Artificial Intelligence 245 (2017) 115–133

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

An initial study of time complexity in infinite-domain constraint satisfaction

Peter Jonsson [a], Victor Lagerkvist [b,*]

[a] Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden
[b] Institut für Algebra, TU Dresden, 01069 Dresden, Germany

ARTICLE INFO

Article history:
Received 6 November 2015
Received in revised form 9 January 2017
Accepted 25 January 2017
Available online 27 January 2017

Keywords:
Constraint satisfaction
Infinite domain
Time complexity

ABSTRACT

The constraint satisfaction problem (CSP) is a widely studied problem with numerous applications in computer science and artificial intelligence. For infinite-domain CSPs, there are many results separating tractable and NP-hard cases while upper and lower bounds on the time complexity of hard cases are virtually unexplored. Hence, we initiate a study of the worst-case time complexity of such CSPs. We analyze backtracking algorithms and determine upper bounds on their time complexity. We present asymptotically faster algorithms based on enumeration techniques and we show that these algorithms are applicable to well-studied problems in, for instance, temporal reasoning. Finally, we prove non-trivial lower bounds applicable to many interesting CSPs, under the assumption that certain complexity-theoretic assumptions hold. The gap between upper and lower bounds is in many cases surprisingly small, which suggests that our upper bounds cannot be significantly improved.

1. Introduction

This introductory section is divided into three parts: we begin by motivating our work, continue by discussing the problems that we study, and finally briefly present our results.

1.1. Motivation

The *constraint satisfaction problem* over a constraint language Γ (CSP(Γ)) is the problem of finding a variable assignment which satisfies a set of constraints, where each constraint is constructed from a relation in Γ. This problem is a widely studied computational problem and it can be used to model many classical problems such as k-coloring and the Boolean satisfiability problem, in a natural and uniform way. In the context of artificial intelligence, CSPs have been used for formalizing a wide range of problems, cf. Rossi et al. [55]. Efficient algorithms for CSP problems are hence of great practical interest. If the domain D is finite, then a CSP(Γ) instance I with variable set V can be solved in $O(|D|^{|V|} \cdot poly(\|I\|))$ time by enumerating all possible assignments. Hence, we have an obvious *upper bound* on the time complexity. This bound can, in many cases, be improved if additional information about Γ is known, cf. the survey by Woeginger [65] or the textbook by Gaspers [29]. There is also a growing body of literature concerning *lower bounds* [34,39,42,61].

* Corresponding author.
E-mail addresses: peter.jonsson@liu.se (P. Jonsson), victor.lagerqvist@tu-dresden.de (V. Lagerkvist).

http://dx.doi.org/10.1016/j.artint.2017.01.005
0004-3702/

When it comes to CSPs over infinite domains, there is a large number of results that identify polynomial-time solvable cases, cf. Ligozat [45] or Rossi et al. [55]. However, almost nothing is known about the time complexity of solving NP-hard CSP problems. One may conjecture that a large number of practically relevant CSP problems do not fall into the tractable cases, and this motivates a closer study of the time complexity of hard problems. Thus, we initiate such a study in this article.

