Reasoning Short Cuts in Infinite Domain Constraint Satisfaction: Algorithms and Lower Bounds for Backdoors

Peter Jonsson
Department of Computer and Information Science, Linköping University, Sweden

Victor Lagerkvist
Department of Computer and Information Science, Linköping University, Sweden

Sebastian Ordyniak
Algorithms Group, University of Sheffield, UK

Abstract

A backdoor in a finite-domain CSP instance is a set of variables where each possible instantiation moves the instance into a polynomial-time solvable class. Backdoors have found many applications in artificial intelligence and elsewhere, and the algorithmic problem of finding such backdoors has consequently been intensively studied. Sioutis and Janhunen (KI, 2019) have proposed a generalised backdoor concept suitable for infinite-domain CSP instances over binary constraints. We generalise their concept into a large class of CSPs that allow for higher-arity constraints. We show that this kind of infinite-domain backdoors have many of the positive computational properties that finite-domain backdoors have: the associated computational problems are fixed-parameter tractable whenever the underlying constraint language is finite. On the other hand, we show that infinite languages make the problems considerably harder.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Theory of computation → Complexity theory and logic

Keywords and phrases Constraint Satisfaction Problems, Parameterised Complexity, Backdoors

Digital Object Identifier 10.4230/LIPIcs.CP.2021.32

Funding *Peter Jonsson*: Partially supported by the Swedish Research Council (VR) under grant 2017-04112.
Victor Lagerkvist: Partially supported by the Swedish Research Council (VR) under grant 2019-03690.
Sebastian Ordyniak: Partially supported by the Engineering and Physical Sciences Research Council under grant EP/V00252X/1.

Acknowledgements We thank the anonymous reviewers for several useful comments.

1 Introduction

The *constraint satisfaction problem* (CSP) is the widely studied combinatorial problem of determining whether a set of constraints admits at least one solution. It is common to parameterise this problem by a set of relations (a *constraint language*) which determines the allowed types of constraints, and by choosing different languages one can model different types of problems. Finite-domain languages e.g. makes it possible to formulate Boolean *satisfiability* problems and *coloring* problems while infinite-domain languages are frequently used to e.g. model classical qualitative reasoning problems in artificial intelligence such as *Allen's interval algebra* and the *region-connection calculus* (RCC). Under the lens of classical complexity a substantial amount is known: every finite-domain CSP is either tractable or is NP-complete [5, 34], and for infinite domains there exists a wealth of dichotomy results separating tractable from intractable cases [1].

The vast expressibility of infinite-domain CSPs makes the search for efficient solution methods extremely worthwhile. While worst-case complexity results indicate that many interesting problems should be insurmountably hard to solve, they are nevertheless solved in practice on a regular basis. The discrepancy between theory and practice is often explained by the existence of "hidden structure" in real-world problems [15]. If such a hidden structure exists in CSPs, then it may be exploited and offer a way of constructing improved constraint solvers. To this end, *backdoors* have been proposed as a concrete way of exploiting this structure. A backdoor represents a "short cut" to solving a hard problem instance and may be seen as a measurement for how close a problem instance is to being polynomial-time solvable [23]. The existence of a backdoor then allows one to solve a hard problem by brute forcing solutions to the (hopefully small) backdoor and then solving the resulting problems in polynomial time. This approach has been highly successful: applications can be found in e.g. (quantified) propositional satisfiability [29, 30], abductive reasoning [28], argumentation [8], planning [24], logic [26], and answer set programming [10]. Williams et al. [33] argue that backdoors may explain why SAT solvers occasionally fail to solve randomly generated instances with only a handful of variables but succeed in solving real-world instances containing thousands of variables. This argument appears increasingly relevant since modern SAT solvers frequently handle real-world instances with *millions* of variables. Might it be possible to make similar headway for infinite-domain CSP solvers? For example, can solvers in qualitative reasoning (see, e.g., Section 3.3 in [9]) be analysed in a backdoor setting? Or are the various problems under consideration so different that a general backdoor definition does not make sense?

We attack the problem from a general angle and propose a backdoor notion applicable to virtually all infinite-domain CSPs of practical and theoretical interest. Our departure is a recent paper by Sioutis and Janhunen [32] where backdoors are studied for qualitative constraint networks (which corresponds to CSPs over certain restricted sets of binary relations). We begin in Section 3 by showing why the finite-domain definition of backdoors is inapplicable in the infinite-domain setting and then continue by presenting our alternative definition, based on the idea of defining a backdoor with respect to *relationships* between variables rather than individual variables (which is the basis for the finite-domain definition [13]). We consider CSPs with respect to a fixed set of binary[1] basic relations, e.g. the basic relation in RCC-5, and then consider constraint languages definable by (not necessarily binary) first-order formulas over the basic relations. In this setting we then define a backdoor as a set of tuples of variables so that once the relationship between these variables are fixed, the resulting problem belongs to a given tractable class. If we contrast our approach with that of Sioutis and Janhunen [32], then our method is applicable to CSPs over relations of arbitrarily high arity, and we require only mild, technical assumptions on the set of binary basic relations. Crucially, Sioutis and Janhunen [32] do not consider the computational complexity of any backdoor related problems, and thus do not obtain any algorithmic results.

One of the most important properties of finite-domain backdoors is that they have desirable computational properties. Unsurprisingly, backdoor detection is NP-hard under the viewpoint of classical complexity, even for severely restricted cases. However, the situation changes if we adopt a *parameterized* complexity view. Here, the idea is to approach hard computational problems by characterizing problem parameters that can be expected to be small in applications, and then design algorithms polynomial in input size combined with a super-polynomial dependence on the parameter. We say that a problem is *fixed-*

[1] The generalisation to higher-arity relations is straightforward.

parameter tractable if its running time is bounded by $f(p) \cdot n^{O(1)}$ where n is the instance size, p the parameter, and f is a computable function. The good news is then that the backdoor detection/evaluation problem for finite-domain CSPs with a fixed finite and tractable constraint language, is *fixed-parameter tractable* (fpt) when parameterized by the size p of the backdoor [14], i.e. solvable in $f(p) \cdot n^{O(1)}$ time where n is instance size. However, if the constraint language is not finite, the basic computational problems become W[2]-hard [6]. Thus, if the backdoor size is reasonably small, which we expect for many real-world instances with hidden structure, backdoors can both be found efficiently and be used to simplify the original problem. So-called XP algorithms with a running time bounded by $n^{\text{poly}(p)}$ are polynomial-time when p is fixed, too. However, since p appears in the exponent, they become impractical when large instances are considered, and fpt algorithms are thus considered significantly better. If the constraint language is infinite, then the detection problem is not fixed-parameter tractable in general and the complexity landscape becomes more complex. Note that if the detection problem is not efficiently solvable, then the complexity of the evaluation problem is of minor importance.

While there are profound differences between finite- and infinite-domain CSPs, many important properties of backdoors fortunately remain valid when switching to the world of infinite domains. We construct algorithms (Section 4) for backdoor detection and evaluation showing that these problems are fixed-parameter tractable (with respect to the size of the backdoor) for infinite-domain CSPs based on *finite* constraint languages. Many CSPs studied in practice fulfill this condition and our algorithms are directly applicable to such problems. Algorithms for the corresponding finite-domain problems are based on enumeration of domain values. This is clearly not possible when handling infinite-domain CSPs, so our algorithms enumerate other kinds of objects, which introduces certain technical difficulties. Once we leave the safe confinement of finite languages the situation changes drastically (Section 5). We prove that the backdoor detection problem is W[2]-hard for infinite languages, making it unlikely to be fixed-parameter tractable. Importantly, our W[2]-hardness result is applicable to *all* infinite-domain CSPs where constraints are represented by first-order formulas over a fixed relational structure, meaning that it is not possible to circumvent this difficulty by targeting other classes of problems. Hence, while some cases of hardness are expected, given earlier results for satisfiability and finite-domain CSPs [15], it is perhaps less obvious that essentially all infinite-domain CSPs exhibit the same source of hardness. Again, these negative results are *not* restricted to specific problems and instead show a general difficulty in applying backdoors over infinite constraint languages. We conclude the paper with a discussion concerning future research directions (Section 6).

Throughout, some proofs have been moved to the appendix, and the affected statements are marked with an asterisk ($*$).

2 Preliminaries

2.1 Relations and Formulas

A *relational structure* over a set of values D (a *domain*) is a tuple $(D; R_1, \ldots, R_m)$ where each R_i is a (finitary) relation over D. For simplicity we do not distinguish between the signature of a relational structure and its relations. Assume that $\mathcal{R}$ contains binary relations over the domain D. We say that the relations in $\mathcal{R}$ are *jointly exhaustive* (JE) if $\bigcup \mathcal{R} = D^2$, and that they are *pairwise disjoint* (PD) if $R \cap R' = \varnothing$ for all distinct $R, R' \in \mathcal{R}$. Additionally, a constraint language which is both JE and PD is said to be JEPD.

