COMPUTATIONAL SHORT CUTS IN INFINITE DOMAIN CONSTRAINT SATISFACTION

PETER JONSSON, VICTOR LAGERKVIST, AND SEBASTIAN ORDYNIAK

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 (Proc. 42nd German Conference on AI (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: the general backdoor detection problem is W[2]-hard and fixed-parameter tractability is ruled out under standard complexity-theoretic assumptions. We demonstrate that backdoors may have suboptimal behaviour on binary constraints—this is detrimental from an AI perspective where binary constraints are predominant in, for instance, spatiotemporal applications. In response to this, we introduce *sidedoors* as an alternative to backdoors. The fundamental computational problems for sidedoors remain fixed-parameter tractable for finite constraint language (possibly also containing non-binary relations). Moreover, the sidedoor approach has appealing computational properties that sometimes leads to faster algorithms than the backdoor approach.

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 model classical qualitative reasoning problems 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 [8, 54], and for infinite domains there exists a wealth of dichotomy results separating tractable from intractable cases [3].

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 [21]. If such a hidden structure exists, 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 [35]. The existence of a backdoor then allows one to solve a hard problem by brute-force enumeration of assignments 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 [48, 49], abductive reasoning [45], argumentation [12], planning [36], logic [40], and answer set programming [15]. Williams et al. (2003) 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., the survey [13]) be analysed in a backdoor setting? Or are the various problems under consideration so different that a general backdoor definition does not make sense?
Backdoors for infinite-domain CSPs. We attack the problem from a general angle and propose a backdoor notion applicable to a large class of infinite-domain CSPs with pronounced practical and theoretical interest. Our departure is a recent paper [50] where backdoors are studied for qualitative constraint networks (which corresponds to CSPs over certain restricted sets of binary relations). We begin by demonstrating 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 [19]). We consider CSPs with respect to a fixed set of binary[1] basic relations $\mathcal{R}$, that we refer to as the *base language*. The base language may, for instance, be some simple mathematical structure like $(\mathbb{Q}; <)$ or a more complex structure such as the basic relations in Allen's algebra or RCC-5. We then consider constraint languages that are definable by first-order formulas over $\mathcal{R}$. For example, *temporal constraints* are based on relations that are first-order definable over $(\mathbb{Q}; <)$. The binary relations that are first-order definable in this structure form the *point algebra* [51] and the corresponding CSP is tractable. However, CSPs based on temporal constraints contain relevant problems that are not expressible via binary relations; examples include AND/OR precedence constraints in scheduling [41], the ORD-Horn class in temporal reasoning [43], and ordering CSPs [22, 23, 24] which subsumes some classical NP-complete problems such as BETWEENNESS and CYCLIC ORDERING [18, Problem MS1 and MS2].

Let us now consider two constraint languages $\mathcal{S}$ (the *source* language) and $\mathcal{T}$ (the *target* language) definable by first-order formulas over $\mathcal{R}$. In this setting we then define a backdoor as a set of pairs of variables so that once the relationship between these variables are fixed, a given CSP($\mathcal{S}$) instance is transformed into a CSP($\mathcal{T}$): this is advantageous if CSP($\mathcal{T}$) is tractable and the backdoor is reasonably small. The connection between CSP($\mathcal{S}$) and CSP($\mathcal{T}$) is described via a *simplification map* and we show that such maps can be automatically computed whenever $\mathcal{S}$ and $\mathcal{T}$ are finite languages. The backdoor approach to a problem now consists of two steps: first a backdoor is computed (the *backdoor detection* problem), and then the backdoor is used to solve the given CSP instance (the *backdoor evaluation* problem). If we contrast our approach with that of Sioutis and Janhunen [50], 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 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

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

algorithms that are polynomial in the input size combined with a super-polynomial dependence on the parameter. We say that a problem is *fixed-parameter tractable* (fpt) 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 fpt when parameterized by the size of the backdoor [20]. 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. However, if the constraint language is not finite, then the basic computational problems may become W[2]-hard [10], and this rules out fpt algorithms under widely accepted complexity-theoretic assumptions.

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 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. 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* base languages, meaning that it is not possible to circumvent this difficulty by simply targeting some other base language. Hence, while some cases of hardness are expected, given earlier results for satisfiability and finite-domain CSPs [21], it is perhaps less obvious that essentially all infinite-domain CSPs exhibit the same source of hardness.

Sidedoors. AI research on infinite-domain CSPs has historically focused on binary relations: it is, for example, well-known that binary relations can capture many relevant facets of space and time, and thus constitute a suitable basis for spatiotemporal reasoning. This represents a possible problem for the backdoor approach outlined in the previous section since the possibilities for substantially faster algorithms appear to be quite limited. Recall that a backdoor is defined as a set of pairs of variables so that once the relationship between these variables are fixed, the resulting problem belongs to a given tractable class. If the instance we start with only contains binary relations, then the backdoor approach merely turns into a straightforward method based on replacing binary constraints with other binary constraints. This may sound like a highly primitive approach but it is still valuable: it is reassuring to see that it leads to substantial speed-ups in certain cases (see, for instance, Example 4). However, this observation gives us the impression that one can do better. To this end, we suggest a new method: the *sidedoor* approach.

The idea behind the sidedoor approach is inspired by the behaviour of the backdoor approach on binary constraints. Let I be an CSP instance over a language $\mathcal{S}$. Assume that $R(x, y)$ is a binary constraint in I where relation R is the disjunction of two relations R_1 and R_2. Then we can do the following simple branching: remove the constraint $R(x, y)$ from I and replace it with $R_1(x, y)$ in the first branch and with $R_2(x, y)$ in the second, and thereafter solve the resulting instances recursively. If the branches of this process ends in instances over some language $\mathcal{T}$, then we can solve I with the aid of a solver for CSP($\mathcal{T}$).

We generalise this idea by branching on subinstances containing a certain number of variables (which we call the *radius*) and replacing the constraints within the subinstances with other constraints in a way that preserves the solutions. This generalisation allows the algorithm to cover several constraints in each branch. We see, for instance, that the number of variable pairs that are covered by a sidedoor increases quadratically with its radius. Hence, the number of binary constraints that can be covered at once increases quite rapidly with increasing radius. If this procedure leads to instances within a tractable class and the number of subinstances that we need to replace is sufficiently small, then it appears to be a viable approach. One should also note that this approach is not geared towards infinite domains and it works for general CSPs regardless of domain size.

Just like the backdoor approach, we have two fundamental computational problems: *sidedoor detection* and *sidedoor evaluation.* We prove that the detection and evaluation problems for sidedoors are fixed-parameter tractable (when parameterized by the size of the sidedoor) under mild additional assumptions. The detection problem can be solved in $O((rs)^{rs})$ time where s is sidedoor size and r is the chosen radius—thus, the detection problem is fpt even for infinite constraint languages. However, there is an important caveat: the radius must satisfy $r \geq a$ where a is the maximum arity of the relations in the given instance. Thus, even if the detection problem is formally fpt for fixed source and target languages together with a fixed radius, the method is restricted to a subset of the possible instances.

The main obstacle for applying the sidedoor approach is to identify and describe suitable branchings on subinstances: we do this via *branching maps.* Unlike simplification maps for backdoors, the computation of branching maps appears to be a difficult problem. Another important difference is that the exact choice of branching map strongly affects the complexity of sidedoor evaluation; this is not the case for the backdoor evaluation problem. We present an algorithmic method to compute branching maps for languages that satisfy a particular definability condition. We stress that this method does not produce branching maps with optimal behaviour and that more research is needed into the construction of branching maps.

We conclude with a remark: the sidedoor approach exploits another kind of hidden structure than the backdoor approach. While backdoors focus on the tuples within the relations, sidedoors focus on how constraints are connected to each other. Thus, backdoors and sidedoors may be viewed as complementary methods based on the kind of structure they exploit. This distinction resembles the two main paths that have been followed in the search for tractable CSP fragments [9]: either one studies the CSP with a restricted set of allowed relations, or one studies the CSP where restrictions are imposed on which variables may be mutually constrained.

Outline. The article has the following structure. Section 2 presents some technical background concerning constraint satisfaction and parameterised complexity. We present the generalisation of backdoors to infinite domains in Section 3 and we introduce sidedoors in Section 4. Section 5 concludes the article with a summary and a comprehensive discussion concerning future research directions. This article is an extended version of a conference paper [32] and the main extension is the addition of the sidedoor concept. The reader should be aware of the fact that we have changed some of the underlying assumptions and the terminology in this article compared to the conference paper. For instance, we now assume that the base structure in the backdoor approach is a partition scheme (in the sense of Ligozat and Renz (2004)) in order to streamline the presentation. Concerning the terminology, we have for example changed *language triple* into *backdoor triple* to emphasise the connection with backdoors.

2. Preliminaries

This section presents the basic definitions that we need in this article.

2.1. **Relations and Formulas.**

A *relational signature* τ is a set of relation symbols R_i where each R_i is associated with an *arity* $k \in \mathbb{N}$. A *relational structure* Γ over the relational signature τ is a set D_Γ (the *domain*) together with a relation $R_i^\Gamma \subseteq D_\Gamma^{k_i}$ for each relation symbol R_i in τ of arity k_i. For simplicity, we often (1) do not distinguish between the signature of a relational structure and its relations, and (2) view a relational structure as a set of relations.

Assume that the relational structure $\mathcal{R}$ is a set of 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 relational structure which is both JE and PD is said to be JEPD. We say that $\mathcal{R}$ is a *partition scheme* [39] if it is JEPD, it contains the equality relation $\{(d,d) \mid d \in D\}$, and for every $R \in \mathcal{R}$, the language $\mathcal{R}$ contains the converse relation $R^{\smile} = \{(d', d) \mid (d, d') \in R\}$.