1.2. Computational problems

Assume that we are given an instance of CSP(Γ) where Γ is a constraint language over an infinite domain. Which upper bounds can we provide for CSP(Γ)? Clearly, the method for finite-domain CSPs, based on enumerating all possible variable assignments, no longer work since the domain is infinite. In fact, infinite-domain CSPs are in general *undecidable* [7]. A first step is therefore to only consider decidable infinite-domain CSPs. However, even for such problems, for every recursive function, one can find a decidable CSP problem which cannot be solved faster than this [4]. Hence, we first need to fix a class of constraint languages X such that CSP(Γ) is included in a reasonable complexity class for every $\Gamma \in X$. Througout this article we exclusively study the case when CSP(Γ) is included in NP, since this is a natural and well-studied class of problems. However, when considering CSPs over infinite domains, representational issues also become highly important. A relation in a finite-domain CSP problem is easy to represent by simply listing the allowed tuples. When considering infinite-domain CSPs, the relations need to be implicitly represented. A natural way is to consider disjunctive formulas over a finite set of basic relations. Let $\mathcal{B}$ denote some finite set of basic relations such that CSP($\mathcal{B}$) is tractable. Let $\mathcal{B}^{\vee\omega}$ denote the closure of $\mathcal{B}$ under finitary disjunctions, and let $\mathcal{B}^{\vee k}$ be the subset of $\mathcal{B}^{\vee\omega}$ containing only disjunctions of length at most k. We first consider a finite-domain example for illustrative purposes: let $D = \{true, false\}$ and let $\mathcal{B} = \{B_1, B_2\}$ where $B_1 = \{true\}$ and $B_2 = \{false\}$. In other words a unary constraint of the form $B_1(x)$ forces the variable x to be mapped to *true*, and $B_2(y)$ forces the variable y to be mapped to *false*. It is then easy to see that CSP($\mathcal{B}^{\vee\omega}$) corresponds to the Boolean SAT problem while CSP($\mathcal{B}^{\vee k}$) corresponds to the k-SAT problem. Early examples of disjunctive constraints over infinite-domains can be found in, for instance, temporal reasoning [43,37,58], reasoning about action and change [26], and deductive databases [41]. More recent examples include interactive graphics [48], rule-based reasoning [46], and set constraints (with applications in descriptive logics) [10]. There are also works studying disjunctive constraints from a general point of view [16,21] but they are only concerned with the separation of polynomial cases from NP-hard cases, and do not further investigate the time complexity of the hard cases.

There is also an important connection to constraint languages containing first-order definable relations (see Section 2.2 for details). Assume Γ is a finite constraint language containing relations that are first-order definable in $\mathcal{B}$, and that the first order theory of $\mathcal{B}$ admits quantifier elimination. Then, upper bounds on CSP(Γ) can be inferred from results such as those that will be presented in Sections 3 and 4. This indicates that studying the time complexity of CSP($\mathcal{B}^{\vee\omega}$) is worthwhile, especially since our understanding of first-order definable constraint languages is rapidly increasing [8].

CSPs in certain AI applications are often based on binary basic relations and unions of them (instead of free disjunctive formulas). This is the predominant way of representing constraints in, for instance, spatial reasoning. Clearly, such relations are a subset of the relations in $\mathcal{B}^{\vee k}$ and we let $\mathcal{B}^{\vee=}$ denote this set of relations. We do not explicitly bound the length of disjunctions since they are bounded by $|\mathcal{B}|$. The literature on such CSPs is voluminous and we refer the reader to Renz and Nebel [54] for an introduction. We remark that there exists examples of undecidable CSP problems over constraint languages of the form $\mathcal{B}^{\vee=}$ [32]. Hence, even for such restricted problems it is impossible to give general upper bounds, unless additional restrictions are imposed on the set $\mathcal{B}$ of basic relations.

1.3. Our results

Throughout the article, we primarily measure time complexity in the number of variables. Historically, this has been the most common way of measuring time complexity: the vast majority of work concerning finite-domain CSPs concentrates on the number of variables — a SAT instance $I = (V, C)$ (where V is the set of variables and C is the set of clauses) may contain up to $2^{2|V|}$ distinct clauses if repeated literals are disallowed — and measuring in the instance size may give far too optimistic figures. This may be quite detrimental since naturally appearing test examples tend to contain a moderate number of constraints. In light of this, it is much more informative to know that SAT can be solved in $O(2^{|V|} \cdot poly(\|I\|))$ time (where $\|I\|$ denotes the total number of bits needed for representing I) instead of merely knowing that it is solvable in $O(2^{\|I\|} \cdot poly(\|I\|))$ time (which of course is true since $|V| \leq \|I\|$). For instance, we immediately conclude from the bound $O(2^{|V|} \cdot poly(\|I\|))$ that increasing the number of variables increases the run time much more rapidly than increasing the number of clauses. This is something that one cannot immediately infer from the bound $O(2^{\|I\|} \cdot poly(\|I\|))$.

