifically for polymorphisms over infinite domains is *Ramsey theory* (Chapter 11). The basic idea here is to apply Ramsey theory to show that polymorphisms must act *canonically* on large parts of their domain. If we are working over a homogeneous structure with finite relational signature, then there are only finitely many behaviours of canonical functions of a given arity, and so this technique allows to perform combinatorial analysis when proving classification results. With this approach we can also show that, under further assumptions on $\mathfrak{B}$, many questions about the expressive power of $\mathfrak{B}$ become decidable, such as the question whether a given quantifier-free first-order formula is in $\mathfrak{B}$ equivalent to a primitive positive formula.

In Chapter 12 we use polymorphisms to classify the computational complexity of a large family of constraint satisfaction problems, namely *temporal CSPs*, i.e., CSPs for first-order reducts of $(\mathbb{Q}; <)$ which includes many CSPs in qualitative temporal reasoning and scheduling. Our classification confirms the mentioned infinite-domain tractability conjecture for this family. The class of temporal CSPs is particularly important because it often provides counterexamples for naive generalisations of known facts for finite-domain constraint satisfaction; moreover, the polynomial-time algorithms are particularly interesting in this class. Similar classifications have also been obtained for the countable homogeneous universal poset [241], the model companion of the class of C-relations [64], and for all homogeneous graphs [80].

Finally, in Chapter 13, we present results that show that the CSPs for certain natural classes of infinite structures do *not* have a complexity dichotomy.

Chapter outline. Constraint satisfaction problems can appear in different forms, because there are several ways to formalise CSPs. The differences in formalising CSPs are related to the way that instances are coded and to the way that the problem itself is described. In the next sections we present four formalisations; each of them is attached to different lines of research. In later sections some arguments are more natural from one perspective than from the other, so it will be convenient to have them all discussed here. Table 1 shows the relationships among the four perspectives in tabular form.

1.1. The homomorphism perspective

A *relational signature* τ is a set of relation symbols R each of which has an associated finite *arity* $k \in \mathbb{N}$. A *relational structure* $\mathfrak{A}$ over the signature τ (also called τ-*structure*) consists of a set A (the *domain* or *base set*) together with

Perspective	*Instance*	*Problem description*	*Section*
Homomorphism	Structure	Structure	1.1
Sentence Evaluation	Sentence	Structure	1.2
Satisfiability	Sentence	Sentences	1.3
Existential Second-Order	Structure	Sentence	1.4

Table 1. The four perspectives on the definition of CSPs.

a relation $R^{\mathfrak{A}} \subseteq A^k$ for each relation symbol $R \in \tau$ of arity k. It causes no harm and will be convenient to allow structures whose domain is empty.

A *homomorphism* from a τ-structure $\mathfrak{A}$ with domain A to a τ-structure $\mathfrak{B}$ with domain B is a function $h\colon A \to B$ that *preserves* all relations; that is, if $R \in \tau$ has arity k and $(a_1, \dots, a_k) \in R^{\mathfrak{A}}$, then $(h(a_1), \dots, h(a_k)) \in R^{\mathfrak{B}}$. If a structure $\mathfrak{A}$ has a homomorphism to $\mathfrak{B}$ we also say that $\mathfrak{A}$ is *homomorphic to* $\mathfrak{B}$ or that $\mathfrak{A}$ *maps homomorphically* to $\mathfrak{B}$. An *isomorphism* is a bijective homomorphism h such that the inverse map $h^{-1}\colon B \to A$, which sends $h(x)$ to x, is a homomorphism, too.

In this text, a *constraint satisfaction problem* (*CSP*) is a computational problem that is specified by a single structure with a finite relational signature, called the *template* (or the *constraint language*; the name 'constraint language' is typically used in the context of the second perspective on CSPs that we present in Section 1.2). Such problems are sometimes also called *non-uniform CSPs* because $\mathfrak{B}$ is not part of the input, but fixed.