We continue by recalling some terminology from logic. First-order formulas over a signature τ are inductively defined using the logical symbols of universal and existential quantification, disjunction, conjunction, negation, equality, variable symbols and the relation symbols in τ. A τ-formula without free variables is called a *τ-sentence.* We write $\Gamma \models \phi$ if the relational structure Γ (with signature τ) is a model for the τ-sentence ϕ.

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 write

$$R(x_1, \ldots, x_n) \equiv \varphi(x_1, \ldots, x_n)$$

to indicate that the relation R equals $\mathrm{Sol}(\varphi(x_1, \ldots, x_n))$. In this case, we say that R is *first-order definable* (fo-definable) in Γ. Note that our definitions are always parameter-free, i.e. we do not allow the use of domain elements in them. 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.

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. However, 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. There is a large number of structures admitting quantifier elimination and interesting examples are presented in every standard textbook on model theory, cf. Hodges (1993). 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 (2011)). Some general quantifier elimination results that are highly relevant for computer science and AI are discussed in Bodirsky [3, Sec. 4.3.1].

2.2. The Constraint Satisfaction Problem.

2.2. The Constraint Satisfaction Problem. Let Γ_D denote a relational structure with domain D. The *constraint satisfaction problem* over Γ_D (CSP(Γ_D)) is the computational problem of determining whether a set of constraints over Γ_D admits at least one satisfying assignment.

CSP(Γ_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 $R(x_1, \ldots, x_K) \in C$?

The relational structure Γ_D is often called a *constraint language.* We slightly abuse notation and write $\mathrm{Sol}(I)$ for the set of all satisfying assignments to a CSP(Γ) 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 relation* $\delta \subseteq D^3$ *be defined via the formula* $(x = y \wedge x \neq z) \vee (x \neq y \wedge y = z)$, *then CSP*($\{\delta\}$) *is known to be* NP*-complete* [5]. *On the other hand, CSP*($\{=, \neq\}$) *is well-known to be tractable.*

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 $\beta = \{(x, y, z) \in \mathbb{Q}^3 \mid \min(x, z) < y < \max(x, z)\}$ *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.*

It will often be useful to assume that the underlying relational structure is JEPD or that it is a partition scheme. 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 the partition scheme* $(\mathbb{N}; =, \neq)$, *and a temporal language as a first-order reduct of the partition scheme* $(\mathbb{Q}; =, <, >)$. *More ideas for transforming non-JEPD languages into JEPD languages and partition schemes can be found in Bodirsky & Jonsson* [4, Sec. 4.2].

Constraint languages in this framework 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. These five relations obviously form a partition scheme. 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 the complexity of CSPs, we use the framework of *parameterized complexity* [11, 16] 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 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,

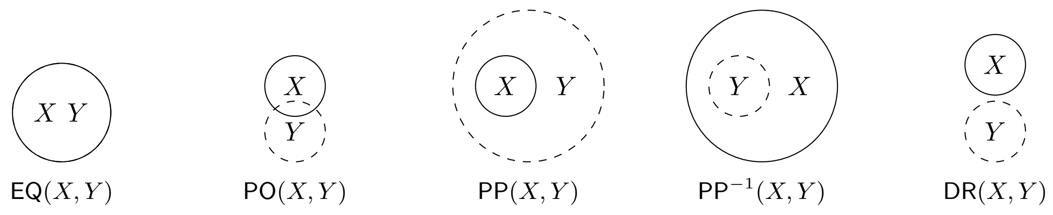

FIGURE 1. Illustration of the basic relations of RCC-5 with two-dimensional disks.

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.

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 $\mathsf{FPT} \neq \mathsf{W}[1]$. $\mathsf{W}[1]$ is a complexity class in the *weft hierarchy* $\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots$. We note that proving $\mathsf{W}[k]$-hardness, $k \geq 1$, for a problem implies $\mathsf{W}[1]$-hardness and the nonexistence of fixed-parameter algorithms.

3. Backdoors

This section is devoted to the introduction of a general backdoor concept for CSPs. The underlying motivation is discussed in Section 3.1 and the formal definition of backdoors is presented in 3.2. Section 3.3 contains algorithms for various backdoor problems over finite constraint languages. Finally, certain hardness results are proved in Section 3.4 for backdoor problems over infinite constraint languages.

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 1 (See, for instance, [19] or [52]). *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 2. *Let us first see why Definition 1 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-domain 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 $\mathcal{R}$ is a partition scheme.

Definition 2. *Let $\mathcal{R} = (D; R_1, R_2, \dots)$ be a partition scheme and let $\mathcal{S}, \mathcal{T}$ be qffo reducts of $\mathcal{R}$. We say that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a* backdoor 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. Generalisations of partition schemes to higher-arity relations is, for instance, described and discussed by Dylla et al. [13, Section 3.1].

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

Definition 3. *Let $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ be a backdoor 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 backdoor 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 4. *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 3.3.1. 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 a priori have no general way of computing simplification maps so they need to be constructed on a case-by-case basis; we return to this problem in Section 3.3.1.

Definition 5. *Let $[\mathcal{S},\mathcal{T},\mathcal{R}]$ be a backdoor 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 $R(x_1,\dots,x_k) \in C$.*

Let us consider a few examples before turning to computational aspects of finding and using backdoors. We use equality languages as a first illustration of backdoors into infinite-domain CSPs.

Example 3. *Consider equality languages, i.e. languages that are fo-definable over the base structure $(\mathbb{N}; =, \neq)$. Recall the ternary relation δ 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 $\delta(x,y,z)$, but that we can simplify $\delta(x,y,z)_{|\alpha}$ if (e.g.) $\alpha(x,y)$ is '$=$', or if $\alpha(x,y)$ is $\neq$. Hence, it is important to realise that we cannot simplify all constraints, but if we end up in a situation where the reduced instance cannot be simplified into $\{=,\neq\}$, then this simply means that the underlying set $B \subseteq V^2$ was not chosen correctly. Let (V,C) be an instance of the* NP*-hard problem CSP($\{\delta\}$). Consider the set $B = \{(x,y) \mid \delta(x,y,z) \in C\} \subseteq V^2$. We claim that B is a backdoor with respect to $\{=,\neq\}$. Let $\alpha\colon B \to \{=,\neq\}$, and consider an arbitrary constraint $\delta(x,y,z) \in C$. Clearly, $(x,y) \in B$. Then, regardless of the relation between x and y, the constraint can be removed and replaced by $\{=,\neq\}$-constraints.*

The next example is particularly important and it will be revisited in Section 4.1 and in Examples 9 and 11.

Example 4. *Recall the definitions of Θ and $\Theta^{\vee=}$ for RCC-5 from Section 2. The problem CSP(Θ) is known to be polynomial-time solvable while CSP($\Theta^{\vee=}$) is* NP*-complete* [46]*. 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. The currently fastest known algorithm for CSP*$(\Theta^{\vee=})$ *runs in* $2^{O(n \log n)}$ *time where* n *is the number of variables* [33]. *Hence, backdoors provide an appealing approach for instances having small backdoors: the backdoor approach is superior for instances with backdoor size in* $o(n \log n)$.

The languages considered in Example 3 and Example 4 are special cases of so-called *homogeneous* and *finitely bounded* languages, where a complexity dichotomy has been conjectured (see, e.g., [2]). We refrain from stating this conjecture and the aforementioned properties formally, but remark that such languages form natural examples of languages triples where our backdoor approach is applicable. In fact, the following holds:

Example 5. *If* $\mathcal{R}$ *is a finitely bounded homogeneous structure, then* $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ *is a backdoor triple for* any *first-order reducts* $\mathcal{S}, \mathcal{T}$ *of* $\mathcal{R}$.

Last, we illustrate that the finite domain case can be seen a special case of our backdoor notion.

Example 6. *Let* $D = \{1, \ldots, d\}$ *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 clearly consists of binary JEPD relations but is technically not a partition scheme since it does not contain the equality relation over* D. *We can, however, easily qffo-define equality in* $\mathbf{D}$ *via*

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

Note then that any *constraint language* Γ *with domain* D *can be viewed as a qffo reduct over* $\mathbf{D}$. *Let* $t = (t_1, \ldots, t_k) \in D^k$. *The relation* R_t *that only contain the tuple* t *can be defined as*

$$R_t(x_1, \ldots, x_k) \equiv \bigwedge_{i=1}^{k} R_{t_i t_i}(x_i, x_i).$$

Any relation $R \subseteq D^k$ *can now be defined as*

$$R(x_1, \ldots, x_k) \equiv \bigvee_{t \in R} R_t(x_1, \ldots, x_k).$$

Hence, a backdoor in the style of Definition 1 is a special case of Definition 5, 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. Hence, we only care whether each constraint, locally, can be simplified to the target language.

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.

3.3. Algorithms for Finite Languages.

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

3.3.1. *Computing Simplification Maps.*

We discussed (in Section 3.2) the fact that a simplification map from $\mathcal{S}$ 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 backdoor triple. We first make the following observation concerning relations that are qffo-defined in a partition scheme $\mathcal{R}$ (for additional details, see e.g. Sec. 2.2. in Lagerkvist & Jonsson (2017)). If R is qffo-defined in $\mathcal{R}$ and $\mathcal{R}$ is finite, then R 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 (2017) 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 7. *Consider the partition scheme* $(\mathbb{Q}; <, >, =)$ *based on the rationals under the natural ordering. Let us once again consider the betweenness relation* $\beta = \{(x, y, z) \in \mathbb{Q}^3 \mid x < y < z \vee z < y < x\}$ *from Example 1. Assume that*

$$I = (\{x, y, z, w\} \mid \{\beta(x, y, z), \beta(y, z, w)\})$$

is an instance of CSP($\{\beta\}$)*. 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 1. *Assume* $\mathcal{R}$ *is a partition scheme 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*.*

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 construct a complete $\mathcal{R}$-certificate $\mathcal{C} = (V, \widehat{C})$ for I such that f is a solution to $\mathcal{C}$. 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. □

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 2. *Let $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ be a backdoor 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.*

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 1. We know that $\mathcal{C}$ is a certificate for I_t so Lemma 1 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 1, 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. □

Finally, we present our method to compute simplification maps.