Let us now turn to the time complexity of solving infinite-domain CSPs. To solve such problems in practice, backtracking algorithms are usually employed. The literature on heuristically guided backtracking algorithm and empirical analyses of such algorithms is huge: we refer the reader to any good textbook (such as Dechter [24] or the handbook edited by Rossi et al. [55]) on constraint satisfaction for more information about this. What we find lacking in the literature are analyses of the asymptotical performance of such algorithms, i.e. their worst-case behavior. Unfortunately, we show in Section 3 that they can be highly inefficient in the worst case. Let p denote the maximum arity of the relations in the set of basic

relations $\mathcal{B}$, let $m = |\mathcal{B}|$, and let $|V|$ denote the number of variables in a given CSP instance. We show (in Section 3.1) that the time complexity ranges from $O(2^{2^{m\cdot|V|^p}\cdot\log(m\cdot|V|^p)}\cdot poly(\|I\|))$ (which is doubly exponential with respect to the number of variables) for CSP($\mathcal{B}^{\vee\omega}$) to $O(2^{2^m\cdot|V|^p\cdot\log m}\cdot poly(\|I\|))$ time for $\mathcal{B}^{\vee=}$ (and the markedly better bound of $O(2^{|V|^p\log m}\cdot poly(\|I\|))$ if $\mathcal{B}$ consists of pairwise disjoint relations). The use of heuristics can probably improve these figures in some cases, but we have not been able to find such results in the literature and it is not obvious how to analyze backtracking combined with heuristics. At this stage, we are mostly interested in obtaining a baseline: we need to know the performance of simple algorithms before we start studying more sophisticated ones. However, some of these bounds can be improved by utilizing standard methods described in the literature: we demonstrate this in Section 3.2 by applying the highly influential *sparsification* method by Impagliazzo, Paturi, and Zane [36].

In Section 4 we switch strategy and show that disjunctive CSP problems can be solved significantly more efficiently via *enumerative methods*. By an enumerative method, we mean a method that is based on enumerating some kind of objects that can be used for determining whether the given instance has a solution or not. Let us for a moment go back to the simplest possible method for solving CSPs over a finite domain D: enumerate all assignments of values from D to the variable set V. This process yields a (very simple) algorithm running in $O(|D|^{|V|}\cdot poly(\|I\|))$ time. This is the archetypical example of an enumerative method. However, it is not directly applicable to infinite-domain CSPs due to the size of the set D.

We introduce two enumerative methods in this article: *structure enumeration* and *domain enumeration*. Structure enumeration is inspired by model checking for finite structure: we enumerate a sequence of structures (which themselves are small CSP instances) and check whether the given instance is satisfied by the (implicitly represented) solutions of the structures. Domain enumeration is more closely related to the enumerative approach to finite-domain CSPs. In certain cases, one can identify finite sets of 'canonical' domain elements with the following property: there exists a solution if and only if there is a solution that only uses the canonical elements. There are several important differences between these two methods but there is a general rule of thumb: structure enumeration is typically easier to apply and it has a greater range of applicability but it gives worse complexity figures than domain enumeration.

By using structure enumeration, we obtain the upper bound $O(2^{|V|^p\cdot m}\cdot poly(\|I\|))$ for CSP($\mathcal{B}^{\vee\omega}$). If we additionally assume that $\mathcal{B}$ is jointly exhaustive and pairwise disjoint then the running time is improved further to $O(2^{|V|^p\cdot\log m}\cdot poly(\|I\|))$. This bound beats or equals every bound presented in Section 3. We then proceed to show even better bounds for certain choices of $\mathcal{B}$ by using domain enumeration. For instance, we consider certain temporal CSPs.

In the last part of the article (Section 5), we consider the problem of determining lower bounds for CSP($\mathcal{B}^{\vee\omega}$), i.e. identifying functions f such that no algorithm for CSP($\mathcal{B}^{\vee\omega}$) has a better running time than $O(f(|V|))$. We accomplish this by relating CSP problems and certain complexity-theoretical conjectures, and obtain strong lower bounds for the majority of the problems considered in Section 4. As an example, we show that the temporal CSP($\{<,>,=\}^{\vee\omega}$) problem, where $<,>$ and $=$ are the order relations on $\mathbb{Q}$, is solvable in time $O(2^{|V|\log|V|}\cdot poly(\|I\|))$ but, assuming a conjecture known as the *strong exponential time hypothesis* (SETH), not solvable in $O(c^{|V|})$ time for *any* $c > 1$. Hence, even though the algorithms we present are rather straightforward, there is, in many cases, very little room for improvement, unless the SETH fails. It appears much more difficult to obtain lower bounds for problems of the type CSP($\mathcal{B}^{\vee=}$). However, we succeed in giving the lower bound $O((\sqrt{2})^{|V|})$ for Allen's interval algebra. This bound is not based on the (strong) exponential time hypothesis but on bounds on computing the chromatic number of graphs. The upper bound for Allen's algebra is $O(2^{2|V|\cdot(1+\log|V|)})$ so there is plenty of room for improvements in this case.