Let $\varphi(x_1, \ldots, x_n)$ be a first-order formula (with equality) over free variables $x_1, \ldots, x_n$ over a relational structure $\Gamma = (D; R_1, \ldots, R_m)$. We write $\mathrm{Sol}(\varphi(x_1, \ldots, x_n))$ for the set of models of $\varphi(x_1, \ldots, x_n)$ with respect to $x_1, \ldots, x_n$, i.e., $(d_1, \ldots, d_n) \in \mathrm{Sol}(\varphi(x_1, \ldots, x_n))$ if and only if $(D; R_1, \ldots, R_m) \models \varphi(d_1, \ldots, d_n)$, and we use the notation $R(x_1, \ldots, x_n) \equiv \varphi(x_1, \ldots, x_n)$ to define R as $\mathrm{Sol}(\varphi(x_1, \ldots, x_n))$. In this case, we say that R is *first-order definable* (fo-definable) in Γ. In addition to first-order logic, we sometimes use the quantifier-free (qffo), the primitive positive (pp), and the quantifier-free primitive positive (qfpp) fragments. The qffo fragment consists of all formulas without quantifiers, the pp fragment consists of formulas that are built using existential quantifiers, conjunction and equality, and the qfpp consists of quantifier-free pp formulas. We lift the notion of definability to these fragments in the obvious way. It is important to note that if R is pp-definable in Γ, then R is not necessarily qfpp-definable in Γ even if Γ admits quantifier elimination.

If the structure Γ admits quantifier elimination (i.e. every first-order formula has a logically equivalent formula without quantifiers), then fo-definability coincides with qffo-definability. This is sometimes relevant in the sequel since our results are mostly based on qffo-definability. There is a large number of structures admitting quantifier elimination and interesting examples are presented in every standard textbook on model theory, cf. Hodges [17]. Well-known examples include *Allen's interval algebra* (under the standard representation via intervals in $\mathbb{Q}$) and the spatial formalisms RCC-5 and RCC-8 (under the model-theoretically pleasant representation suggested by Bodirsky & Wölfl [4]). Some general quantifier elimination results that are highly relevant for computer science and AI are discussed in Bodirsky [1, Sec. 4.3.1].

2.2 The Constraint Satisfaction Problem

Turning to computability, a *constraint language*, or simply language, over a domain D is a set of relations Γ_D over D. The *constraint satisfaction problem* over a constraint language Γ_D ($\mathrm{CSP}(\Gamma_D)$) is then the computational problem of determining whether a set of constraints over Γ_D admits at least one satisfying assignment.

$\mathrm{CSP}(\Gamma_D)$	
Input:	A tuple (V, C) where V is a set of variables and C a set of constraints of the form $R(x_1, \ldots, x_k)$, where $R \in \Gamma_D$ and $x_1, \ldots, x_k \in V$.
Question:	Does there exists a satisfying assignment to (V, C), i.e., a function $f\colon V \to D$ such that $(f(x_1), \ldots, f(x_k)) \in R$ for each constraint $R(x_1, \ldots, x_K) \in C$?

We write $\mathrm{Sol}(I)$ for the set of all satisfying assignments to a $\mathrm{CSP}(\Gamma)$ instance I. Finite-domain constraints admit a simple representation obtained by explicitly listing all tuples in the involved relation. For infinite domain CSPs, it is frequently assumed that Γ is a *first-order reduct* of an underlying relational structure $\mathcal{R}$, i.e., each $R \in \Gamma$ is fo-definable in $\mathcal{R}$. Whenever $\mathcal{R}$ admits quantifier elimination, then we can always work with the *qffo reduct* where each $R \in \Gamma$ is qffo-definable in $\mathcal{R}$.

▶ **Example 1.** An *equality* language is a first-order reduct of a structure $(D; \emptyset)$ where D is a countably infinite domain. Each literal in a first-order formula over this structure is either of the form $x = y$, or $x \neq y \equiv \neg(x = y)$. The structure $(D; \emptyset)$ admits quantifier elimination so every first-order reduct can be viewed as a qffo reduct. For example, if we let S be defined via the formula $(x = y \wedge x \neq z) \vee (x \neq y \wedge y = z)$ then $\mathrm{CSP}(\{S\})$ is known to be NP-complete [3]. On the other hand, $\mathrm{CSP}(\{=, \neq\})$ is well-known to be tractable.

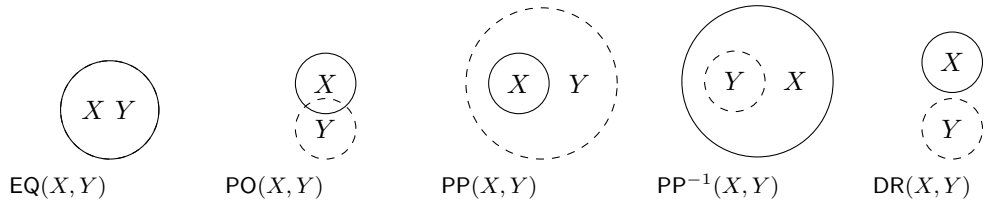

Figure 1 Illustration of the basic relations of RCC-5 with two-dimensional disks.

A *temporal language* is a first-order reduct of $(\mathbb{Q}; <)$. The structure $(\mathbb{Q}; <)$ admits quantifier elimination so it is sufficient to consider qffo reducts. For example, the *betweenness relation* (Betw) can be defined via the formula $(x < y \wedge y < z) \vee (z < y \wedge y < x)$, and the resulting CSP is well-known to be NP-complete, due to Bodirsky & Kára [3].

It will occasionally be useful to assume that the underlying relational structure is JEPD. Clearly, neither $(\mathbb{N}; \emptyset)$, nor $(\mathbb{Q}; <)$ are JEPD, but they can easily be expanded to satisfy the JEPD condition by (1) adding the converse of each relation, and (2) adding the complement of each relation. Thus, an equality language can be defined as a first-order reduct of $(\mathbb{N}; =, \neq)$, and a temporal language as a first-order reduct of $(\mathbb{Q}; =, <, >)$. More ideas for transforming non-JEPD languages into JEPD languages can be found in [2, Sec. 4.2].

Constraint languages in this framework also capture many problems of particular interest in artificial intelligence. For example, consider the *region connection calculus* with the 5 basic relations $\Theta = \{\mathsf{DR}, \mathsf{PO}, \mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{EQ}\}$ (RCC-5). See Figure 1 for a visualisation of these relations. In the traditional formulation of this calculus one then allows unions of the basic relations, which (for two regions X and Y) e.g. allows us to express that X is a proper part of Y or X and Y are equal. This relation can easily be defined via the (quantifier-free) first-order formula $(x\mathsf{PP}y) \vee (x\mathsf{EQ}y)$. Hence, if we let $\Theta^{\vee=}$ be the constraint language consisting of all unions of basic relations in Θ, then $\Theta^{\vee=}$ is a qffo reduct of Θ.

2.3 Parameterized Complexity

To analyse complexity of CSPs we use the framework of *parameterized complexity* [7, 11] where the run-time of an algorithm is studied with respect to a parameter $p \in \mathbb{N}$ and the input size n. Given an instance I of some computational problem, we let $||I||$ denote the bit-size of I. Many important CSPs are NP-hard on general instances and are regarded as being theoretically intractable. However, realistic problem instances are not chosen arbitrarily and they often contain structure that can be exploited for solving the instance efficiently. The idea behind parameterized analysis is that the parameter describes the structure of the instance in a computationally meaningful way. The result is a fine-grained complexity analysis that is more relevant to real-world problems while still admitting a rigorous theoretical treatment including, for instance, algorithmic performance guarantees.

The most favourable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided in $f(p) \cdot n^{O(1)}$ time, where f is a computable function. However, parameterised complexity offers strong theoretical evidence that inclusion in FPT is highly unlikely for some problem, e.g. those that are hard for the class W[1]. The latter contains all problems that admit many-to-one reductions from the Parameterised Clique problem, which asks whether a graph has a clique of size p, where p is the parameter, and the reduction runs in fixed-parameter time with respect to p.

We will prove that certain problems are not in FPT and this requires some machinery. A *parameterized problem* is, formally speaking, a subset of $\Sigma^* \times \mathbb{N}$ where Σ is the input alphabet. Reductions between parameterized problems need to take the parameter into

account. To this end, we will use *parameterized reductions* (or fpt-reductions). Let L_1 and L_2 denote parameterized problems with $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$. A parameterized reduction from L_1 to L_2 is a mapping $P : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ such that (1) $(x, k) \in L_1$ if and only if $P((x, k)) \in L_2$, (2) the mapping can be computed by an fpt-algorithm with respect to the parameter k, and (3) there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in L_1$ if $(x', k') = P((x, k))$, then $k' \leq g(k)$. The class $\mathsf{W}[1]$ contains all problems that are fpt-reducible to Independent Set when parameterized by the size of the solution, i.e. the number of vertices in the independent set. Showing $\mathsf{W}[1]$-hardness (by an fpt-reduction) for a problem rules out the existence of a fixed-parameter algorithm under the standard assumption $\mathsf{FPT} \neq \mathsf{W}[1]$.