Definition 1.1.1 (CSP($\mathfrak{B}$)). Let $\mathfrak{B}$ be a (possibly infinite) structure with a finite relational signature τ. Then CSP($\mathfrak{B}$) is the computational problem of deciding whether a given finite τ-structure $\mathfrak{A}$ maps homomorphically to $\mathfrak{B}$.

CSP($\mathfrak{B}$) can be considered to be a class—the class of all finite τ-structures that map homomorphically to $\mathfrak{B}$. A homomorphism from a given τ-structure $\mathfrak{A}$ to $\mathfrak{B}$ is called a *solution* of $\mathfrak{A}$ for CSP($\mathfrak{B}$). It is in general not clear how to represent solutions for CSP($\mathfrak{B}$) on a computer; however, for the definition of the problem CSP($\mathfrak{B}$) we do not need to represent solutions, since we only have to decide the *existence* of solutions. To represent an input structure $\mathfrak{A}$ of CSP($\mathfrak{B}$) we fix a representation of the relation symbols in the signature τ (the precise choice of the representation is irrelevant because of the assumption that τ is *finite*). Thus, CSP($\mathfrak{B}$) is a well-defined computational problem for *any* infinite structure $\mathfrak{B}$ with finite relational signature.

Example 1.1.2 (Digraph acyclicity). Next, consider the problem CSP($\mathbb{Z}; <$). Here, the relation $<$ denotes the strict linear order of the integers $\mathbb{Z}$. An instance $\mathfrak{A}$ of this problem is a finite $\{<\}$-structure, which can be viewed as a directed

graph (also called *digraph*), potentially with loops. It is easy to see that $\mathfrak{A}$ maps homomorphically to $(\mathbb{Z}; <)$ if and only if there is no directed cycle in $\mathfrak{A}$ (loops are considered to be directed cycles, too). It is easy to see and well known that this can be tested in linear time, for example by performing a depth-first search on the digraph $\mathfrak{A}$.

Example 1.1.3 (Betweenness). The so-called *Betweenness Problem* [296] can be modelled as CSP($\mathbb{Z}$; Betw) where Betw is the ternary relation

$$\{(x, y, z) \in \mathbb{Z}^3 \mid (x < y < z) \vee (z < y < x)\}.$$

This problem is one of the NP-complete problems listed in the book of Garey and Johnson [176].

Example 1.1.4 (Cyclic ordering). The *Cyclic-ordering Problem* [174] can be modelled as CSP($\mathbb{Z}$; Cycl) where Cycl is the ternary relation

$$\{(x, y, z) \in \mathbb{Z}^3 \mid (x < y < z) \vee (y < z < x) \vee (z < x < y)\}.$$

This problem is again NP-complete and can be found in [176].

Example 1.1.5 ($\mathfrak{H}$-colouring problems). Let $\mathfrak{H}$ be an (undirected) graph. We view undirected graphs as τ-structures where τ contains a single binary relation symbol E, which denotes a symmetric relation; in our setting, it is natural and useful to allow *loops*, i.e., edges of the form (x, x). Then the $\mathfrak{H}$-*colouring problem* is the computational problem to decide for a given finite graph $\mathfrak{G}$ whether there exists a homomorphism from $\mathfrak{G}$ to $\mathfrak{H}$. For instance, if $\mathfrak{H}$ is the complete graph on three vertices K_3 without loops, then the $\mathfrak{H}$-colouring problem is the famous 3-*colorability problem* (see e.g. [176]). Similarly, for every fixed k, the k-colorability problem can be modelled as CSP(K_k), where K_k is the complete graph with k vertices, also called a k-*clique*.

The next lemma (Lemma 1.1.8) is a useful test to determine whether a computational problem can be formulated as CSP($\mathfrak{B}$) for some relational structure $\mathfrak{B}$.

DEFINITION 1.1.6. An (*induced*) *substructure* of a τ-structure $\mathfrak{A}$ is a τ-structure $\mathfrak{B}$ with $B \subseteq A$ and $R^{\mathfrak{B}} = R^{\mathfrak{A}} \cap B^n$ for each n-ary $R \in \tau$; we also say that $\mathfrak{B}$ is *induced on* B *by* $\mathfrak{A}$, and write $\mathfrak{A}[B]$ for $\mathfrak{B}$.