This article is a revised and extend version of an earlier conference publication [38].

2. Preliminaries

In this section, we formally define the constraint satisfaction problem, discuss first-order definable relations, and provide some basic definitions concerning SAT problems and the exponential time hypothesis.

2.1. Constraint satisfaction

We begin by providing a formal definition of the CSP problem when it is parameterized by a set of relations.

Definition 1. Let Γ be a set of finitary relations over some set D of values. The constraint satisfaction problem over Γ (CSP(Γ)) is defined as follows:

Instance: A set V of variables and a set C of *constraints* of the form $R(v_1,\ldots,v_k)$, where k is the arity of R, $v_1,\ldots,v_k \in V$ and $R \in \Gamma$.
Question: Is there a function $f : V \to D$ such that $(f(v_1),\ldots,f(v_k)) \in R$ for every $R(v_1,\ldots,v_k) \in C$?

The set Γ is referred to as the *constraint language*. Observe that we do not require Γ or D to be finite. Given an instance I of CSP(Γ) we write $\|I\|$ for the number of bits required to represent I. We now turn our attention to constraint languages based on disjunctions. Let D be a set of values and let $\mathcal{B} = \{B_1,\ldots,B_m\}$ denote a finite set of relations over D, i.e. $B_i \subseteq D^j$ for some $j \geq 1$. Let the set $\mathcal{B}^{\vee\omega}$ denote the set of relations defined by finitary disjunctions over $\mathcal{B}$. That is, $\mathcal{B}^{\vee\omega}$ contains every p-ary relation R such that $R(x_1,\ldots,x_p)$ if and only if $B_1(\mathbf{x}_1)\vee\cdots\vee B_t(\mathbf{x}_t)$ where $\mathbf{x}_1,\ldots,\mathbf{x}_t$ are sequences of variables

from $\{x_1, \ldots, x_p\}$ such that the length of $\mathbf{x}_j$ equals the arity of B_j, and $B_1, \ldots, B_t \in \mathcal{B}$. We refer to $B_1(\mathbf{x}_1), \ldots, B_t(\mathbf{x}_t)$ as the *disjuncts* of R. We assume, without loss of generality, that a disjunct occurs at most once in a disjunction. For $k \geq 1$ we define $\mathcal{B}^{\vee k}$ as the subset of $\mathcal{B}^{\vee\omega}$ where each relation is defined by a disjunction of length at most k. Hence, we also allow relations definable by disjunctions of length exactly k since, as was pointed out in Section 1.2 and will be evident in Section 2.3, it gives CSP($\mathcal{B}^{\vee k}$) a natural relationship to k-SAT. It is common, especially in qualitative temporal and spatial constraint reasoning, to study a restricted variant of $\mathcal{B}^{\vee k}$ where all relations in $\mathcal{B}$ have the same arity p. Define $\mathcal{B}^{\vee=}$ to contain every p-ary relation R such that $R(\mathbf{x})$ if and only if $B_1(\mathbf{x}) \vee \cdots \vee B_t(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_p)$.

We adopt a simple representation of relations in $\mathcal{B}^{\vee\omega}$: every relation R in $\mathcal{B}^{\vee\omega}$ is represented by its defining disjunctive formula. Note that two objects $R, R' \in \mathcal{B}^{\vee\omega}$ may denote the same relation. Hence, $\mathcal{B}^{\vee\omega}$ is not a constraint language in the sense of Definition 1. We avoid tedious technicalities by ignoring this issue and view constraint languages as multisets. Given an instance $I = (V, C)$ of CSP($\mathcal{B}^{\vee\omega}$) under this representation, we let

$$\mathrm{Disj}(I) = \{B_{i_1}(\mathbf{x}_1), \ldots, B_{i_t}(\mathbf{x}_t) \mid B_{i_1}(\mathbf{x}_1) \vee \cdots \vee B_{i_t}(\mathbf{x}_t) \in C\}$$

denote the set of all disjuncts appearing in I.