3 Backdoors

This section is devoted to the motivation behind and the introduction of a general backdoor concept for CSPs.

3.1 Motivation

We begin by recapitulating the standard definition of backdoors for finite-domain CSPs. Let $\alpha \colon X \to D$ be an assignment. For a k-ary constraint $c = R(x_1, \ldots, x_k)$ we denote by $c_{|\alpha}$ the constraint over the relation R_0 and with scope X_0 obtained from c as follows: R_0 is obtained from R by (1) removing $(d_1, \ldots, d_k)$ from R if there exists $1 \leq i \leq k$ such that $x_i \in X$ and $\alpha(x_i) \neq d_i$, and (2) removing from all remaining tuples all coordinates d_i with $x_i \in X$. The scope X_0 is obtained from $x_1, \ldots, x_k$ by removing every $x_i \in X$. For a set C of constraints we define $C_{|\alpha}$ as $\{c_{|\alpha} \colon c \in C\}$. We now have everything in place to define the standard notion of a (strong) backdoor, in the context of Boolean satisfiability problems and finite-domain CSPs.

▶ **Definition 2** ([13, 33]). *Let $\mathcal{H}$ be a set of CSP instances. A $\mathcal{H}$-*backdoor *for a CSP(Γ_D) instance (V, C) is a set $B \subseteq V$ where $(V \setminus B, C_{|\alpha}) \in \mathcal{H}$ for each $\alpha \colon B \to D$.*

In practice, $\mathcal{H}$ is typically defined as a polynomial-time solvable subclass of CSP and one is thus interested in finding a backdoor into the tractable class $\mathcal{H}$. If the CSP instance I has a backdoor of size k, then it can be solved in $|D|^k \cdot \mathrm{poly}(||I||)$ time. This is an exponential running time with the advantageous feature that it is exponential not in the instance size $||I||$, but in the domain size and backdoor set size only.

▶ **Example 3.** Let us first see why Definition 2 is less impactful for infinite-domain CSPs. Naturally, the most obvious problem is that one, even for a fixed $B \subseteq V$, need to consider infinitely many functions $\alpha \colon V \to D$, and there is thus no general argument which resolves the backdoor evaluation problem. However, even for a fixed assignment $\alpha \colon V \to D$ we may run into severe problems. Consider a single equality constraint of the form $(x = y)$ and an assignment α where $\alpha(x) = 0$ but where α is not defined on y. Then $(x = y)_{|\alpha} = \{(0)\}$, i.e., the constant 0 relation, which is *not* an equality relation. Similarly, consider a constraint XrY where r is a basic relation in RCC-5. Regardless of r, assigning a fixed region to X but not to Y results in a CSP instance which is not included in *any* tractable subclass of RCC-5 (and is not even an RCC-5 instance).

Hence, the usual definition of a backdoor fails to compensate for a fundamental difference between finite and infinite-domain CSPs: that assignments to variables are typically much less important than the *relation* between variables.

3.2 Basic Definitions and Examples

Recall that we in the infinite setting are mainly interested in CSP(Γ) problems where each relation in Γ is qffo-definable over a fixed relational structure $\mathcal{R}$. Hence, in the backdoor setting we obtain three components: a relational structure $\mathcal{R}$ and two qffo reducts $\mathcal{S}$ and $\mathcal{T}$ over $\mathcal{R}$, where CSP($\mathcal{S}$) is the (likely NP-hard) problem which we want to solve by finding a backdoor to the (likely tractable) problem CSP($\mathcal{T}$). Additionally, we will assume that $\mathcal{R}$ only consists of binary JEPD relations and that the equality relation is qffo-definable in $\mathcal{R}$.

▶ **Definition 4.** *Let $\mathcal{R} = (D; R_1, R_2, \dots)$ be a relational structure where the relations are binary and JEPD, the equality relation on D is qffo-definable in $\mathcal{R}$, and let $\mathcal{S}$ and $\mathcal{T}$ be two qffo reducts of $\mathcal{R}$. We say that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a* language triple *and we refer to*

- *$\mathcal{S}$ as the* source language,
- *$\mathcal{T}$ as the* target language, *and*
- *$\mathcal{R}$ as the* base language.

Note that $\mathcal{S}$ and $\mathcal{T}$ may contain non-binary relations even though $\mathcal{R}$ only contains binary relations. One should note that all concepts work equivalently well for higher arity relations in $\mathcal{R}$ but it complicates the presentation. Also note that the equality relation needs to be qffo-definable in $\mathcal{R}$ since this relation is always available in first-order formulas. An alternative way is to require that the equality relation is a member of $\mathcal{R}$ but this assumption is stronger than is needed for our purposes (see Example 8 for an example).

We begin by describing how constraints can be simplified in the presence of a partial assignment of relations from $\mathcal{R}$ to pairs of variables.

▶ **Definition 5.** *Let $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ be a language triple and let (V, C) be an instance of CSP($\mathcal{S}$). Say that a partial mapping $\alpha\colon B \to \mathcal{R}$ for $B \subseteq V^2$ is* consistent *if the CSP($\mathcal{R}$) instance*

$$(V, \{R(x, y) \mid x, y \in V,\ \alpha(x, y) \text{ is defined},\ R = \alpha(x, y)\})$$

is satisfiable. We define a reduced constraint *with respect to a consistent α as:*

$$R(x_1, \dots, x_k)_{|\alpha} = R(x_1, \dots, x_k) \wedge \bigwedge_{\alpha(x_i, x_j) = S, x_i, x_j \in \{x_1, \dots, x_k\}} S(x_i, x_j).$$

Next, we describe how reduced constraints can be translated to the target language. Let $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ be a language triple and let $a \in \mathbb{N} \cup \{\infty\}$ equal $\sup\{i \mid R \in \mathcal{S} \text{ has arity } i\}$. Let

$$\mathbf{S} = \{\mathrm{Sol}(R(x_1, \dots, x_k)_{|\alpha}) \mid R \in \mathcal{S},\ \alpha\colon \{x_1, \dots, x_k\}^2 \to \mathcal{R}\}$$

and

$$\mathbf{T} = \{\varphi(x_1, \dots, x_k) \mid k \leq a, \varphi(x_1, \dots, x_k) \text{ is a qfpp-definition over } \mathcal{T}\}.$$

We interpret these two sets as follows. Each mapping $\alpha\colon \{x_1, \dots, x_k\}^2 \to \mathcal{R}$ applied to a constraint $R(x_1, \dots, x_k)$ results in a (potentially) simplified constraint $R(x_1, \dots, x_k)_{|\alpha}$, which might or might not be expressible via a CSP($\mathcal{T}$) instance. Then the condition that $\mathrm{Sol}(R(x_1, \dots, x_k)_{|\alpha}) \in \mathbf{S}$ is qfpp-definable over $\mathcal{S}$ simply means that the set of models of the constraint $R(x_1, \dots, x_k)_{|\alpha}$ can be defined as the set of models of a CSP($\mathcal{S}$) instance. Thus, the set $\mathbf{S}$ represents all possible simplifications of constraints (with respect to $\mathcal{S}$) and the set $\mathbf{T}$ represents all possibilities of expressing constraints (up to a fixed arity) by the language $\mathcal{T}$. Crucially, note that $\mathbf{S}$ and $\mathbf{T}$ are finite whenever $\mathcal{S}$ and $\mathcal{T}$ are finite. With the help of the two sets $\mathbf{S}$ and $\mathbf{T}$ we then define the following method for translating (simplified) $\mathcal{S}$-constraints into $\mathcal{T}$-constraints.

▶ **Definition 6.** *A* simplification map *is a partial mapping* Σ *from* $\mathbf{S}$ *to* $\mathbf{T}$ *such that for every* $R \in \mathbf{S}$*:* $\Sigma(R) = \varphi(x_1, \dots, x_k)$ *if* $R(x_1, \dots, x_k) \equiv \varphi(x_1, \dots, x_k)$ *for some* $\varphi(x_1, \dots, x_k) \in \mathbf{T}$*, and is undefined otherwise.*

We typically say that a simplification map goes from the source language $\mathcal{S}$ to the target language $\mathcal{T}$ even though it technically speaking is a map from $\mathbf{S}$ to $\mathbf{T}$. Note that if $\mathcal{S}$ and $\mathcal{T}$ are both finite, then there always exists a simplification map of finite size, and one may without loss of generality assume that it is possible to access the map in constant time. We will take a closer look at the computation of simplification maps for finite language in Section 4.1. If $\mathcal{S}$ is infinite, then the situation changes significantly. First of all, a simplification map has an infinite number of inputs and we cannot assume that it is possible to access it in constant time. We need, however, always assume that it can be accessed in polynomial time. Another problem is that we have no general way of computing simplification maps so they need to be constructed on a case-by-case basis.