Lemma 3. *Let X be the set of backdoor 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.

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 4. 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 2. 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 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. □

3.3.2. *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 1. *Assume that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a backdoor triple such that $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{R}$ are finite and CSP($\mathcal{T}$) and CSP($\mathcal{R}$) are polynomial-time solvable. Then, $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Evaluation is solvable in $\mathcal{O}(|\mathcal{R}|^k \cdot \mathrm{poly}(||I||))$ where k is the size of the backdoor. Hence, the problem is in FPT when parameterised by k.*

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 1 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 (which is constant-time accessible since $\mathcal{S}$ and $\mathcal{T}$ are finite), we can reduce $I_{|\alpha}$ to an instance of CSP($\mathcal{T}$), which can be solved in polynomial-time. The total running time is $O(m^k \cdot \text{poly}(||I|||)$. □

We remark that the assumption that CSP($\mathcal{R}$) is in P is only used to verify whether α is consistent, and this can sometimes be achieved even when CSP($\mathcal{R}$) is NP-hard; one such case is when $\mathcal{R}$ is finitely bounded and homogeneous (see Example 5).

3.3.3. *Backdoor Detection.* Theorem 1 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 2. We will now prove that the problem can be solved efficiently if the size of the backdoor is sufficiently small.

Theorem 2. *Assume that* $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ *is a backdoor triple such that* $\mathcal{S}$, $\mathcal{T}$, $\mathcal{R}$ *are finite and CSP*($\mathcal{T}$) *and CSP*($\mathcal{R}$) *are polynomial-time solvable. Then,* $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-Backdoor Detection *is solvable in* $\mathcal{O}(\binom{a}{2}^{k+1} \cdot |\mathcal{R}|^k \cdot \text{poly}(||I||))$ *time where* a *is the maximum arity of the relations in* $\mathcal{S}$ *and* k *is the size of the backdoor. Hence, the problem is in* FPT *when parameterised by* k.

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 relation in $\mathcal{S}$, 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 returns 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 spent 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 the relations in of $\mathcal{S}$) 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—keep in mind that Σ is constant-time accessible since $\mathcal{S}$ and $\mathcal{T}$ are finite). 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. ☐

Again, we remark that the assumption that CSP($\mathcal{R}$) is in P is only used to verify whether α is consistent (see the comment after Theorem 1).

3.4. **Hardness Results for Infinite Languages.** We will now see that the positive fpt results from the two previous sections are, unfortunately, restricted to finite languages: 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. We note that analysing CSPs with infinite constraint languages is often problematic since the way in which the input is encoded may influence the complexity. We will therefore consider concrete, infinite languages in the sequel, and will for all involved relations explicitly state their defining formulas, making it easy to represent the associated computational problems.

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:

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

We illustrate this construction with $R_3(x_1, x_2, x_3)$:

$$(x_1 \neq x_2 \vee x_1 = x_2) \wedge (x_1 \neq x_2 \vee x_1 = x_3) \wedge (x_1 \neq x_2 \vee x_2 = x_3) \wedge$$
$$(x_1 \neq x_3 \vee x_1 = x_2) \wedge (x_1 \neq x_3 \vee x_1 = x_3) \wedge (x_1 \neq x_3 \vee x_2 = x_3) \wedge$$
$$(x_2 \neq x_3 \vee x_1 = x_2) \wedge (x_2 \neq x_3 \vee x_1 = x_3) \wedge (x_2 \neq x_3 \vee x_2 = x_3)$$

or, slightly simplifed,

$$(x_1 \neq x_2 \vee x_1 = x_3) \wedge (x_1 \neq x_2 \vee x_2 = x_3) \wedge$$
$$(x_1 \neq x_3 \vee x_1 = x_2) \wedge (x_1 \neq x_3 \vee x_2 = x_3) \wedge$$
$$(x_2 \neq x_3 \vee x_1 = x_2) \wedge (x_2 \neq x_3 \vee x_1 = x_3).$$

Now, note that $R_3(x_1, x_2, x_3) \wedge (x_1 = x_2)$ is equivalent to

$$(x_1 = x_3) \wedge (x_2 = x_3)$$

and $R_3(x_1, x_2, x_3) \wedge (x_1 \neq x_2)$ is equivalent to

$$(x_1 \neq x_2) \wedge (x_1 \neq x_3) \wedge (x_2 \neq x_3).$$

The important properties of R_k are summarised in the next lemma.

Lemma 4. *Let $\mathcal{R}$ be a partition scheme with infinite domain D. Then the following holds for every $k \geq 2$.*

1. *$R_k(x_1, \ldots, x_k)$ cannot be qfpp-defined over $\mathcal{R} \cup \{\neq\}$, and*
2. *for every pair i, j with $1 \leq i < j \leq k$ and every assignment α of (x_i, x_j) to $\mathcal{R}$, it holds that $R'_k(x_1, \ldots, x_k) \equiv R_k \wedge \alpha((x_i, x_j))(x_i, x_j)$ can be qfpp-defined over $\mathcal{R} \cup \{\neq\}$.*

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

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 $\{R(x_i, x_j) \mid R \in \mathcal{R} \cup \{\neq\}, i, j \in [k]\}$. 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 $R(x_i, x_j)$ in ϕ where R is not the equality relation, then ϕ cannot be satisfied by an assignment where all variables are assigned the same value, since R by assumption is pairwise disjoint with the equality relation. Hence, R_k cannot be qfpp-defined by $\mathcal{R}$.

For the second statement, let α be an assignment of a pair (x_i, x_j) to some $R \in \mathcal{R}$. We observe the following.

- If R is the equality relation, then $R_k(x_1, \ldots, x_k) \wedge R(x_i, x_j)$ 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 $R(x_i, x_j)$, it follows that all variables must be assigned the same value.
- If R is not the equality relation, then $R_k(x_1, \ldots, x_k) \wedge R(x_i, x_j)$ is logically equivalent to the conjunction of $(x_i \neq x_j)$ for every $i, j \in [k]$ where $i \neq j$. To see this, note that 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 $R(x_i, x_j)$ (again, recall that R by assumption is pairwise disjoint with the equality relation), 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. □

Assume that $\mathcal{R}_e$ is a partition scheme. Let $\mathcal{S}_e = \{R_i \mid i \geq 1\}$ where R_i is defined as in Lemma 4, and let $\mathcal{T}_e = \mathcal{R}_e \cup \{\neq\}$. Note that both $\mathcal{S}_e$ and $\mathcal{T}_e$ are qffo reducts of $\mathcal{R}_e$ regardless of the precise choice of $\mathcal{R}_e$. We first verify that $\mathcal{S}_e$, despite being infinite, admits a straightforward simplification map to the target language $\mathcal{T}_e = \mathcal{R}_e \cup \{\neq\}$.

Lemma 5. *There is a simplification map Σ_e from $\mathcal{S}_e$ to $\mathcal{T}_e$ that can be accessed in polynomial time.*

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 compute $\Sigma_e(R_k(x_1, y, x_2, \ldots, x_{k-})_{|\alpha})$ 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 $\Sigma_e(R_k(x_1, y, \ldots, x_k)_{|\alpha'})$ to a suitable CSP($\mathcal{T}_e$) instance as prescribed by Lemma 4.

For the other case, 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 4. Otherwise, we map $\Sigma_e(R_k(x_1, \ldots, x_k)_{|\alpha})$ to a suitable CSP($\mathcal{T}_e$) instance as prescribed by Lemma 4. We conclude the proof by noting that these computations are easy to perform in polynomial time so Σ_e is trivially polynomial-time accessible. □

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 [11] but this does not hold if the sets in $\mathcal{F}$ have size bounded by some constant.

Theorem 3. $[\mathcal{S}_e, \mathcal{T}_e, \mathcal{R}_e]$-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 4). 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 $\alpha : B \to \mathcal{R}_e$ 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 5), 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 $\mathcal{R}_e$ constraints, due to the first condition of Lemma 4. □

One may note that CSP($\mathcal{S}_e$) is polynomial-time solvable since $(0,\ldots,0) \in R_k$ for all k. Thus, the $[\mathcal{S}_e, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION problem is 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 5 can be extended to the source language $\mathcal{S}'_e$ so the proof of Theorem 3 implies W[2]-hardness of $[\mathcal{S}'_e, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION, too.

Corollary 1. *Let $\mathcal{R}$ denote a partition scheme with infinite domain D. There exists an infinite constraint language $\mathcal{S}$ and a finite language $\mathcal{T}$ that are qffo definable in $\mathcal{R}$ such that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$-BACKDOOR DETECTION is W[2]-hard.*

Finally, we can now answer the question (that was raised in Section 3.3.3) 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 3 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 2. *The problem $[\{R_3\}, \mathcal{T}_e, \mathcal{R}_e]$-BACKDOOR DETECTION is NP-hard.*

4. SIDEDOORS

We introduce the notion of *sidedoors* in this section. The basic idea is illustrated via an example in Section 4.1. The example points out an intrinsic problem with the backdoor approach, and that the sidedoor approach has the potential of being faster than the backdoor approach in certain cases. We continue by formally defining sidedoors in Section 4.2. In analogy with simplification maps for backdoors, our sidedoor definition is based on the idea of *branching maps* and we discuss the construction of such maps in Section 4.3. We finally prove that the sidedoor evaluation and detection problems are fixed-parameter tractable when parameterised by solution size (Sections 4.4 and 4.5, respectively).