The *union* of two τ-structures $\mathfrak{A}, \mathfrak{B}$ is the τ-structure $\mathfrak{A} \cup \mathfrak{B}$ with domain $A \cup B$ and the relation $R^{\mathfrak{A}\cup\mathfrak{B}} := R^{\mathfrak{A}} \cup R^{\mathfrak{B}}$ for every $R \in \tau$. The *intersection* $\mathfrak{A} \cap \mathfrak{B}$ of $\mathfrak{A}$ and $\mathfrak{B}$ is defined analogously. A *disjoint union* of $\mathfrak{A}$ and $\mathfrak{B}$ is the union of isomorphic copies of $\mathfrak{A}$ and $\mathfrak{B}$ with disjoint domains. As disjoint unions are unique up to isomorphism, we usually speak of *the* disjoint union of $\mathfrak{A}$ and $\mathfrak{B}$, and denote it by $\mathfrak{A} \uplus \mathfrak{B}$. The disjoint union of a set of τ-structures $\mathcal{C}$ is defined analogously (and the disjoint union of an empty set of structures is the τ-structure with empty domain). A structure is called *connected* if it is not the

disjoint union of two non-empty structures. A *maximal* connected substructure of $\mathfrak{B}$ (i.e., a connected substructure of $\mathfrak{B}$ such that every substructure of $\mathfrak{B}$ with a larger domain is not connected) is called a *connected component* of $\mathfrak{B}$.

Definition 1.1.7. We say that a class $\mathcal{C}$ of relational structures is

- *closed under homomorphisms* if whenever $\mathfrak{A} \in \mathcal{C}$ and $\mathfrak{A}$ maps homomorphically to $\mathfrak{B}$ we have $\mathfrak{B} \in \mathcal{C}$;
- *closed under inverse homomorphisms* if whenever $\mathfrak{B} \in \mathcal{C}$ and $\mathfrak{A}$ maps homomorphically to $\mathfrak{B}$ we have $\mathfrak{A} \in \mathcal{C}$;
- *closed under (finite) disjoint unions* if whenever $\mathfrak{A}, \mathfrak{B} \in \mathcal{C}$ the disjoint union of $\mathfrak{A}$ and $\mathfrak{B}$ is also in $\mathcal{C}$.

Note that a class $\mathcal{C}$ of τ-structures is closed under inverse homomorphisms if and only if its complement in the class of all τ-structures is closed under homomorphisms. When a class is closed under inverse homomorphisms, or closed under homomorphisms, it is in particular closed under isomorphisms.

Let $\mathcal{F}$ be a class of τ-structures. We say that a structure $\mathfrak{A}$ is *$\mathcal{F}$-free* if no $\mathfrak{B} \in \mathcal{F}$ maps homomorphically to $\mathfrak{A}$ (so $\mathcal{F}$ denotes the homomorphically *forbidden* structures). The class of all finite $\mathcal{F}$-free structures we denote by $\mathrm{Forb}^{\mathrm{hom}}(\mathcal{F})$. The following is a simple, but fundamental lemma for CSPs.

Lemma 1.1.8. *Let τ be a finite relational signature and let $\mathcal{C}$ a class of finite τ-structures. Then the following are equivalent.*

(1) $\mathcal{C} = \mathrm{CSP}(\mathfrak{B})$ *for some τ-structure $\mathfrak{B}$.*
(2) $\mathcal{C} = \mathrm{Forb}^{\mathrm{hom}}(\mathcal{F})$ *for a class of finite connected τ-structures $\mathcal{F}$.*
(3) *$\mathcal{C}$ is closed under disjoint unions and inverse homomorphisms.*
(4) $\mathcal{C} = \mathrm{CSP}(\mathfrak{B})$ *for a countable τ-structure $\mathfrak{B}$.*

Proof. It suffices to prove the implications (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4). For the implication from (1) to (2), let $\mathcal{F}$ be the class of all finite connected τ-structures that do not map homomorphically to $\mathfrak{B}$. If a structure $\mathfrak{A}$ maps homomorphically to $\mathfrak{B}$ then no $\mathfrak{C} \in \mathcal{F}$ can map homomorphically to $\mathfrak{A}$ because of the transitivity of the homomorphism relation. Conversely, if $\mathfrak{A}$ does not map homomorphically to $\mathfrak{B}$, then already one of the connected components of $\mathfrak{A}$ does not map homomorphically to $\mathfrak{B}$. This connected component is in $\mathcal{F}$, which shows that a structure from $\mathcal{F}$ maps homomorphically to $\mathfrak{A}$.