▶ **Definition 7.** *Let* $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ *be a language triple, and let* Σ *be a simplification map from* $\mathcal{S}$ *to* $\mathcal{T}$*. For an instance* (V, C) *of CSP*$(\mathcal{S})$ *we say that* $B \subseteq V^2$ *is a* backdoor *if, for every consistent* $\alpha \colon B \to \mathcal{R}$*,* $\Sigma(R(x_1, \dots, x_k)_{|\alpha})$ *is defined for every constraint* $R(x_1, \dots, x_k) \in C$*.*

Before turning to computational aspects of finding and using backdoors, let us continue by providing additional examples, starting with finite-domain languages.

▶ **Example 8.** Let $D = \{1, \dots, d\}$ for some $d \in \mathbb{N}$ and define the relational structure $\mathbf{D} = (D; R_{ij} \mid 1 \leq i, j \leq d)$ where $R_{ij} = \{(i, j)\}$. This structure consists of binary JEPD relations and the equality relation on D is qffo-definable in $\mathbf{D}$ via

$$x =_D y \equiv R_{11}(x, y) \vee R_{22}(x, y) \vee \dots \vee R_{dd}(x, y).$$

Note that *any* constraint language Γ with domain D can be viewed as a first-order reduct over $\mathbf{D}$. Hence, a backdoor in the style of Definition 2 is a special case of Definition 7, meaning that our backdoor notion is not merely an adaptation of the finite-domain concept, but a strict generalisation, since we allow arbitrary binary relations (and not only unary relations) in the underlying relational structure.

▶ **Example 9.** Consider equality languages, i.e. languages that are fo-definable over the base structure $(\mathbb{N}; =, \neq)$. Recall the NP-hard ternary relation S from Example 1 and consider a simplification map Σ with respect to the tractable target language $\{=, \neq\}$. Note that we *cannot* simplify an arbitrary constraint $S(x, y, z)$, but that we can simplify $S(x, y, z)_{|\alpha}$ if (e.g.) $\alpha(x, y)$ is '=', or if $\alpha(x, y)$ is $\neq$. Let (V, C) be an instance of CSP$(\{S\})$. Consider the set $B = \{(x, y) \mid S(x, y, z) \in C\} \subseteq V^2$. We claim that B is a backdoor with respect to $\{=, \neq\}$. Let $\alpha \colon B \to \{=, \neq\}$, and consider an arbitrary constraint $S(x, y, z) \in C$. Clearly, $(x, y) \in B$. Then, regardless of the relation between x and y, the constraint can be removed and replaced by $\{=, \neq\}$-constraints.

▶ **Example 10.** Recall the definitions of Θ and $\Theta^{\vee=}$ for RCC-5 from Section 2. Consider a reduced constraint $R_{|\alpha}(x, y)$ with respect to an instance (V, C) of CSP$(\Theta^{\vee=})$, a set $B \subseteq V^2$, and a function $\alpha \colon B \to \Theta$. If $(x, y) \in B$ (or, symmetrically, $(y, x) \in B$) then $R(x, y) \wedge (\alpha(x, y))(x, y)$ is (1) unsatisfiable if $\alpha(x, y) \cap R = \emptyset$, or (2) equivalent to $\alpha(x, y)$. Hence, the simplification map in this case either outputs an unsatisfiable CSP(Θ) instance or replaces the constraints with the equivalent constraint over a basic relation. This results in an $O(5^{|B|}) \cdot \text{poly}(||I||)$ time algorithm for RCC-5, which can slightly be improved to $O(4^{|B|}) \cdot \text{poly}(||I||)$ with the observation that only the trivial relation $(\mathsf{DR} \cup \mathsf{PO} \cup \mathsf{PP} \cup \mathsf{PP}^{-1} \cup \mathsf{EQ})$ contains all the five basic relations.

We now have a working backdoor definition for infinite-domain CSPs, but it remains to show that they actually simplify CSP solving, and that they can be found efficiently. We study such computational aspects in the following section.

4 Algorithms for Finite Languages

This section is divided into three parts where we analyse various computational problems associated with backdoors.

4.1 Computing Simplification Maps

We discussed (in Section 3.2) the fact that a simplification map $\mathcal{S} \to \mathcal{T}$ always exists when $\mathcal{S}$ and $\mathcal{T}$ are finite languages. How to compute such a map is an interesting question in its own right. In the finite-domain case, the computation is straightforward (albeit time-consuming) since one can enumerate all solutions to a CSP instance in finite time. This is clearly not possible when the domain is infinite. Thus, we introduce a method that circumvents this difficulty by enumerating other objects than concrete solutions.

Assume that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a language triple. We first make the following observation concerning relations that are qffo-defined in some binary and JEPD relational structure $\mathcal{R}$ (for additional details, see e.g. Sec. 2.2. in Lagerkvist & Jonsson [19]). If R is qffo-defined in $\mathcal{R}$, then it can be defined by a DNF $\mathcal{R}$-formula that involves only positive (i.e. negation-free) atomic formulas of type $R(\bar{x})$, where R is a relation in $\mathcal{R}$: every atomic formula $\neg R(x, y)$ can be replaced by

$$\bigvee_{S \in \mathcal{R} \setminus \{R\}} S(x, y)$$

and the resulting formula being transformed back to DNF.

Let $I = (V, C)$ denote an arbitrary instance of, for instance, CSP($\mathcal{S}$). An *$\mathcal{R}$-certificate* for I is a satisfiable instance $\mathcal{C} = (V, C')$ of CSP($\mathcal{R}$) that *implies* every constraint in C, i.e. for every $R(v_1, \ldots, v_k)$ in C, there is a clause in the definition of this constraint (as a DNF $\mathcal{R}$-formula) such that all literals in this clause are in C'. It is not difficult to see that I has a solution if and only if I admits an $\mathcal{R}$-certificate (see, for instance, Theorem 6 by Jonsson and Lagerkvist [19] for a similar result). Hence, we will sometimes also say that an $\mathcal{R}$-certificate $\mathcal{C}$ of a CSP($\mathcal{S}$) instance I *satisfies* I. We will additionally use *complete certificates*: a CSP($\cdot$) instance is *complete* if it contains a constraint over every 2-tuple of (not necessarily distinct) variables, and a certificate is complete if it is a complete instance of CSP($\cdot$).

▶ **Example 11.** Consider the structure $(\mathbb{Q}; <, >, =)$, i.e. the rationals under the natural ordering. Let $B = \{(x, y, z) \in \mathbb{Q}^3 \mid x < y < z \vee z < y < x\}$. Let $I = (\{x, y, z, w\} \mid \{B(x, y, z), B(y, z, w)\})$ be an instance of CSP($\{B\}$). The instance I is satisfiable and this is witnessed by the solution $f(x) = 0, f(y) = 1, f(z) = 2, f(w) = 3$. A certificate for this instance is $\{x < y, y < z, z < w\}$ and a complete certificate is

$$\begin{array}{llllll} x < y, & x < z, & x < w, & y < z, & y < w, & z < w, \\ y > x, & z > x, & w > x, & z > y, & w > y, & w > z, \\ x = x, & y = y, & z = z, & w = w. & & \end{array}$$

We first show that satisfiable CSP instances always have complete certificates under fairly general conditions. Furthermore, every solution is covered by at least one such certificate.

▶ **Lemma 12.** (∗) *Assume $\mathcal{R}$ is JEPD and that Γ is qffo-definable in $\mathcal{R}$. An instance (V, C) of CSP(Γ) has a solution $f\colon V \to D$ if and only if there exists a complete $\mathcal{R}$-certificate for I that has solution f.*

We use the previous lemma for proving that two instances I_s and I_t have the same solutions if and only if they admit the same complete certificates.

▶ **Lemma 13.** (∗) *Let $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ be a language triple such that $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{R}$ are finite. Given instances $I_s = (V, C)$ of CSP($\mathcal{S}$) and $I_t = (V, C')$ of CSP($\mathcal{T}$), the following are equivalent:*

1. $\mathrm{Sol}(I_s) = \mathrm{Sol}(I_t)$ *and*
2. *I_s and I_t have the same set of complete $\mathcal{R}$-certificates.*

Finally, we present our method for computing simplification maps.

▶ **Lemma 14.** (∗) *Let X be the set of language triples $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ that enjoy the following properties:*
1. *$\mathcal{S}$, $\mathcal{T}$, and $\mathcal{R}$ are finite and*
2. *CSP($\mathcal{R}$) is decidable.*

The problem of constructing simplification maps for members of X is computable.

4.2 Backdoor Evaluation

We begin by studying the complexity of the following problem, which intuitively, says to which degree the existence of a backdoor helps to solve the original problem.

$[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Evaluation	
Input:	A CSP instance (V, C) of CSP($\mathcal{S}$) and a backdoor $B \subseteq V^2$ into CSP($\mathcal{T}$).
Question:	Is (V, C) satisfiable?

Clearly, $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-backdoor evaluation is in many cases NP-hard: simply pick a language $\mathcal{S}$ such that CSP($\mathcal{S}$) is NP-hard. Note that one, strictly speaking, is not forced to use the backdoor when solving the $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Evaluation problem, but if the size of the backdoor is sufficiently small then we may be able to solve the instance faster via the backdoor. Indeed, as we will now prove, the problem is in FPT for finite languages when parameterised by the size of the backdoor.