We close this section by introducing some notions that are common in qualitative spatial and temporal reasoning problems. Let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be a set of relations (over a domain D) such that all $B_1, \ldots, B_m$ have arity p. We say that $\mathcal{B}$ is *jointly exhaustive* (JE) if $\bigcup \mathcal{B} = D^p$ and that $\mathcal{B}$ is *pairwise disjoint* (PD) if $B_i \cap B_j = \emptyset$ whenever $i \neq j$. If $\mathcal{B}$ is both JE and PD we say that it is JEPD or, in mathematical terminology, $\mathcal{B}$ is a partitioning of the set D^p. Observe that if $B_1, \ldots, B_m$ have different arity then these properties are clearly not relevant since the intersection between two such relations is always empty.

Let Γ be an arbitrary set of relations with arity $p \geq 1$. We say that Γ is *closed under intersection* if $R_1 \cap R_2 \in \Gamma$ for all choices of $R_1, R_2 \in \Gamma$. Let R be an arbitrary binary relation. We define the *converse* $R^\smile$ of R such that $R^\smile = \{(y, x) \mid (x, y) \in R\}$. If Γ is a set of binary relations, then we say that Γ is *closed under converse* if $R^\smile \in \Gamma$ for all $R \in \Gamma$.

2.2. First-order definable relations

Languages of the form $\mathcal{B}^{\vee\omega}$ have a close connection to languages defined over first-order structures admitting quantifier elimination, i.e. every first-order definable relation can be defined by an equivalent formula without quantifiers. We have the following lemma.

Lemma 2. *Let Γ be a finite constraint language first-order definable over a relational structure $(D, R_1, \ldots, R_m)$ admitting quantifier elimination, where $R_1, \ldots, R_m$ are JEPD. Then there exists a k such that*

1. CSP(Γ) *is polynomial-time reducible to* CSP($\{R_1, \ldots, R_m\}^{\vee k}$) *and*
2. *if* CSP($\{R_1, \ldots, R_m\}^{\vee k}$) *is solvable in $O(f(|V|) \cdot poly(\|I\|))$ time, then* CSP(Γ) *is solvable in $O(f(|V|) \cdot poly(\|I\|))$ time.*

Proof. Assume that every relation $R \in \Gamma$ is definable through a quantifier-free first-order formula ϕ_i over $R_1, \ldots, R_m$. Let ψ_i be ϕ_i rewritten in conjunctive normal form. We need to show that every disjunction in ψ_i can be expressed as a disjunction over $R_1, \ldots, R_m$. Clearly, if ψ_i only contains positive literals, then this is trivial. Hence, assume there is at least one negative literal. Since $R_1, \ldots, R_m$ are JEPD it is easy to see that for any negated relation in $\{R_1, \ldots, R_m\}$ there exists $\Gamma \subseteq \{R_1, \ldots, R_m\}$ such that the union of Γ equals the complemented relation. We can then reduce CSP(Γ) to CSP($\{R_1, \ldots, R_m\}^{\vee k}$) by replacing every constraint by its conjunctive normal formula over $R_1, \ldots, R_m$. This reduction can be done in polynomial time with respect to $\|I\|$ since each such definition can be stored in a table of fixed size. Moreover, since this reduction does not increase the number of variables, it follows that CSP(Γ) is solvable in $O(f(|V|) \cdot poly(\|I\|))$ time whenever CSP($\mathcal{B}^{\vee k}$) is solvable in $O(f(|V|) \cdot poly(\|I\|))$ time. □

As we will see in Section 4, this result is useful since we can use upper bounds for CSP($\mathcal{B}^{\vee k}$) to derive upper bounds for CSP(Γ), where Γ consists of first-order definable relations over $\mathcal{B}$. There is a large number of structures admitting quantifier elimination and interesting examples are presented in every standard textbook on model theory, cf. Hodges [33]. A selection of problems that are highly relevant for computer science and AI are discussed in Bodirsky [8].