4.1. **Motivating Example.** We use RCC-5 as the basis for this example so $\Theta = \{\mathsf{DR}, \mathsf{PO}, \mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{EQ}\}$ denotes the set of basic relations in RCC-5 and $\Theta^{\vee=}$ is the set of all relations definable by unions of the basic relations. The problem CSP(Θ) is tractable while CSP($\Theta^{\vee=}$) is NP-complete [46]. Now, recall Example 4. 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 $\alpha\colon B \to \Theta$. We know that if $(x,y) \in B$ (or, symmetrically, $(y,x) \in B$) then $R(x,y) \wedge (\alpha(x,y))(x,y)$ is either

(1) unsatisfiable if $\alpha(x,y) \cap R = \emptyset$, or
(2) equivalent to $\alpha(x,y)$.

This implies that the simplification map either outputs an unsatisfiable CSP(Θ) instance or replaces the constraints with the equivalent constraint over the basic relations in Θ. Furthermore, this implies that a backdoor $B \subseteq V^2$ has to cover *every* constraint in the instance which is not already included in Θ. This results in an $O(5^{|B|}) \cdot \mathrm{poly}(||I||)$ time algorithm for RCC-5, which can be slightly improved to $O(4^{|B|}) \cdot \mathrm{poly}(||I||)$ with the observation that only the trivial relation $(\mathsf{DR}, \mathsf{PO}, \mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{EQ})$ contains all the five basic relations.

It is easy to improve upon this by considering a larger tractable constraint language than the set of basic relations. Consider the language

$$\Gamma' = \Theta^{\vee=} \setminus \{(\mathsf{PP}, \mathsf{PP}^{-1}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{DR}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{EQ}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{EQ})\}$$

which is a well-known tractable class of RCC-5 [46]. Assume that we extend the definition of a backdoor so that we allow α to be a function from B to $\Theta^{\vee=}$, i.e., one may assign a pair of variables a union of the basic relations. Then we can use Γ' in connection with the simplification map

$$\Sigma(\{R(x,y)\}) = \{\{(R \cap \theta)(x,y)\} \mid \theta \in \{(\mathsf{PP}), (\mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{PO}, \mathsf{EQ})\}\} \text{ and } R \cap \theta \neq \emptyset.\}$$

Note that $R \cap \theta$ with $\theta \in \{(\mathsf{PP}), (\mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{PO}, \mathsf{EQ}))\}$ is always a relation in Γ'. With this extension, we can perform backdoor solving with Γ' and Σ in $2^{|B|} \cdot \text{poly}(||I||)$ time; the algorithm is a straightforward generalisation of the algorithm underlying Theorem 1.

This idea can be generalised as follows. Instead of looking at subinstances containing two variables, we may look at subinstances with a larger number of variables and perform a similar branching process where we replace subinstances containing "problematic" constraints with logically equivalent instances that do not contain such constraints. This approach has the potential of sometimes leading to faster algorithms than the backdoor approach (as indicated above and in the forthcoming Example 11). We formalise and analyse the sidedoor approach in the following sections.

4.2. Definition of Sidedoors.

We will now formally define sidedoors. We simplify the presentation by first introducing *sidedoor triples* which should be viewed as the sidedoor analogue of backdoor triples.

Definition 6. *Let $\mathcal{S}$ and $\mathcal{T}$ denote two relational structures such that $\mathcal{T} \subseteq \mathcal{S}$, and let $r \geq 1$ be an integer. We say that $[\![\mathcal{S}, \mathcal{T}, r]\!]$ is a* sidedoor triple *and we refer to*

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

In a backdoor triple $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$, we require $\mathcal{S}$ and $\mathcal{T}$ to be qffo reducts of $\mathcal{R}$, so $\mathcal{S}$ and $\mathcal{T}$ have an underlying connection via the structure $\mathcal{R}$. Such a connection between $\mathcal{S}$ and $\mathcal{T}$ is not enforced in a sidedoor triple $[\![\mathcal{S}, \mathcal{T}, r]\!]$ where $\mathcal{S}$ and $\mathcal{T}$ can be chosen freely under the condition $\mathcal{T} \subseteq \mathcal{S}$.

Definition 7. *Assume $[\![\mathcal{S}, \mathcal{T}, r]\!]$ is a sidedoor triple and let (V, C) be an instance of CSP($\mathcal{S}$). We say that $S \subseteq 2^V$ is a* sidedoor *into CSP($\mathcal{T}$) with radius r if the following hold:*

(1) *each set in S has size at most r, and*
(2) *for arbitrary $R(x_1, \ldots, x_k) \in C$ with $R \notin \mathcal{T}$, there is a set $s \in S$ such that $\{x_1, \ldots, x_k\} \subseteq s$.*

We say that a constraint $R(x_1, \ldots, x_k) \in C$ is covered *by the sidedoor S if there is a set $s \in S$ such that $\{x_1, \ldots, x_k\} \subseteq s$.*

The intuitive idea is that the constraints that are covered by the sidedoor are "difficult" to handle. Thus, one can concentrate on "simplifying" the constraints that are covered by the sidedoor: this can be done via a branching strategy where the constraints that are covered by the sidedoor are replaced by computationally tractable constraints in a solution-preserving way. To make this work, one needs to restrict the size of the sets in the sidedoor: if we remove (1) from Definition 7, then every instance (V, C) of CSP($\mathcal{S}$) has the sidedoor $\{V\}$ of size 1! Note that if a CSP($\mathcal{S}$) instance (V, C) contains a constraint $R(x_1, \ldots, x_k)$ where $R \notin \mathcal{T}$ and where $x_1, \ldots, x_k$ are all distinct variables, then (V, C) does not admit any sidedoor with radius $r < k$. We remark that sidedoors work equally well for infinite and finite domains since we do not make any assumptions concerning the domains of the languages $\mathcal{S}$ and $\mathcal{T}$.

We next introduce branching maps in order to formalise the idea of "simplifying hard constraints".

Definition 8. *Assume that $[\![\mathcal{S}, \mathcal{T}, r]\!]$ is a sidedoor triple and let $x_1, \ldots, x_r$ denote distinct variables. Let $\mathcal{S}_r$ be the set of CSP($\mathcal{S}$) instances with variable set $\{x_1, \ldots, x_r\}$ and let $\mathcal{T}_r$ be the set of CSP($\mathcal{T}$)*

instances with variable set $\{x_1, \ldots, x_r\}$. Let Ω be a mapping from $\mathcal{S}_r$ to $2^{\mathcal{T}_r}$ that satisfies the following condition: for every $I \in \mathcal{S}_r$,

$$\mathrm{Sol}(I) = \bigcup_{I' \in \Omega(I)} \mathrm{Sol}(I').$$

Then, we say that Ω is a branching map *from $\mathcal{S}$ to $\mathcal{T}$ with radius r.*

Unlike simplification maps, it is essential that a branching map is a total function from $\mathcal{S}_r$ to $2^{\mathcal{T}_r}$. The *branching factor* of Ω is $\max\{|\Omega(I)| \,:\, I \in \mathcal{S}_r\}$, and it is directly correlated to the complexity of the sidedoor evaluation problem as we will see in Section 4.4. Note that if a a language Γ contains infinitely many relations, then there may not be an upper bound on the size of CSP(Γ) instances on r variables. Thus, we sometimes need to restrict ourselves to polynomial-time computable branching maps. We now reconsider Examples 3 and 4 with respect to sidedoors.

Example 8. *Recall that the equality relation δ from Example 1 is defined as follows:*

$$\delta = \{(y_1, y_2, y_3) \in \mathbb{Q}^3 \mid (y_1 = y_2 \wedge y_1 \neq y_3) \vee (y_1 \neq y_2 \wedge y_2 = y_3)\}.$$

Based on the definition of δ, we see that the NP-hard problem CSP($\{\delta\}$) admits a simple branching map Ω from $\{\delta\}$ to $\{=, \neq\}$ with radius 3 and branching factor 2. Let x_1, x_2, x_3 denote distinct variables and let $V = \{x_1, x_2, x_3\}$. We see, for instance, that

$$\Omega((V, \{\delta(x_1, x_2, x_3)\})) = \{(V, \{x_1 = x_2, x_1 \neq x_3\}), (V, \{x_1 \neq x_2, x_2 = x_3\})\},$$
$$\Omega((V, \{\delta(x_1, x_1, x_3)\})) = \{(V, \{x_1 \neq x_3\})\},$$

and

$$\Omega((V, \{\delta(x_1, x_2, x_3), \delta(x_2, x_3, x_1)\})) = \emptyset$$

Example 9. *We know from Section 4.1 that CSP(Γ') is polynomial-time solvable when*

$$\Gamma' = \Theta^{\vee=} \setminus \{(\mathsf{PP}, \mathsf{PP}^{-1}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{DR}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{EQ}), (\mathsf{PP}, \mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{EQ})\}.$$

We will use the relation $(\mathsf{PP}^{-1}, \mathsf{DR}, \mathsf{PO}, \mathsf{EQ})) \in \Gamma'$ frequently so we let λ denote it. Let x_1, x_2, x_3 denote distinct variables and let $V_2 = \{x_1, x_2\}$ and $V_3 = \{x_1, x_2, x_3\}$.