▶ **Theorem 15.** *$[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Evaluation is in FPT when parameterised by the size of the backdoor, if $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{R}$ are finite and CSP($\mathcal{T}$) and CSP($\mathcal{R}$) are tractable.*

Proof. Let $\Sigma\colon \mathcal{S} \to \mathcal{T}$ be a simplification map that has been computed off-line, let $I = (V, C)$ be an instance of CSP($\mathcal{S}$), let $B \subseteq V^2$ be a backdoor of size k, and let $m = |\mathcal{R}|$. Then, we claim that I is satisfiable if and only if there is a consistent assignment $\alpha\colon B \to \mathcal{R}$ such that the CSP($\mathcal{T}$) instance $I_{|\alpha} = (V, \{\Sigma(c_{|\alpha}) \mid c \in C\})$ is satisfiable.

Forward direction. Assume that I is satisfiable. Since $\mathcal{R}$ is JEPD, and since $\mathcal{S}$ is qffo-definable in $\mathcal{R}$, we know from Lemma 12 that I admits a complete certificate $(V, \widehat{C})$. For every pair $(x, y) \in B$ then define α to agree with the complete certificate $(V, \widehat{C})$, i.e., $\alpha(x, y) = S$ for $S(x, y) \in \widehat{C}$. Naturally, α is consistent since $(V, \widehat{C})$ is a complete certificate for I, and since B is a backdoor set it also follows that the CSP($\mathcal{T}$) instance $(V, \{\Sigma(c_{|\alpha}) \mid c \in C)$ is

well-defined. Pick an arbitrary constraint $\Sigma(R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha})$. It follows (1) that $(V,\widehat{C})$ satisfies $R(x_1,\ldots,x_{\mathrm{ar}(R)})$, and (2) that if $\alpha(x_i,x_j)=S$ for $x_i,x_j\in\{x_1,\ldots,x_{\mathrm{ar}(R)}\}$ then $(V,\widehat{C})$ satisfies $S(x_i,x_j)$, meaning that $(V,\widehat{C})$ satisfies

$$R(x_1,\ldots,x_{\mathrm{ar}(R)})\wedge \bigwedge_{\alpha(x_i,x_j)=S,x_i,x_j\in\{x_1,\ldots,x_{\mathrm{ar}(R)}\}} S(x_i,x_j),$$

and hence also $\Sigma(R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha})$, since Σ is a simplification map.

Backward direction. Assume that there exists a consistent $\alpha\colon B\to\mathcal{R}$ such that $(V,\{\Sigma(c_{|\alpha})\mid c\in C\})$ is satisfiable, and let $(V,\widehat{C})$ be a complete certificate witnessing this. Naturally, for any pair $(x,y)\in B$ it must then hold that $S(x,y)\in\widehat{C}$ for $\alpha(x,y)=S$, since $(V,\widehat{C})$ could not be a complete certificate otherwise. Pick a constraint $R(x_1,\ldots,x_{\mathrm{ar}(R)})\in C$, and let $\Sigma(R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha})\equiv\varphi(x_1,\ldots x_{\mathrm{ar}(R)})$ for some $\varphi(x_1,\ldots,x_{\mathrm{ar}(R)})\in\mathbf{T}$. It follows that $(V,\widehat{C})$ satisfies

$$\varphi(x_1,\ldots,x_{\mathrm{ar}(R)})$$

and since

$$\mathrm{Sol}(\varphi(x_1,\ldots,x_{\mathrm{ar}(R)}))=\mathrm{Sol}(R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha})$$

it furthermore follows that $R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha}$ must be satisfied, too. However, since

$$R(x_1,\ldots,x_{\mathrm{ar}(R)})_{|\alpha}\equiv R(x_1,\ldots,x_{\mathrm{ar}(R)})\wedge \bigwedge_{\alpha(x_i,x_j)=S,x_i,x_j\in\{x_1,\ldots,x_{\mathrm{ar}(R)}\}} S(x_i,x_j),$$

and since every constraint $S(x_i,x_j)$ is clearly satisfied, it must also be the case that $(V,\widehat{C})$ is a complete certificate of I.

Put together, it thus suffices to enumerate all m^k choices for α and to check whether α is consistent and whether $I_{|\alpha}$ is satisfiable. CSP($\mathcal{R}$) is tractable so checking whether α is consistent can be done in polynomial time. Moreover, using the simplification map Σ, we can reduce $I_{|\alpha}$ to an instance of CSP($\mathcal{T}$), which can be solved in polynomial-time. The total running time is $O(m^k\cdot\mathrm{poly}(||I||))$. ◀

4.3 Backdoor Detection

Theorem 15 implies that small backdoors are desirable since they can be used to solve CSP problems faster. Therefore, let us now turn to the problem of finding backdoors. The basic backdoor detection problem is defined as follows.

$[\mathcal{S},\mathcal{T},\mathcal{R}]$-Backdoor Detection	
Input:	A CSP instance (V,C) of CSP($\mathcal{S}$) and an integer k.
Question:	Does (V,C) have a backdoor B into $\mathcal{T}$ of size at most k? (and if so output such a backdoor)

The problem is easily seen to be NP-hard even when $\mathcal{S}$ and $\mathcal{T}$ are finite; we will provide a proof of this in Corollary 20. We will now prove that the problem can be solved efficiently if the size of the backdoor is sufficiently small.

▶ **Theorem 16.** *$[\mathcal{S},\mathcal{T},\mathcal{R}]$-Backdoor Detection is in FPT when parameterized by k, if $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{R}$ are finite, and CSP($\mathcal{T}$) and CSP($\mathcal{R}$) are tractable.*

Proof. Let $I = ((V, C), k)$ be an instance of $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Detection, let a be the maximum arity of any constraint of I, and let Σ be a simplification map from $\mathcal{S}$ to $\mathcal{T}$ which we assume has been computed off-line. We solve $((V, C), k)$ using a bounded depth search tree algorithm as follows.

We construct a search tree T, for which every node is labeled by a set $B \subseteq V^2$ of size at most k. Additionally, every leaf node has a second label, which is either Yes or No. T is defined inductively as follows. The root of T is labeled by the empty set. Furthermore, if t is a node of T, whose first label is B, then the children of t in T are obtained as follows. If for every consistent assignment $\alpha\colon B \to \mathcal{R}$, where $\mathcal{R} = \{R_1, \ldots, R_m\}$, and every $c \in C$, we have that $\Sigma(c_{|\alpha})$ is defined, then B is a backdoor into $\mathcal{T}$ of size at most k and therefore t becomes a leaf node, whose second label is Yes. Otherwise, i.e., if there is a consistent assignment $\alpha\colon B \to \mathcal{R}$ and a constraint $c \in C$ such that $\Sigma(c_{|\alpha})$ is not defined, we distinguish two cases: (1) $|B| = k$, then t becomes a leaf node, whose second label is No, and (2) $|B| < k$, then for every pair p of variables in the scope of c with $p \notin B$, t has a child whose first label is $B \cup \{p\}$.

If T has a leaf node, whose second label is Yes, then the algorithm returns the first label of that leaf node. Otherwise the algorithm return No. This completes the description of the algorithm.

We now show the correctness of the algorithm. First, suppose the search tree T built by the algorithm has a leaf node t whose second label is Yes. Here, the algorithm returns the first label, say B of t. By definition, we obtain that B is a backdoor into $\mathcal{T}$ of size at most k.

Now consider the case where the algorithm returns No. We need to show that there is no backdoor set B into $\mathcal{T}$ with $|B| \leq k$. Assume, for the sake of contradiction that such a set B exists.

Observe that if T has a leaf node t whose first label is a set B' with $B' \subseteq B$, then the second label of t must be Yes. This is because, either $|B'| < k$ in which case the second label of t must be Yes, or $|B'| = k$ in which case $B' = B$ and by the definition of B it follows that the second label of t must be Yes.

It hence remains to show that T has a leaf node whose first label is a set B' with $B' \subseteq B$. This will complete the proof about the correctness of the algorithm. We will show a slightly stronger statement, namely, that for every natural number ℓ, either T has a leaf whose first label is contained in B or T has an inner node of distance exactly ℓ from the root whose first label is contained in B. We show the latter by induction on ℓ.

The claim obviously holds for $\ell = 0$. So assume that T contains a node t at distance ℓ from the root of T whose first label, say B', is a subset of B. If t is a leaf node of T, then the claim is shown. Otherwise, there is a consistent assignment $\alpha'\colon B' \to \mathcal{R}$ and a constraint $c \in C$ such that $\Sigma(c_{|\alpha'})$ is not defined.