(2) $\Rightarrow$ (3). Suppose (2), and let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be two structures from $\mathrm{Forb}^{\mathrm{hom}}(\mathcal{F})$. Since every $\mathfrak{C} \in \mathcal{F}$ is connected, every homomorphism from $\mathfrak{C}$ to $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ must already be a homomorphism to $\mathfrak{A}_1$ or to $\mathfrak{A}_2$, which is impossible. Hence, $\mathrm{Forb}^{\mathrm{hom}}(\mathcal{F})$ is closed under disjoint unions. Closure under inverse homomorphisms follows straightforwardly from the transitivity of the homomorphism relation.

(3) $\Rightarrow$ (4). Suppose that $\mathcal{C}$ is a class of relational structures that is closed under disjoint unions and inverse homomorphisms. Let $\mathcal{C}'$ be a subclass of $\mathcal{C}$ where we select one structure from each isomorphism class of structures

Triangle-Freeness
Instance: An undirected graph $\mathfrak{G}$.
Question: Is $\mathfrak{G}$ triangle-free?

Acyclic-Bipartition
Instance: A digraph $\mathfrak{G}$.
Question: Is there a partition $V = V_1 \uplus V_2$ of the vertices V of $\mathfrak{G}$ such that both $\mathfrak{G}[V_1]$ and $\mathfrak{G}[V_2]$ are acyclic?

No-Mono-Tri
Instance: An undirected graph $\mathfrak{G}$.
Question: Is there a partition $V = V_1 \uplus V_2$ of the vertices V of $\mathfrak{G}$ such that both $\mathfrak{G}[V_1]$ and $\mathfrak{G}[V_2]$ are triangle-free?

TABLE 2. Three computational problems that are closed under disjoint unions and inverse homomorphisms.

in $\mathcal{C}$. Let $\mathfrak{B}$ be the (countable) disjoint union over all structures in $\mathcal{C}'$ (if $\mathcal{C}'$ is empty then $\mathfrak{B}$ is by definition the empty structure[1]). Clearly, every structure in $\mathcal{C}$ maps homomorphically to $\mathfrak{B}$. Now, let $\mathfrak{A}$ be a finite structure with a homomorphism h to $\mathfrak{B}$. By the construction of $\mathfrak{B}$, the set $h(A)$ is contained in the disjoint union $\mathfrak{C}$ of a finite set of structures from $\mathcal{C}'$. Since $\mathcal{C}$ is closed under disjoint unions, $\mathfrak{C}$ is in $\mathcal{C}$. Clearly, $\mathfrak{A}$ maps homomorphically to $\mathfrak{C}$, and because $\mathcal{C}$ is closed under inverse homomorphisms, $\mathfrak{A}$ is in $\mathcal{C}$ as well. ⊣

Example 1.1.9. The computational problems in Table 2 are closed under disjoint unions and inverse homomorphisms. Hence, Lemma 1.1.8 shows that they can be formulated as CSP($\mathfrak{B}$) for some relational structure $\mathfrak{B}$. It is easy to see that none of those three problems can be formulated as CSP($\mathfrak{B}$) for a *finite* structure $\mathfrak{B}$.