A suitable branching map from $\Theta^{\vee=}$ to Γ' with radius 2 and branching factor 2 is the following:

$$\Omega_2((V_2, \{R(x_1, x_2)\})) = \{(V_2, \{(R \cap (\mathsf{PP}))(x_1, x_2)\}), (V_2, \{(R \cap \lambda)(x_1, x_2)\})\}.$$

To construct a branching map Ω_3 from Θ to Γ' with radius 3, we simply extend the idea behind Ω_2. Thus, consider the following branching map:

$$\Omega((V_3, \{R_{12}(x_1, x_2), R_{23}(x_2, x_3), R_{13}(x_1, x_3)\})) =$$
$$\{(V_3, \{(R_{12} \cap r_{12})(x_1, x_2), (R_{23} \cap r_{23})(x_2, x_3), (R_{13} \cap r_{13})(x_1, x_3)\}) \mid r_{12}, r_{23}, r_{13} \in \{(\mathsf{PP}), \lambda\}\}$$

It has branching factor 8 but this can be improved. It is easy to verify that the constraint set $\{\mathsf{PP}(x_1, x_2), \mathsf{PP}(x_2, x_3), \lambda(x_1, x_3)\}$ is not satisfiable: note that the constraints $\mathsf{PP}(x_1, x_2)$ and $\mathsf{PP}(x_2, x_3)$ forces the relation $\mathsf{PP}(x_1, x_3)$ to hold since PP is a transitive relation. Thus, Ω can be refined into a branching map Ω_3 with branching factor 7.

4.3. **Computing Branching Maps.**

Assume that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a backdoor triple with finite $\mathcal{S}$ and $\mathcal{T}$. Then, a simplification map from $\mathcal{S}$ to $\mathcal{T}$ always exists, and it can be computed whenever CSP($\mathcal{R}$) is decidable by Lemma 3. If we consider a sidedoor triple $[\![\mathcal{S}, \mathcal{T}, r]\!]$, then a branching map does not always exist. This is obvious if the arity of some relation in $\mathcal{S} \setminus \mathcal{T}$ exceeds r, but there are other reasons for the non-existence of a branching map, too. For instance, if we let $R_+ = \{(x, y, z) \in \mathbb{Q} \mid x = y + z\}$ and $R_* = \{(x, y, z) \in \mathbb{Q} \mid x = y \cdot z\}$, then it is easy to verify that $[\![\{R_+, R_*\}, \{R_+\}, 3]\!]$ does not admit a branching map and, consequently, $[\![\{R_+, R_*\}, \{R_+\}, r]\!]$ does not admit a branching map for any $r \geq 3$.

For every backdoor triple $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$, we know that $\mathcal{S}$ and $\mathcal{T}$ are qffo definable in $\mathcal{R}$. Inspired by this, we show how to exploit qffo definability for identifying a natural class of sidedoor triples that admit branching maps. Recall that a first-order formula is *positive* if it does not contain negation.

Lemma 6. *Let $[\![\mathcal{S}, \mathcal{T}, r]\!]$ be a sidedoor triple where $\mathcal{S}$ and $\mathcal{T}$ are finite. Assume that the following holds:*

(1) *$\mathcal{S}$ is quantifier-free positively definable in $\mathcal{T}$, and*
(2) *the maximum arity a of the relations in $\mathcal{S}$ satisfies $a \leq r$.*

Then, one can compute a branching map from $\mathcal{S}$ to $\mathcal{T}$ with radius r.

Proof. Let $x_1, \ldots, x_r$ be distinct variables. Let $\mathcal{S}_r$ and $\mathcal{T}_r$ denote the set of all CSP($\mathcal{S}$) and CSP($\mathcal{T}$) instances on variable set $\{x_1, \ldots, x_r\}$, respectively. We construct a branching map Ω. Arbitrarily pick $I = (V, C) \in \mathcal{S}_r$.

If every constraint $R(\bar{x})$ in C satisfies $R \in \mathcal{T}$, then we let $\Omega(I) = I$. Otherwise, pick one constraint $R(\bar{x}) \in C$ such that R is not in $\mathcal{T}$. We may (without loss of generality) assume that the positive qffo $\mathcal{T}$-definition of R is in DNF since the conversion to DNF can be done without introducing any negations. Hence, $\phi(\bar{x}) = \psi_1(\bar{x}) \vee \cdots \vee \psi_m(\bar{x})$ is the DNF definition of $R(\bar{x})$ in $\mathcal{T}$ where each ψ_i is a conjunction of constraints based on (unnegated) relations in $\mathcal{T}$. Obviously, each ψ_i can be viewed as an instance of CSP($\mathcal{T}$). Let

$$X = \{(V, (C \setminus \{R(\bar{x})\}) \cup \psi_1(\bar{x})), \ldots, (V, (C \setminus \{R(\bar{x})\}) \cup \psi_m(\bar{x}))\}.$$

It is an immediate consequence of $\phi(\bar{x})$ being a definition of $R(\bar{x})$ that

$$\mathrm{Sol}(I) = \bigcup_{I' \in X} \mathrm{Sol}(I')$$

If the instances in X now contain constraints with relations from $\mathcal{T}$ only, then we are done and we let $\Omega(I) = X$. Otherwise, repeat the process of recursively replacing relations in X that are not in Γ' by their definitions in Γ'. At least one constraint $R(\bar{x}) \in C$ with $R \in \mathcal{S} \setminus \mathcal{T}$ is removed in every recursive step so we will eventually end up with a non-empty set of instances X' satisfying $\mathrm{Sol}(I) = \bigcup_{I' \in X'} \mathrm{Sol}(I')$ and where each member of X' is an instance of CSP($\mathcal{T}$) (since $r \geq a$). Thus, we let $\Omega(I) = X'$. $\square$

Example 10. *It is easy to find examples where Lemma 6 is applicable: if $\mathcal{T}$ is homogeneous and finitely bounded (see Example 5) and $\mathcal{S}$ a first-order reduct of $\mathcal{T}$ then it can also be defined as a qffo-reduct of $\mathcal{T}$.*

The branching map computed in Lemma 6 has only a finite number of possible inputs and can consequently be accessed in constant time. It is obviously not guaranteed to have minimal branching factor—in particular, note that the branching factor may vary depending on the choice of DNF definitions for the relations in $\mathcal{S}$.

We emphasise that the restriction to quantifier-free definitions in Lemma 6 is necessary. Define $A_k(x, y) \equiv x = y + k$. We see that $A_2(x, y) = \exists z. A_1(x, z) \wedge A_1(z, y)$ so A_2 is fo-definable in $\{A_1\}$. Let $\mathcal{T} = \{A_1\}$ and $\mathcal{S} = \{A_1, A_2\}$. There is no branching map from $\mathcal{S}$ to $\mathcal{T}$ with radius 2: this would imply that $A_2(x, y)$ is logically equivalent to either $A_1(x, y)$ or $A_1(x, y) \vee A_1(y, x)$.

It is also necessary that the definitions are positive. Consider the equality and disequality relations $=$ and $\neq$ over some domain D. Define the source language $\mathcal{S} = \{=, \neq\}$ and the target language $\mathcal{T} = \{=\}$. Obviously, every relation in $\mathcal{S}$ is qffo definable in $\mathcal{T}$ (but $\neq$ is not qffo positive definable in $\mathcal{T}$). We see that there is no branching map of radius 2 from $\mathcal{S}$ to $\mathcal{T}$: this would imply that $\neq$ equals $=$. However, if the target language is finite and JEPD, then the result holds even if we use general quantifier-free definitions: every atomic formula $\neg R(x, y)$ can be replaced by $\bigvee_{S \in \Gamma' \setminus \{R\}} S(x, y)$ as

```
                    **algorithm** $A((V, C), S)$
(1) **if** $S = \emptyset$ **then return** $A'((V, C))$
(2) arbitrarily choose $s \in S$
(3) let $\{I'_1, \ldots, I'_p\} = \Omega'((V, C)[s])$
(4) apply algorithm $A$ to the instances
        • $J_1 = ((V, C) \oplus I_1, S \setminus \{s\})$
        • $J_2 = ((V, C) \oplus I_2, S \setminus \{s\})$
        • $\vdots$
        • $J_p = ((V, C) \oplus I_p, S \setminus \{s\})$
(5) **if** any call returns YES **then return** YES **else return** NO
```

FIGURE 2. Algorithm for sidedoor evaluation.

was pointed out in Section 3.3.1. This implies the following connection between backdoors and sidedoors.

Lemma 7. *Assume that $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$ is a backdoor triple with finite $\mathcal{S}, \mathcal{T}, \mathcal{R}$ and that a is the maximum arity of the relations in $\mathcal{S}$. Then, there is a branching map for the sidedoor triple $[\![\mathcal{S} \cup \mathcal{R}, \mathcal{T} \cup \mathcal{R}, r]\!]$ whenever $r \geq \max(2, a)$.*

Proof. This follows immediately from Lemma 6 since $\mathcal{S} \cup \mathcal{R}$ is qffo positive definable in the JEPD language $\mathcal{R}$ which is a subset of $\mathcal{T} \cup \mathcal{R}$. □

We require that $r \geq \max(2, a)$ since the relations in $\mathcal{R}$ have arity 2. This way we cover the (fairly trivial) case when $\mathcal{S}$ only contains relations of arity 1.

4.4. Sidedoor Evaluation.

We begin this section by introducing the sidedoor evaluation problem.

$[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR EVALUATION	
Input:	An instance (V, C) of CSP($\mathcal{S}$) and a sidedoor $S \subseteq 2^V$ of radius r into CSP($\mathcal{T}$).
Question:	Is (V, C) satisfiable?