Let $\alpha\colon B \to \mathcal{R}$ be any consistent assignment of the pairs in B that agrees with α' on the pairs in B'. Then, $\Sigma(c_{|\alpha})$ is defined because B is a backdoor set into $\mathcal{T}$. By definition of the search tree T, t has a child t' for every pair p of variables in the scope of some constraint $c \in C$ such that $\Sigma(c_{|\alpha'})$ is not defined. We claim that B contains at least one pair of variables within the scope of c. Indeed, suppose not. Then $\Sigma(c_{|\alpha}) = \Sigma(c_{|\alpha'})$ and this contradicts our assumption that $\Sigma(c_{|\alpha})$ is defined. This concludes our proof concerning the correctness of the algorithm.

The running time of the algorithm is obtained as follows. Let T be a search tree obtained by the algorithm. Then the running time of the depth-bounded search tree algorithm is $O(|V(T)|)$ times the maximum time that is spend on any node of T. Since the number of children of any node of T is bounded by $\binom{a}{2}$ (recall that a is the maximum arity of any

constraint of (V, C)) and the longest path from the root of T to some leaf of T is bounded by $k+1$, we obtain that $|V(T)| \leq \mathcal{O}(((\binom{a}{2}))^{k+1})$. Furthermore, the time required for any node t of T is at most $\mathcal{O}(m^k|C| \cdot \text{poly}(||I||)$ (where the polynomial factors stems from checking whether α is consistent). Therefore we obtain $\mathcal{O}(((\binom{a}{2}))^{k+1}m^k|C|)$ as the total run-time of the algorithm showing that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Detection is FPT when parameterized by k. ◀

5 Hardness Results for Infinite Languages

Our positive FPT results are mainly restricted to finite languages. In Section 5.1, we investigate how the situation differs for infinite languages, and will see that finiteness is not merely a simplifying assumption, but in many cases absolutely crucial for tractability. We remind the reader that if the source language is infinite, then there are an infinite number of possible inputs for the simplificaton map, and this implies that it is not necessarily accessible in polynomial time. However, we will see that simplification maps with good computational properties do exist in certain cases. Even under this assumption, we prove that the backdoor detection problem is in general W[2]-hard. We do not study the backdoor evaluation problem since the hardness of backdoor detection makes the evaluation problem less interesting.

5.1 Hardness of Backdoor Detection

We begin by establishing the existence of a relation which turns out to be useful as a gadget in the forthcoming hardness reduction. For every $k \geq 2$, we let the k-ary equality relation R_k be defined as follows:

$$R_k(x_1, \ldots, x_k) \equiv \bigwedge_{i,j,l,m \in [k] \text{ with } i \neq j \text{ and } l \neq m} (x_i \neq x_j \vee x_l = x_m)$$

▶ **Lemma 17.** (∗) *The following holds for every $k \geq 2$.*

1. *$R_k(x_1, \ldots, x_k)$ cannot be written as a conjunction of binary equality relations $x_i = x_j$ and $x_i \neq x_j$, and*
2. *for every pair i, j with $1 \leq i < j \leq k$ and every assignment α of (x_i, x_j) to $\{(x_i = x_j, x_i \neq x_j)\}$, it holds that $R_k \wedge \alpha((x_i, x_j))$ can be written as a conjunction of binary equality relations.*

Moreover, the definition of R_k can be computed in time k^4.

Let $\mathcal{S}_e = \{R_i \mid i \geq 1\}$ where R_i is defined as in Lemma 17 and let $\mathcal{T}_e = \{=, \neq\}$. Note that both $\mathcal{S}_e$ and $\mathcal{T}_e$ are equality languages so they are qffo reducts of $\mathcal{R}_e = \{=, \neq\}$. We first verify that $\mathcal{S}_e$, despite being infinite, admits a straightforward simplification map to the target language $\mathcal{T}_e = \{=, \neq\}$.

▶ **Lemma 18.** (∗) *There is a simplification map Σ_e from $\mathcal{S}_e$ to $\mathcal{T}_e$ that can be accessed in polynomial time.*

Our reduction is based on the following problem.

Hitting Set	
Input:	A finite set U, a family $\mathcal{F}$ of subsets of U, and an integer $k \geq 0$.
Question:	Is there a set $S \subseteq U$ of size at most k such that $S \cap F \neq \emptyset$ for every $F \in \mathcal{F}$?

Hitting Set is NP-hard even if the sets in $\mathcal{F}$ are restricted to sets of size 2: in this case, the problem is simply the Vertex Cover problem. Furthermore, Hitting Set is W[2]-hard when parameterized by k [7] but this does not hold if the sets in $\mathcal{F}$ have size bounded by some constant.

▶ **Theorem 19.** *$[\mathcal{S}_e, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION is W[2]-hard when parameterised by the size of the backdoor.*

Proof. We give a parameterized reduction from the Hitting Set problem. Given an instance $(U, \mathcal{F}, k)$ of Hitting set, let (V, C) be the CSP($\mathcal{S}_e$) instance with $V = U \cup \{n\}$ having one constraint C_F for every $F \in \mathcal{F}$, whose scope is $F \cup \{n\}$ and whose relation is $R_{|F|+1}$ (as defined in connection with Lemma 17). This can easily be accomplished in polynomial time. Next, we verify that $(U, \mathcal{F}, k)$ has a hitting set of size at most k if and only if (V, C) has a backdoor set of size at most k into CSP($\mathcal{T}_e$).

Forward direction. Let S be a hitting set for $\mathcal{F}$. We claim that $B = \{(n, s) | s \in S\}$ is a backdoor set into CSP($\mathcal{T}_e$). Because S is a hitting set for $\mathcal{F}$, B contains at least two variables from the scope of every constraint in C. Let $\alpha : B \to \{=, \neq\}$ be an arbitrary consistent assignment. Arbitrarily choose a constraint $R_k(x_1, \ldots, x_k)$ in C. By the construction of the simplification map (Lemma 18), it follows that $\Sigma_e(R_k(x_1, \ldots, x_k)_{|\alpha})$ is defined so B is indeed a backdoor.

Backward direction. Let B be a backdoor set for (V, C) into CSP($\mathcal{T}_e$). Note first that we can assume that $b = (x, n)$ or $b = (n, x)$ for every $b \in B$. To see this, note that if this is not the case for some $b \in B$, then we can replace one of the variables in b with n, while still obtaining a backdoor set, since it is sufficient to fix a single relation between pairs of variables in R_k in order to simplify to CSP($\mathcal{T}_e$). We claim that $(\bigcup_{b \in B} b) \setminus \{n\}$ is a hitting set for $\mathcal{F}$. This is clearly the case because for every constraint in C, there must be at least one pair $b \in B$ such that both variables in b are in the scope of the constraint. Otherwise, there would exist a constraint whose simplification is the constraint itself, and such a constraint cannot be expressed as a conjunction of $\{=, \neq\}$ constraints, due to the first condition of Lemma 17. ◀

One may note that CSP($\mathcal{S}_e$) is polynomial-time solvable and that the $[\mathcal{S}_e, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION problem is thus computationally harder than the CSP problem that we attempt to solve with the backdoor approach. This indicates that the backdoor approach must be used with care and it is, in particular, important to know the computational complexity of the CSPs under consideration. Certainly, there are also examples of infinite source languages with an NP-hard CSP such that backdoor detection is W[2]-hard. For instance, let $\mathcal{S}'_e = \mathcal{S} \cup \{S\}$ where S is the relation defined in Example 1 – it follows immediately that CSP($\mathcal{S}'_e$) is NP-hard. Furthermore, it is not hard to verify that Lemma 18 can be extended to the source language $\mathcal{S}'_e$ so the proof of Theorem 19 implies W[2]-hardness of $[\mathcal{S}'_e, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION, too.

Finally, we can now answer the question (that was raised in Section 4.3) concerning the complexity of $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-BACKDOOR DETECTION when $\mathcal{S}$ and $\mathcal{T}$ are finite. By observing that the reduction employed in Theorem 19 is a polynomial-time reduction from Hitting Set and using the fact that Hitting Set is NP-hard even if all sets have size at most 2, we obtain the following result.

▶ **Corollary 20.** *The problem $[\{R_3\}, \mathcal{T}_e, \mathcal{S}_e]$-BACKDOOR DETECTION is NP-hard.*

6 Concluding Remarks

We have generalised the backdoor concept to CSPs over infinite domains and we have presented parameterized complexity results for infinite-domain backdoors. Interestingly, despite being a strict generalisation of finite-domain backdoors, both backdoor detection and evaluation turned out to be in FPT. Hence, the backdoor paradigm is applicable to infinite-domain CSPs, and, importantly, it is indeed possible to have a uniform backdoor definition (rather than having different definitions for equality languages, temporal languages, RCC-5, and so on). Let us now discuss a few different directions for future research.

Backdoor detection and evaluation for infinite languages

Our results show that there is a significant difference between problems based on finite constraint languages and those that are based on infinite languages. The backdoor detection and evaluation problems are fixed-parameter tractable when the languages are finite. In the case of infinite languages, we know that the backdoor detection problem is W[2]-hard for certain choices of languages. This raises the following question: for which infinite source languages is backdoor detection fixed-parameter tractable? This question is probably very hard to answer in its full generality so it needs to be narrowed down in a suitable way. A possible approach is to begin by studying this problem for equality languages.