We verify this for the problem of Triangle-freeness. For a fixed n, consider the graph that contains vertices $x_1, \ldots, x_n$, and that contains for every pair i, j with $1 \leq i < j \leq n$ two additional vertices $u_{i,j}, v_{i,j}$ and the edges $(x_i, u_{i,j})$, $(u_{i,j}, v_{i,j})$, $(v_{i,j}, x_j)$. The resulting graph is clearly triangle-free. But note that every homomorphism f from this graph to a graph $\mathfrak{H}$ with strictly less than n vertices must identify at least two of the vertices $x_1, \ldots, x_n$. So suppose that $f(x_i) = f(x_j)$. Then $(f(x_i), f(u_{i,j}))$, $(f(u_{i,j}), f(v_{i,j}))$, and $(f(v_{i,j}), f(x_j))$ are edges in $\mathfrak{H}$ because f is a homomorphism. Hence, $\mathfrak{H}$ either contains a triangle or a loop. In both cases, $\mathfrak{H}$ cannot be the template for Triangle-Freeness.

[1] Structures with an empty domain are often forbidden in model theory. Lemma 1.1.8 is one of the places that motivates our decision to allow them in this text.

We have thus ruled out all templates of size $n-1$, which concludes the proof since n was chosen arbitrarily.

The central conjecture for finite-domain constraint satisfaction problems was the dichotomy conjecture, due to Feder and Vardi [170], which states that for every structure $\mathfrak{B}$ with a finite relational signature and a finite domain, CSP($\mathfrak{B}$) is in P or NP-complete. A solution to the dichotomy conjecture has been announced by Bulatov [111] and, independently, by Zhuk [348]. Before, it has been verified for many classes of structures $\mathfrak{B}$, for instance

- for structures $\mathfrak{B}$ with a two-element domain (Schaefer's theorem [319]; see Section 6.2);
- structures over a 3-element domain [108];
- for finite undirected graphs $\mathfrak{B}$ (the theorem of Hell and Nešetřil [195]; see Section 6.8);
- finite digraphs without sources and sinks [25];
- finite structures $\mathfrak{B}$ that contain a unary relation symbol for each subset of the domain of $\mathfrak{B}$, due to [106] (see also [17] and [110]).

There are many equivalent descriptions of the border between polynomial-time tractable and NP-complete finite-domain CSPs; see Sections 3.7, 6.3.5, 6.6, 6.9.

We close this section with an important concept for finite structures $\mathfrak{B}$, the notion of a *core*; generalisations to infinite structures $\mathfrak{B}$ are presented in Section 2.6.2. The motivation for this concept is that for every finite-domain CSP has a template of minimal size which is unique up to isomorphism, and this template has many pleasant properties. To formalise this, we need the following definitions. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ are called *homomorphically equivalent* if there exists a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ and vice versa. An *embedding* of $\mathfrak{A}$ into $\mathfrak{B}$ is an injective map $f : A \to B$ such that $(a_1, \ldots, a_k)$ is in $R^{\mathfrak{A}}$ if and only if $(f(a_1), \ldots, f(a_k))$ is in $R^{\mathfrak{B}}$. An *endomorphism* of a structure $\mathfrak{B}$ is a homomorphism from $\mathfrak{B}$ to $\mathfrak{B}$.

Definition 1.1.10. A structure $\mathfrak{B}$ is a *core* if all its endomorphisms are embeddings[2]. For structures $\mathfrak{A}, \mathfrak{B}$ of the same signature, the structure $\mathfrak{B}$ is called a *core of* $\mathfrak{A}$ if $\mathfrak{B}$ is a core and homomorphically equivalent to $\mathfrak{A}$.

In fact, we speak of *the core* of a finite structure $\mathfrak{A}$, due to the following fact, whose proof is easy and left to the reader.

Proposition 1.1.11. *Let $\mathfrak{A}$ be a finite structure. Then $\mathfrak{A}$ has a core, and all cores of $\mathfrak{A}$ are isomorphic.*

Core structures $\mathfrak{B}$ have many pleasant properties when it comes to studying the computational complexity of CSP($\mathfrak{B}$) (see for instance Proposition 1.2.11

[2]For finite structures $\mathfrak{B}$, injective self-maps must be bijective, and in fact every injective endomorphism of a structure $\mathfrak{B}$ must be an automorphism. For infinite structures, however, this need not be true, and for reasons that become clear in Chapter 4 we choose the present formulation of the definition.