We will now analyse the complexity of $[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR EVALUATION. We show that this problem is fixed-parameter tractable (under natural side conditions) when parameterised by sidedoor size. The additional conditions concern the existence of a branching map that can be accessed in polynomial time and that has bounded branching factor. We need some notation for describing our algorithm for this problem. Given a CSP instance $I = (V, C)$ and a set $V' \subseteq V$, we define the *restriction* of I to V' (denoted by $I[V']$) as the instance $(V', \{R(x_1, \ldots, x_k) \in C \mid \{x_1, \ldots, x_k\} \subseteq V'\})$.

Let $I = (V, C)$ be an arbitrary instance of CSP(Γ) and $I' = (V', C')$ be an arbitrary instance of CSP(Γ') with $V' \subseteq V$. We let $I \oplus I'$ denote the instance

$$(V, (C \setminus \{R(x_1, \ldots, x_k) \in C \mid \{x_1, \ldots, x_k\} \subseteq V'\}) \cup I').$$

That is, $I \oplus I'$ is the instance I where every constraint that is completely covered by V' is removed, and then the constraints in I' are added to I. Clearly, if $I[V']$ and I' have the same set of satisfying assignments, then I is satisfiable if and only if $I \oplus I'$ is satisfiable.

Theorem 4. *Assume that $[\![\mathcal{S}, \mathcal{T}, r]\!]$ is a sidedoor triple. Let Ω denote a polynomial-time computable branching map from $\mathcal{S}$ to $\mathcal{T}$ with radius r and branching factor $c < \infty$. If CSP($\mathcal{T}$) is polynomial-time solvable, then $[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR EVALUATION is solvable in time $c^{|S|} \cdot \mathrm{poly}(||I||)$ for a given sidedoor S. In particular, $[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR EVALUATION is fpt when parameterised by $|S|$.*

Proof. We assume that A' is a polynomial-time algorithm for CSP($\mathcal{T}$). Let (V, C) be an instance of CSP($\mathcal{S}$) that has a sidedoor S into CSP($\mathcal{T}$) with radius r. We assume (without loss of generality) that every $s \in S$ satisfies $|s| = r$. If this is not the case, then those sets that are smaller than r can be augmented with arbitrary variables. The branching map Ω is only defined for constraints over variable sets $X = \{x_1, \ldots, x_r\}$. We modify it into a map Ω' that is applicable to arbitrary sets of variables as follows. Let $Y = \{y_1, \ldots, y_r\}$ denote a set of variables and let σ denote an arbitrary bijection from X to Y. Given an instance (Y, C_Y) of CSP($\mathcal{S}$), we let $\sigma((Y, C_Y)) = (X, C_X)$ denote the instance (Y, C_Y) with variables y_i, $1 \leq i \leq r$, replaced by $\sigma(y_i)$. Let $\{I_1, \ldots, I_p\} = \Omega((X, C_X))$ and $\Omega'((Y, C_Y)) = \{\sigma^{-1}(I_1), \ldots, \sigma^{-1}(I_p)\}$.

We claim that (V, C) can be solved by Algorithm A in Figure 2. We first show that the algorithm always computes the correct answer. To prove this, we begin by verifying that $S \setminus \{s\}$ is a sidedoor for $(V, C) \oplus I_i$, $1 \leq i \leq p$. Arbitrarily choose one of the instances $J_q = ((V, C_q), S \setminus \{s\})$, $1 \leq q \leq p$, from line (4) where $C_q = C \setminus \{R(x_1, \ldots, x_k) \in C \mid \{x_1, \ldots, x_k\} \subseteq B'\} \cup C_q$ by the definition of $(V, C) \oplus I_q$. We verify the two properties in Definition 7.

(1) The set S is a sidedoor for (V, C) with radius r so each set in S has size r. The same obviously holds for $S \setminus \{s\}$.
(2) We need to show that for arbitrary $R(x_1, \ldots, x_k) \in C_q$ with $R \notin \mathcal{T}$, there is a set $s' \in S \setminus \{s\}$ such that $\{x_1, \ldots, x_k\} \subseteq s'$. Arbitrarily choose $R(x_1, \ldots, x_k) \in C_q$ with $R \notin \mathcal{T}$. Assume to the contrary that there is no $s' \in S \setminus \{s\}$ such that $\{x_1, \ldots, x_k\} \subseteq s'$. Since S is a sidedoor for I, this implies that $\{x_1, \ldots, x_k\} \subseteq s$. This leads to a contradiction since all relations in $\Omega'((V, C)[s])$ are in $\mathcal{T}$ by the definition of branching maps.

We prove correctness by induction over $|S|$. If $|S| = 0$, then (V, C) is an instance of CSP($\mathcal{T}$) and the correct answer is computed in line (1). Assume the algorithm computes correct answers whenever $|S| < m$. We show that this holds also when $|S| = m$. Consider the CSP instances

$$\begin{aligned} J'_1 &= (V, C) \oplus I_1 \\ J'_2 &= (V, C) \oplus I_2 \\ &\vdots \\ J'_p &= (V, C) \oplus I_p \end{aligned}$$

that are computed in line (4). Condition (2) in the definition of branching maps immediately implies that

$$\mathrm{Sol}(I) = \bigcup_{i=1}^{p} \mathrm{Sol}(J'_i).$$

If the algorithm answers Yes, then

$$A((V, C) \oplus I_q, S \setminus \{s\}) = \text{Yes}$$

for some $1 \leq q \leq p$. The induction hypothesis implies that $(V, C) \oplus I_q)$ is satisfiable since $|S \setminus \{s\}| < m$ and we know that $S \setminus \{s\}$ is a sidedoor for $(V, C) \oplus I_q$. Hence, $\mathrm{Sol}((V, C))$ is non-empty and (V, C) has a solution. Similarly, if the algorithm answers No, then none of $J'_1, \ldots, J'_p$ has a solution so the set $\mathrm{Sol}((V, C))$ is empty and (V, C) has no solution.

We conclude the proof by analysing the time complexity of algorithm A. Let s denote the size of the sidedoor. Now consider a branch in the algorithm

$$(V, C) \rightarrow (V, C_1) \rightarrow \cdots \rightarrow (V, C_s)$$

where (V, C) is the original instance, (V, C_1) is an instance $(V, C) \oplus I_i$, and so on.

Since Ω (and consequently Ω') is polynomial-time computable, we can assume that $||\Omega(I)|| \leq t(||I||)$ for some polynomial t. This implies that

$$||(V,C) \oplus I_i|| \leq ||(V,C)|| + t(||(V,C)[s]||) \leq ||(V,C)|| + t(||(V,C)||).$$

Hence, we can without loss of generality assume that (V, C_p), $1 \leq p \leq s$, have size at most $||(V,C)|| + p \cdot t(||(V,C)||)$. Let $b = ||(V,C)|| \cdot s \cdot t(||(V,C)||)$ and note that b is trivially an upper bound on $|C| + p \cdot t(||(V,C)||)$, $1 \leq p \leq s$. We simplify our argument by assuming that all CSP instances encountered during an execution of the algorithm has size b.

By inspecting algorithm A, we see that its running time is bounded by a function T in the size of the sidedoor with recursive definition

$$T(s) = \begin{cases} T'(b) & \text{if } s = 0 \\ c \cdot T(s-1) + \text{poly}(b) & \text{if } s > 0 \end{cases}$$

where c is the branching factor of Ω and Ω', and T' is the time complexity of algorithm A'. Thus, for sufficiently large $||I||$ there exists constants q, q' such that

$$T(s) \leq c^s \cdot T'(b) \cdot \text{poly}(b) \leq c^s \cdot b^q \cdot b^{q'} = c^s \cdot (||I|| \cdot s \cdot \alpha)^{q+q'} = c^s \cdot ||I||^{q+q'} \cdot (s \cdot \alpha)^{q+q'}.$$

We conclude that the algorithm is fixed-parameter tractable since c and α are constants that do not depend on the input. $\square$

By applying Theorem 4 to Example 8, we see that CSP($\{S\}$) can be solved in $O(2^s)$ time where s is sidedoor size. We illustrate some more aspects of sidedoors in the next example.

Example 11. *Let us once again consider RCC-5 and Example 9. Assume I is an instance of CSP($\Theta^{\vee=}$) with n variables and that it has an s-element sidedoor with radius r. Theorem 4 implies that I can be solved in $O(2^s \cdot \text{poly}(||I||))$ time by using branching map Ω_2 if $r = 2$ and in $O(3^s \cdot \text{poly}(||I||))$ time by using branching map Ω_3 if $r = 3$. This represents an improvement for instances with small sidedoors since the fastest known algorithm for CSP($\Theta^{\vee=}$) runs in $2^{O(n \log n)}$ time* [33]. *More precisely, the sidedoor approach beats this algorithm for instances with sidedoor size in $o(n \log n)$.*

We now illustrate how sidedoors with large radius may be beneficial. Let $\mathcal{I}$ denote the set of RCC-5 instances $I = (V, C)$ where

- *$V = \{x_0, \ldots, x_{n-1}\}$ where n is divisible by three.*
- *the relations between variables $x_{3k}, x_{3k+1}, x_{3k+2}$ are in $\Gamma \setminus \Gamma'$ for $0 \leq k \leq n/3 - 1$, and*
- *all other constraints are in Γ'.*

We partition V into $n/3$ partitions $V_k = \{x_{3k}, x_{3k+1}, x_{3k+2}\}$, $0 \leq k \leq n/3 - 1$. Arbitrarily choose $I \in \mathcal{I}$. We know that I has a backdoor

$$B = \{\{x_i, x_j\} \mid x_i, x_j \in V_k,\ x_i \neq x_j,\ \text{and } 0 \leq k \leq n/3 - 1\}$$

with $|B| = n$ and this set is additionally a sidedoor with radius 2. If we use the branching map Ω_2 from Example 9 together with the sidedoor B, then the CSP for the instances in $\mathcal{I}$ can be solved in $O(2^n \cdot \text{poly}(||I||))$ time by Theorem 4 since Ω_2 has branching factor 2. Note that the sidedoor approach beats the backdoor approach (using backdoor B) which runs in $O(4^{|B|} \cdot \text{poly}(||I||) = O(4^n \cdot \text{poly}(||I||)$ time according to Example 4.