Broader tractable classes

Recent advances concerning backdoor sets for SAT and finite-domain CSP provide a rather large number of promising and important research directions for future work. For instance, Gaspers et al. [13] have introduced the idea of so-called *hetereogenous* backdoor sets, i.e. backdoor sets into the disjoint union of more than one base language, and Ganian et al. [12] have exploited the idea that if variables in the backdoor set separate the instance into several independent components, then the instance can still be solved efficiently as long as each component is in some tractable base class. Both of these approaches significantly enhance the power and/or generality of the backdoor approach for finite-domain CSP and there is a good chance that these concepts can also be lifted to infinite-domain CSPs. Another promising direction for future research is the use of decision trees (or the even more general concept of *backdoor DNFs*) for representing backdoors [27, 31]. Here the idea is to use decision trees or backdoor DNFs as a compact representation of all (partial) assignments of the variables in the backdoor set. This can lead to a much more efficient algorithm for backdoor evaluation since instead of considering all assignments of the backdoor variables, one only needs to consider a potentially much smaller set of partial assignments of those variables that (1) cover all possible assignments and (2) for each partial assignment the reduced instance is in the base class. It has been shown that this approach may lead to an exponential improvement of the backdoor evaluation problem in certain cases, and it has been verified experimentally that these kinds of backdoors may be substantially smaller than the standard ones [27, 31].

Another direction is to drop the requirement that backdoors move the instance to a polynomial-time solvable class – it may be sufficient that the class is solvable in, say, single-exponential $2^{O(n)}$ time. This can lead to substantial speedups when considering CSPs that are not solvable in $2^{O(n)}$ time. Natural classes of this kind are known to exist under the exponential-time hypothesis [20], and concrete examples are given by certain extensions of Allen's algebra that are not solvable in $2^{o(n \log n)}$ time.

The complexity of simplifying constraints

We have presented an algorithm for constructing simplification maps that works under the condition that the source and target languages are finite. However, we have no general method for computing simplification maps for infinite languages. It seems conceivable that the computation of simplification maps is an undecidable problem and proving this is an interesting research direction. However, could it still make sense to allow suboptimal simplification maps which are oblivious to certain types of constraints, but which can be computed more efficiently? Or simplification maps where not all entries are polynomial time accessible? Thus, the general problem which we want to solve is, given a relation represented by a first-order formula over $\mathcal{R}$ (i.e., corresponding to a simplified constraint) we wish to determine whether it is possible to find a CSP($\mathcal{T}$) instance whose set of models coincides with this relation. This problem is in the literature known as an *inverse constraint satisfaction problem* over a constraint language $\mathcal{T}$ (Inv-CSP($\mathcal{T}$)), and may be defined as follows.

Inv-CSP($\mathcal{T}, \mathcal{R}$)	
Input:	A relation R (represented by an fo-formula over $\mathcal{R}$).
Question:	Can R be defined as the set of models of a CSP($\mathcal{T}$) instance?

We are thus interested in finding polynomial-time solvable cases of this problem, since this would imply the existence of an efficiently computable simplification map to $\mathcal{T}$ even if the source language is infinite. The Inv-CSP problem has been fully classified for the Boolean domain [22, 25], but little is known for arbitrary finite domains, and even less has been established for the infinite case. We suspect that obtaining such a complexity classification is a very hard problem even for restricted language classes such as equality languages. One of the reasons for this is the very liberal way that the input is represented. If one changes the representation, then a complexity classification may be easier to obtain. A plausible way of doing this is to restrict ourselves to ω-*categorical* base structures. The concept of ω-categoricity plays a key role in the study of complexity aspects of CSPs [1], but it is also important from an AI perspective [16, 18, 21]. Examples of such structures include all structures with a finite domain and many relevant infinite-domain structures such as $(\mathbb{N}; =)$, $(\mathbb{Q}; <)$, and the standard structures underlying formalisms such as Allen's algebra and RCC. For ω-categorical base structures $\mathcal{R}$, each fo-definable relation R can be partitioned into a finite number of equivalence classes with respect to the automorphism group of $\mathcal{R}$, and this gives a much more restricted way of representing the input. We leave this as an interesting future research project.

References

1 M. Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction*. Cambridge University Press, 2021. Preprint available from `https://www.math.tu-dresden.de/~bodirsky/Book.pdf`.

2 M. Bodirsky and P. Jonsson. A model-theoretic view on qualitative constraint reasoning. *Journal of Artificial Intelligence Research*, 58:339–385, 2017. `doi:10.1613/jair.5260`.

3 M. Bodirsky and J. Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2):9:1–9:41, 2010.

4 Manuel Bodirsky and Stefan Wölfl. RCC8 is polynomial on networks of bounded treewidth. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2011)*, pages 756–761, 2011.

5 A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proc. 58th Annual Symposium on Foundations of Computer Science (FOCS-2017)*, pages 319–330, 2017.

6 C. Carbonnel, M. C. Cooper, and E. Hebrard. On backdoors to tractable constraint languages. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming*, pages 224–239, Cham, 2014. Springer International Publishing.

7 R. Downey and M. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, 1999. URL: `http://books.google.se/books?id=pt5QAAAAMAAJ`.

8 W. Dvorák, S. Ordyniak, and S. Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186:157–173, 2012.

9 F. Dylla, J. Lee, T. Mossakowski, T. Schneider, A. Van Delden, J. Van De Ven, and D. Wolter. A survey of qualitative spatial and temporal calculi: Algebraic and computational properties. *ACM Computing Surveys*, 50(1):7:1–7:39, 2017.

10 J. Fichte and S. Szeider. Backdoors to tractable answer set programming. *Artificial Intelligence*, 220:64–103, 2015.

11 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

12 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Trans. Algorithms*, 13(2):29:1–29:32, 2017.

13 S. Gaspers, N. Misra, S. Ordyniak, S. Szeider, and S. Živný. Backdoors into heterogeneous classes of SAT and CSP. *Journal of Computer and System Sciences*, 85:38–56, 2017. `doi:10.1016/j.jcss.2016.10.007`.

14 S. Gaspers, S. Ordyniak, and S. Szeider. Backdoor sets for CSP. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 137–157. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

15 S. Gaspers and S. Szeider. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, pages 287–317. Springer, 2012.

16 Robin Hirsch. Relation algebras of intervals. *Artificial intelligence*, 83(2):267–295, 1996.

17 Wilfrid Hodges. *Model theory*. Cambridge University Press, 1993.

18 Jinbo Huang. Compactness and its implications for qualitative spatial and temporal reasoning. In *Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning (KR-2012)*, 2012.

19 P. Jonsson and V. Lagerkvist. An initial study of time complexity in infinite-domain constraint satisfaction. *Artificial Intelligence*, 245:115–133, 2017. `doi:10.1016/j.artint.2017.01.005`.

20 P. Jonsson and V. Lagerkvist. Why are CSPs based on partition schemes computationally hard? In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS-2018)*, pages 43:1–43:15, 2018.

21 Peter Jonsson. Constants and finite unary relations in qualitative constraint reasoning. *Artificial Intelligence*, 257:1–23, 2018.

22 D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28:152–163, 1998.

23 P. Kilby, J. Slaney, S. Thiébaux, and T. Walsh. Backbones and backdoors in satisfiability. In *Proc. 20th National Conference on Artificial Intelligence (AAAI-2005)*, page 1368–1373, 2005.

24 M. Kronegger, S. Ordyniak, and A. Pfandler. Backdoors to planning. *Artificial Intelligence*, 269:49–75, 2019.

25 V. Lagerkvist and B. Roy. Complexity of inverse constraint problems and a dichotomy for the inverse satisfiability problem. *Journal of Computer and System Sciences*, 117:23–39, 2021.

26 A. Meier, S. Ordyniak, M. Ramanujan, and I. Schindler. Backdoors for linear temporal logic. *Algorithmica*, 81(2):476–496, 2019.

27 Sebastian Ordyniak, André Schidler, and Stefan Szeider. Backdoor DNFs. In *Proc. 28th of the International Joint Conference on Artificial Intelligence (IJCAI-2021)*, 2021. To appear. Report version available from `https://www.ac.tuwien.ac.at/files/tr/ac-tr-21-001.pdf`.

28 A. Pfandler, S. Rümmele, and S. Szeider. Backdoors to abduction. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI-2013)*, pages 1046–1052, 2013.

29 M. Samer and S. Szeider. Backdoor sets of quantified boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009.

30 M. Samer and S. Szeider. Fixed-parameter tractability. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 425–454. IOS Press, 2009.

31 Marko Samer and Stefan Szeider. Backdoor trees. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 363–368, 2008.

32 M. Sioutis and T. Janhunen. Towards leveraging backdoors in qualitative constraint networks. In *Proc. 42nd German Conference on AI (KI-2019)*, pages 308–315, 2019.