Now, it is easy to verify that the set $S = \{V_0, \ldots, V_{n/3-1}\}$ is a sidedoor with radius 3 and $|S| = n/3$. This implies (by Theorem 4) that the CSP for the instances in $\mathcal{I}$ can be solved in $O(7^{n/3} \cdot \text{poly}(||I||)) \subseteq O(1.9130^n \cdot \text{poly}(||I||))$ time since Ω_3 has branching factor 7.

4.5. **Sidedoor Detection.** We begin by introducing the sidedoor detection problem.

$[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR DETECTION	
Input:	An instance (V, C) of CSP($\mathcal{S}$) and an integer k
Question:	Does (V, C) contain a sidedoor to $\mathcal{T}$ of radius r and size at most k.

If $\mathcal{S}, \mathcal{T}$ contain only unary relations, then SIDEDOOR DETECTION is a trivial problem (regardless of the radius). Similarly, if $\mathcal{S}, \mathcal{T}$ contain relations that are at most binary, then $[\![\mathcal{S}, \mathcal{T}, 2]\!]$-SIDEDOOR DETECTION is a trivial problem. Thus, the most basic case when SIDEDOOR DETECTION is possibly computationally hard is when $\mathcal{S}, \mathcal{T}$ contain at most binary relations and the radius is 3. We prove NP-hardness of this case via a reduction from a graph partitioning problem. A *partition* of a set S is a collection of disjoint subsets of S such that their union equals S. Let K_n denote the undirected complete graph on n vertices, i.e. every two distinct vertices are connected by an edge.

EDGE PARTITION(r)	
Input:	Undirected graph $G = (V, E)$
Question:	Can E be partitioned into sets $E_1, E_2, \ldots$ such that each E_i induces a subgraph of G that is isomorphic to K_r?

Theorem 5. *([27]) EDGE PARTITION(r) is NP-complete for every $r \geq 3$.*

We can now demonstrate that $(\mathcal{S}, \mathcal{T}, r)$-SIDEDOOR DETECTION is in general an NP-hard problem.

Theorem 6. *$[\![\{R, R'\}, \{R\}, 3]\!]$-SIDEDOOR DETECTION is NP-hard when R, R' are arbitrary binary relations.*

Proof. We prove this by a polynomial-time reduction from EDGE PARTITION(3). Arbitrarily choose an undirected graph $G = (V, E)$ and assume without loss of generality that $|E|/3$ is an integer. Construct an instance $I = ((V, C), k)$ of $[\![\{R, R'\}, \{R\}, r]\!]$-SIDEDOOR DETECTION where $C = \{R(x, y) \mid \{x, y\} \in E\}$ and $k = |E|/3$.

If G is a YES-instance of EDGE PARTITION(3), then let $E_1, \ldots, E_m$ denote such a partition of the edges in G. Each set E_i, $1 \leq i \leq m$, contains three edges so $m = |E|/3$. Each E_i, $1 \leq i \leq m$, is defined on a set of three vertices so we let S_i, $1 \leq i \leq m$, denote the corresponding set of vertices. It is obvious that $\{S_1, \ldots, S_m\}$ is a sidedoor of radius 3 for (V, C) and it contains at most $|E|/3$ sets. Thus I is a YES-instance.

Assume now that I is a YES-instance of $[\![\{R, R'\}, \{R\}, 3]\!]$-SIDEDOOR DETECTION. Then there is a sidedoor $\{S_1, \ldots, S_k\}$ with $k = |E|/3$ and each S_i contains at most three variables since the radius is three. The instance $(V, C)|_{S_i}$, $1 \leq i \leq k$, must thus contain three distinct constraints since the sidedoor cover all constraints in C and $|C| = |E|$. Furthermore, if there is a constraint $R(x, y)$ in C such that $\{x, y\} \subseteq S_i$ and $\{x, y\} \subseteq S_j$ for some $1 \leq i \neq j \leq k$, then the sidedoor covers strictly less that $|E|$ constraints in C. This implies that the sets in the sidedoor are pairwise disjoint. We conclude that the sidedoor is a partitioning of the set V and that $E_1, \ldots, E_k$ with

$$E_i = \{\{v, w\} \mid v, w \text{ are distinct elements in } S_i\}$$

is an edge partition of G. The graph G is consequently a YES-instance. □

We next prove that $[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR DETECTION is fpt when parameterised by the size of the sidedoor. We say that a variable $v \in V$ in a CSP instance (V, C) is *isolated* if it does not appear in the scope of any constraint in C.

Theorem 7. *$[\![\mathcal{S}, \mathcal{T}, r]\!]$-SIDEDOOR DETECTION is solvable in $(rk)^{(rk)} \cdot \mathrm{poly}(||I||)$ time where k is the size of the sidedoor. Hence, the problem is in FPT when parameterised by k.*

	Finite languages		Infinite languages	
	Backdoor	Sidedoor	Backdoor	Sidedoor
Detection	$\binom{a}{2}^{\lvert B\rvert+1} \cdot \lvert\mathcal{R}\rvert^{\lvert B\rvert}$	$(r\lvert S\rvert)^{r\lvert S\rvert}$	W[2]-hard	$(r\lvert S\rvert)^{r\lvert S\rvert}$
Evaluation	$\lvert\mathcal{R}\rvert^{\lvert B\rvert}$	$c^{\lvert S\rvert}$	(*)	$c^{\lvert S\rvert}$

Table 1. Summary of complexity results (polynomial-time factors are omitted). (*) indicates that this problem is not interesting since backdoor detection is computationally hard.

Proof. Let $((V, C), k)$ be an instance of $[\![\mathcal{S}, \mathcal{T}, r]\!]$-Sidedoor Detection. Let (V, C') be (V, C) where all constraints whose relation is in $\mathcal{T}$ are removed. Let (V', C') be (V, C') with all isolated variables removed. If $|V'| > rk$, then (V, C) does not have a sidedoor of size at most k since every variable in V' must be contained in at least one set of the sidedoor, and a sidedoor of radius r with k elements can cover at most rk variables. Otherwise, enumerate all sets $\{S_1, \ldots, S_k\}$ where S_i, $1 \leq i \leq k$ is a subset of V' of size r. The instance $((V, C), k)$ has a sidedoor if and only if at least one of these sets is a sidedoor. We see that there are at most $((rk)^r)^k = (rk)^{rk}$ such sets: our universe has size at most rk, there are at most $(rk)^r$ subsets of size r, and we want to choose at most k such sets. Thus, the algorithm runs in $(rk)^{(rk)} \cdot \text{poly}(||I||)$ time. □

5. Summary and Research Questions

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). We have noted that the backdoor approach sometimes leads to inferior algorithms for binary constraints. Inspired by this, we introduced the sidedoor approach where subinstances induced by the sidedoor are rewritten in a solution-preserving way. We prove that the detection and evaluation problems for sidedoors are in FPT, and we demonstrate that the time complexity of solving CSPs with sidedoors is in general incomparable with the backdoor approach. The running times of the two approaches are summarised in Table 1 where $|B|$ and $|S|$ denote backdoor and sidedoor size, respectively. For a backdoor triple $[\mathcal{S}, \mathcal{T}, \mathcal{R}]$, we let a denote the maximum arity of the relations in $\mathcal{T}$ and for a sidedoor triple $[\![\mathcal{S}, \mathcal{T}, r]\!]$, we let c denote the branching factor of the branching map under consideration. The results for backdoors and sidedoors are, naturally, not directly comparable since their sizes are not connected to each other in some obvious way. One may still draw interesting conclusions, though, such that the complexity of sidedoor evaluation is independent of the sidedoor triple—all that matters is the branching factor of the branching map. Thus, for problems admitting a branching map with low branching factor, the sidedoor approach may be faster than the backdoor approach even if sidedoors happen to be moderately larger than backdoors.

We continue this section by discussing a few possible directions for future research.

Broader tractable classes. Recent advances concerning backdoor sets for SAT and finite-domain CSPs provide a number of promising directions for future work. For instance, Gaspers et al. (2017a) 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. (2017) 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. The detection of heterogeneous backdoors is still fixed-parameter tractable in many natural cases. 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. These approaches are highly interesting for sidedoors, too.

Another interesting direction that is relevant for both back- and sidedoors is to drop the requirement that they 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 [31], and concrete examples are given by certain extensions of Allen's algebra that are not solvable in $2^{o(n \log n)}$ time.

Another promising direction for future research is compact representations of backdoors. This approach has been studied for finite-domain CSPs with the aid of decision trees or the more general concept of *backdoor DNFs* [44, 47]. The idea is to use decision trees or backdoor DNFs for representing 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 [44, 47].

The detection and evaluation problems 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.

We note that the situation is slightly different for sidedoors. Given a sidedoor triple $[\![\mathcal{S}, \mathcal{T}, r]\!]$, we know that the sidedoor detection problem is fpt even if $\mathcal{S}$ and/or $\mathcal{T}$ are infinite. The problem with sidedoors is instead that the radius r limits the set of instances that the method is applicable to. A motivated question here is what happens if we let the radius be a function of the instance instead of being a fixed constant. If the radius is allowed to increase linearly in the number of variables, then it is possible to adapt the W[2]-hardness result in Section 3.4 to the sidedoor setting. If the radius grows sublinearly, then the consequences are not clear. It is, however, clear that the algorithm underlying Theorem 7 cannot lead to fpt algorithms when the radius is allowed to depend on the given instance, so other algorithmic techniques are needed for this approach to work.

Computation of simplification maps. 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 to compute 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. From a general point of view, the problem which we want to address is the following. Given a relation represented by a first-order formula over $\mathcal{R}$ (i.e., corresponding to a simplified constraint) we wish to decide whether it is possible to find a CSP($\mathcal{T}$) instance whose set of models coincides with this relation. This problem is known in the literature as the *inverse constraint satisfaction problem* over a constraint language $\mathcal{T}$ (Inv-CSP($\mathcal{T}, \mathcal{R}$)), and it 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 [34, 37], 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 [3], but it is also important from an AI perspective [25, 28, 29]. 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.

There are several other interesting research directions concerning the computation of simplification maps. Our definition of simplification maps requires them to be maximally defined, in the sense that the map must be defined if a simplification over the target language exists. This is not a problem when both the target and source languages are finite, but if e.g. the source language is infinite then the landscape becomes more complicated, in particular if the general construction problem is undecidable. Thus, does it 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?

Computation of branching maps. Branching is one of the basic techniques for designing exponential-time algorithms for combinatorial problems such as the CSP [42, 53]. Typically, one divide the problem recursively into smaller subproblems, and these smaller subproblems are often computed by rewriting a small neighbourhood around a variable or a constraint in two or more ways. This process is then required to preserve solvability in the following way: the original instance is satisfiable if and only if at least one of the branching instances is satisfiable. The close connection with branching maps is quite obvious: the small neighbourhood is the induced subinstance with a particular radius, the rewriting process is carried out with the aid of the branching map, and we require that this is done in a solution-preserving way (which is a stronger condition than the one needed for branching algorithms). Given that it is a major research area to find good branching algorithms for combinatorial problems, it is safe to assume that the computation of branching maps with reasonable branching factors is a very difficult problem, and that reasonable branching maps are highly problem dependent. Thus, it would be interesting to compare the bounds obtained by sidedoors and backdoors to the current best algorithms for other types of CSPs. For which instances is each approach particularly suitable?

Integer programming. A challenging research direction is to use the backdoor approach for (mixed) integer linear programming ((M)ILP), which can be seen as an example of infinite CSPs that are not based on a finite set of JEPD relations. One very prominent result in this direction is Lenstra's celebrated result showing that MILP is fixed-parameter tractable parameterized by the number of integer variables [38], which can be seen as a (variable) backdoor to LP. In this context it seems particularly promising to explore variable (or even constraint) backdoors to the recently introduced tractable classes using the treedepth of the primal and dual graph of an MILP instance [7, 14] as such backdoors have the potential to generalise Lenstra's algorithm. It seems unlikely, though, that a general approach (as the one introduced in this article) will be fruitful for MILP: we believe it is more likely that the backdoor approach for MILP requires tailor-made solutions for each tractable fragment. However, we still believe that our backdoor approach may serve as inspiration for this research direction. We also believe that sidedoors may be relevant in this context but this is a more speculative idea.

Satisfiability modulo theories. *Satisfiability modulo theories* (SMT) is a formalism where one is given a first-order formula (with respect to a background theory Φ) and wants to determine whether the formula admits at least one model. The reader unfamiliar with this formalism may e.g. consult the introduction by Barrett et al. (2009) . Thus, for a theory Φ we let SMT(Φ) be the problem of determining whether a given first-order Φ-formula is satisfiable. One may discuss what a suitable backdoor concept for SMT is but one plausible idea is that given such a formula, we extend it with atomic formulas (corresponding to the function α in the CSP backdoor approach) and rewrite the resulting formula into some tractable class of formulas (corresponding to the simplification map). However, this idea immediately makes it possible to view the CSP backdoor problem for infinite language as a restricted version of the backdoor problem for SMT, and this even holds for the empty theory which corresponds to equality languages. The consequence of this is obvious: the hardness example in Section 3.4 leads to W[2]-hardness since every relation that is used can be computed in polynomial time.

If we look at sidedoors instead, then the difficulties become worse since we now want to rewrite subformulas of the given input formula. If the formula is written in CNF, then we can most likely use an approach similar to branching maps. However, requiring the formula to be in CNF is unrealisitc since it is directly opposed to the idea of SMT being an efficient way of representing computational problems: naturally, we can convert the input formula to CNF but this is not efficient since there are DNF formulas whose CNF equivalents are exponentially larger.

References

[1] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009.

[2] L. Barto and M. Pinsker. The algebraic dichotomy conjecture for infinite domain constraint satisfaction problems. In *Proc. 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS-2016)*, pages 615–622, 2016.

[3] M. Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction.* Cambridge University Press, 2021.

[4] M. Bodirsky and P. Jonsson. A model-theoretic view on qualitative constraint reasoning. *Journal of Artificial Intelligence Research*, 58:339–385, 2017.

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

[6] M. Bodirsky and S. 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.

[7] C. Brand, M. Koutecký, and S. Ordyniak. Parameterized algorithms for MILPs with small treedepth. In *Proc. 35th AAAI Conference on Artificial Intelligence (AAAI-2021)*, pages 12249–12257, 2021.

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

[9] C. Carbonnel and M. Cooper. Tractability in constraint satisfaction problems: a survey. *Constraints*, 21(2):115–144, 2016.

[10] C. Carbonnel, M. C. Cooper, and E. Hébrard. On backdoors to tractable constraint languages. In *Proc. 20th International Conference on Principles and Practice of Constraint Programming (CP-2014)*, pages 224–239, 2014.

[11] R. Downey and M. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer, 1999.

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

[13] F. Dylla, J. Lee, T. Mossakowski, T. Schneider, A. V. Delden, J. V. D. 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.

[14] F. Eisenbrand, C. Hunkenschröder, K. Klein, M. Koutecký, A. Levin, and S. Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019.

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

[16] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[17] R. Ganian, M. S. Ramanujan, and S. Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Transactions on Algorithms*, 13(2):29:1–29:32, 2017.

[18] M. Garey and D. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman and Company, 1979.

[19] 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.

[20] 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. 2017.

[21] 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.

[22] A. Goerdt. On random ordering constraints. In *Proc. 4th International Computer Science Symposium in Russia (CSR-2009)*, volume 5675, pages 105–116, 2009.

[23] V. Guruswami, J. Håstad, R. Manokaran, P. Raghavendra, and M. Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

[24] W. Guttmann and M. Maucher. Variations on an ordering theme with constraints. In *Proc. 4th IFIP International Conference on Theoretical Computer Science (TCS-2006)*, volume 209, pages 77–90, 2006.

[25] R. Hirsch. Relation algebras of intervals. *Artificial intelligence*, 83(2):267–295, 1996.

[26] W. Hodges. *Model theory.* Cambridge University Press, 1993.

[27] I. Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[28] J. 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.

[29] P. Jonsson. Constants and finite unary relations in qualitative constraint reasoning. *Artificial Intelligence*, 257:1–23, 2018.

[30] P. Jonsson and V. Lagerkvist. An initial study of time complexity in infinite-domain constraint satisfaction. *Artificial Intelligence*, 245:115–133, 2017.

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

[32] P. Jonsson, V. Lagerkvist, and S. Ordyniak. Reasoning short cuts in infinite domain constraint satisfaction: Algorithms and lower bounds for backdoors. In *Proc. 27th International Conference on Principles and Practice of Constraint Programming (CP-2021)*, pages 32:1–32:20, 2021.

[33] P. Jonsson, V. Lagerkvist, and G. Osipov. Acyclic orders, partition schemes and CSPs: Unified hardness proofs and improved algorithms. *Artificial Intelligence*, 296:103505, 2021.

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

[35] 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.

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

[37] 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.

[38] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[39] G. Ligozat and J. Renz. What is a qualitative calculus? A general framework. In *Proc. 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2004)*, pages 53–64, 2004.

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

[41] R. H. Möhring, M. Skutella, and F. Stork. Scheduling with AND/OR precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

[42] D. Morrison, S. Jacobson, J. Sauppe, and E. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[43] B. Nebel and H. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[44] S. Ordyniak, A. Schidler, and S. Szeider. Backdoor DNFs. In *Proc. 30th International Joint Conference on Artificial Intelligence (IJCAI-2021)*, pages 1403–1409, 2021.

[45] 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.

[46] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1-2):69–123, 1999.

[47] M. Samer and S. Szeider. Backdoor trees. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 363–368, 2008.

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

[49] 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.

[50] 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.

[51] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. 5th National Conference on Artificial Intelligence (AAAI-1986)*, pages 377–382, 1986.

[52] 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.

[53] G. Woeginger. Exact algorithms for NP-hard problems: a survey. In M. Juenger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization – Eureka! You Shrink!*, pages 185–207, 2000.

[54] D. Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM*, 67(5):30:1–30:78, 2020.

(P. Jonsson) Dep. Computer and Information Science, Linköpings Universitet, Sweden
Email address: `peter.jonsson@liu.se`

(V. Lagerkvist) Dep. Computer and Information Science, Linköpings Universitet, Sweden
Email address: `victor.lagerkvist@liu.se`

(S. Ordyniak) School of Computing, University of Leeds, Leeds, UK
Email address: `sordyniak@gmail.com`