33 R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 1173–1178, 2003.

34 D. Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM*, 67(5):30:1–30:78, 2020. doi:10.1145/3402029.

A Additional Proofs for Section 4

A.1 Proof of Lemma 12

Proof. Let $I = (V, C)$ be an arbitrary instance of CSP(Γ).

Assume $\mathcal{C} = (V, \widehat{C})$ is a complete $\mathcal{R}$-certificate for I with a solution $f\colon V \to D$. The certificate $\mathcal{C} = (V, \widehat{C})$ implies every constraint in C. Arbitrarily choose a constraint $R(v_1, \ldots, v_k)$ in C. There is a clause in the definition of this constraint (viewed as a DNF $\mathcal{R}$-formula) such that all literals in this clause are in $\widehat{C}$. This implies that $(f(v_1), \ldots, f(v_k)) \in R$ since f is a solution to $\mathcal{C}$. We conclude that f is a solution to I since $R(v_1, \ldots, v_k)$ was chosen arbitrarily.

Assume $f : V \to D$ is a solution to I. We know that $\mathcal{R}$ is JEPD. We construct a complete certificate $\mathcal{C} = (V, \widehat{C})$ such that f is a solution to $\mathcal{C}$. Consider a 2-tuple of (not necessarily distinct) variables (v, v') where $\{v, v'\} \subseteq V$. The tuple $(f(v), f(v'))$ appears in exactly one relation R in $\mathcal{R}$ since $\mathcal{R}$ is JEPD. Add the constraint $R(v, v')$ to $\widehat{C}$. Do the same thing for all 2-tuples of variables. The resulting instance $\mathcal{C}$ is complete and it is satisfiable since f is a valid solution. ◀

A.2 Proof of Lemma 13

Proof. Arbitrarily choose an instance $I_s = (V, C_s)$ of CSP($\mathcal{S}$) and an instance $I_t = (V, C_t)$ of CSP($\mathcal{T}$).

Assume that I_s and I_t have the same set of complete $\mathcal{R}$-certificates. Arbitrarily choose a solution $f : V \to D$ to I_s that is not a solution to I_t (the other direction is analogous). There is a complete $\mathcal{R}$-certificate $\mathcal{C}$ for I_s such that f is a solution to $\mathcal{C}$ by Lemma 12. We know that $\mathcal{C}$ is a certificate for I_t so Lemma 12 implies that f is a solution to I_t, too. This leads to a contradiction.

Assume that I_s and I_t have the same set of solutions. Assume $\mathcal{C}$ is a complete $\mathcal{R}$-certificate for I_s but not for I_t (the other way round is analogous). By Lemma 12, every solution to $\mathcal{C}$ is a solution to I_s. Since I_s and I_t have the same set of solutions, $\mathcal{C}$ is a complete $\mathcal{R}$-certificate for I_t, too, which leads to a contradiction. ◀

A.3 Proof of Lemma 14

Proof. Arbitrarily choose $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ in X. Recall the definitions of $\mathbf{S}$ and $\mathbf{T}$ that were made in connection with Definition 6. Arbitrarily choose a relation $R \in \mathbf{S}$ with arity k and define $I_s = (V, C) = (\{v_1, \ldots, v_k\}, \{R(v_1, \ldots, v_k)\})$. Given a k-ary formula $\varphi \in \mathbf{T}$, let $I_t = (V, \varphi(v_1, \ldots, v_k))$. Then, the following are equivalent

(a) $\mathrm{Sol}(I_s) = \mathrm{Sol}(I_t)$,
(b) I_s and I_t have the same set of complete $\mathcal{R}$-certificates

by Property 1 combined with Lemma 13. Property 2 implies that condition (a) is decidable: there is a straightforward algorithm based on enumerating all complete $\mathcal{R}$-certificates. Compute every possible complete $\mathcal{R}$-certificate on the variables in V and check whether I_s and I_t are equisatisfiable on these certificates. Recall that checking if a certificate implies the constraints in I_s and I_t is a decidable problem since $\mathrm{CSP}(\mathcal{R})$ is decidable. This procedure can be performed in a finite number of steps since the number of complete $\mathcal{R}$-certificates on variable set V is finite.

With this in mind, there is an algorithm that computes a simplification map $\Sigma\colon \mathcal{S} \to \mathcal{T}$. Arbitrarily choose a k-ary relation R in $\mathbf{S}$ and let $I_s = (V, C_s) = (\{v_1, \ldots, v_k\}, \{R(v_1, \ldots, v_k)\})$. Enumerate all $I_t = (V, \varphi(v_1, \ldots, v_k))$ where $\varphi \in \mathbf{T}$ is k-ary. If there exists an I_t that satisfies condition (a), then let $\Sigma(R) = \varphi$, and, otherwise, let $\Sigma(R)$ be undefined. We know that testing condition (a) is decidable by Property 2 and we know that $\mathbf{S}$ and $\mathbf{T}$ are finite sets, so Σ can be computed in a finite number of steps. ◀

B Additional Proofs for Section 5.1

B.1 Proof of Lemma 17

Proof. For proving the first statement, we begin by showing that $(a_1, \ldots, a_k) \in R_k$ if either (1) $a_1 = a_2 = \ldots = a_k$ or (2) $a_i \neq a_j$ for every i and j with $i \neq j$. Assume this is not the case. Then there are $i, j, m, l \in [k]$ with $i \neq j$ and $m \neq l$ such that $a_i = a_j$ and $a_l \neq a_m$. But then the term $(x_i \neq x_j \vee x_l = x_m)$ in the definition of R_k is not satisfied by $(a_1, \ldots, a_k)$. Now, consider a conjunction ϕ of atomic formulas from the set $\{R_i(x_i, x_j) \mid R_i \in \{=, \neq\}, i, j \in [k]\}$. If each atomic formula in ϕ is of the type $x_i = x_j$, then the models of ϕ cannot correctly define R_k: ϕ is not satisfied by any assignment where all variables are assigned distinct values. Similarly, if there exists an atomic formula of the type $x_i \neq x_j$ in ϕ, then ϕ cannot be satisfied by an assignment where all variables are assigned the same value. Hence, R_k cannot be defined as a conjunction of binary equality constraints.

For the second statement, let α be an assignment of a pair (x_i, x_j) to either $x_i \neq x_j$ or $x_i = x_j$. We observe the following.

- if $\alpha(x_i, x_j) = (x_i = x_j)$, then $R_k(x_1, \ldots, x_k) \wedge \alpha(x_i, x_j)$ is logically equivalent to the formula $(x_1 = x_2) \wedge (x_1 = x_3) \wedge \ldots \wedge (x_1 = x_k)$. The definition of R_k contains the clauses $(x_i \neq x_j \vee x_l = x_m)$ for all $1 \leq l \neq m \leq k$. Since $x_i \neq x_j$ does not hold due to $\alpha(x_i, x_j)$, it follows that all variables must be assigned the same value.
- if $\alpha(x_i, x_j) = (x_i \neq x_j)$, then $R_k(x_1, \ldots, x_k) \wedge \alpha(x_i, x_j)$ is logically equivalent to the conjunction of $(x_i \neq x_j)$ for every $i, j \in [k]$ where $i \neq j$. The definition of R_k contains the clauses $(x_i \neq x_j \vee x_l = x_m)$ for all $1 \leq l \neq m \leq k$. Since $x_i = x_j$ does not hold due to $\alpha(x_i, x_j)$, it follows that all variables must be assigned distinct values.

We finally note that the definition of R_k can easily be computed in k^4 time so R_k satisfies the statement of the lemma. ◀

B.2 Proof of Lemma 18

Proof. Consider $\Sigma_e(R_k(x_1, \ldots, x_k)_{|\alpha})$. If $|\{x_1, \ldots, x_k\}| < k$, then we may (without loss of generality) assume that we want to compute $\Sigma_e(R_k(x_1, x_1, x_2 \ldots, x_{k-1})_{|\alpha})$. This is equivalent to computing $\Sigma_e(R_k(x_1, y, x_2, \ldots, x_{k-})_{|\alpha})$ where y is a fresh variable and α is extended to α' so that $\alpha'(x_1, y)$ implies $x_1 = y$. Then, we map $\Sigma(R_k(x_1, y, \ldots, x_k)_{|\alpha'})$ to a suitable

CSP($\mathcal{T}_e$) instance as prescribed by Lemma 17. Assume instead that $|\{x_1, \ldots, x_k\}| = k$. We let $\Sigma_e(R_k(x_1, \ldots, x_k)_{|\alpha})$ be undefined if $\alpha(x_i, x_j)$ is not defined for any distinct $x_i, x_j \in \{x_1, \ldots, x_k\}$ – this is justified by Lemma 17. Otherwise, we map $\Sigma(R_k(x_1, \ldots, x_k)_{|\alpha})$ to a suitable CSP($\mathcal{T}_e$) instance as prescribed by Lemma 17. We conclude the proof by noting that these computations are easy to perform in polynomial time so Σ_e is trivially polynomial-time accessible. ◀