2.3. SAT and the exponential time hypothesis

The *propositional satisfiability problem* (SAT) will be important both for obtaining upper and lower bounds in later parts of this article. We define the SAT problem as usual: given a set of propositional clauses, decide whether there is a satisfying assignment or not. We sometimes consider the SAT problem restricted to clauses of length at most k and we denote this problem k-SAT. We pointed out the following fact in the introduction but it is worth repeating: if $D = \{true, false\}$ and $\mathcal{B} = \{B_1, B_2\}$ where $B_1 = \{true\}$ and $B_2 = \{false\}$, then CSP($\mathcal{B}^{\vee\omega}$) corresponds to SAT while CSP($\mathcal{B}^{\vee k}$) corresponds to k-SAT. Note that the problem CSP($\mathcal{B}^{\vee=}$) is different in this respect since it can be seen as an alternative formulation of 1-SAT, i.e., SAT restricted to unary clauses. SAT and k-SAT are NP-complete problems when $k \geq 3$ while 2-SAT and 1-SAT are solvable in polynomial time. We often use the domain $\{0, 1\}$ for Boolean values where 1 is interpreted as 'true' and 0 as 'false'.

NP-hardness does not give us any information concerning the running times of algorithms for solving such problems (besides the fact that they are superpolynomial under the side condition that P $\neq$ NP). For instance, under the sole assumption P $\neq$ NP, we cannot, for instance, rule out that SAT can be solved in $O(|V|^{\log |V|})$ time. The existence of such efficient algorithms are considered unlikely and to rule out such algorithms we need complexity assumptions that are stronger than P $\neq$ NP. The *exponential time hypothesis* (ETH) and the *strong exponential time hypothesis* (SETH) have been suggested as plausible stronger assumptions. These two hypotheses have been used quite intensively in the study of central problems in AI such as planning and constraint satisfaction, cf. Bäckström and Jonsson [2,3], Kanj and Szeider [42], and Traxler [61].

The ETH states that there exists a $\delta > 0$ such that 3-SAT is not solvable in $O(2^{\delta|V|})$ time by any deterministic algorithm, i.e. it is not solvable in subexponential time [34]. If the ETH holds, then there is an increasing sequence $\delta_3, \delta_4, \ldots$ of reals such that k-SAT cannot be solved in time $2^{(\delta_k-\epsilon)|V|}$ but it can be solved in $2^{(\delta_k+\epsilon)|V|}$ time for arbitrary $\epsilon > 0$. The *strong exponential-time hypothesis* (SETH) is the conjecture that the limit of the sequence $\delta_3, \delta_4, \ldots$ equals 1, and, as a consequence, that SAT is not solvable in time $O(2^{\delta|V|})$ for any $\delta < 1$ [34]. These conjectures have in recent years successfully been used for proving lower bounds of many NP-complete problems [47]. The plausibility of the (S)ETH is debatable due to the same reasons as the plausibility of P $\neq$ NP is debatable: our understanding of this kind of complexity questions is not sufficient. One ought to note, however, that the failure of any of these hypotheses would have far-reaching and surprising consequences in connection with, for instance, the existence of subexponential algorithms for many NP-complete problems [36,39,56], the complexity and approximability of optimization problems [18,49], and parameterized complexity theory [19,20].

3. Fundamental algorithms

In this section we investigate the complexity of algorithms for CSP($\mathcal{B}^{\vee\omega}$) and CSP($\mathcal{B}^{\vee k}$) based on branching on the disjuncts in constraints (Section 3.1) and the sparsification method (Section 3.2). Throughout this section we assume that $\mathcal{B}$ is a set of basic relations such that CSP($\mathcal{B}$) is in P. The reason behind this assumption is that the algorithms that we investigate in this section works by repeatedly choosing a set of disjuncts, and then checks whether this instance of CSP($\mathcal{B}$) is satisfiable or not. Clearly, this assumption is not the only possible one, but in practice it is not a great restriction, since the most frequently studied problems of the form CSP($\mathcal{B}^{\vee\omega}$) satisfy this condition.

3.1. Branching on disjuncts

Let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be a set of basic relations with maximum arity $p \geq 1$. Assume we have an instance I of CSP($\mathcal{B}^{\vee\omega}$) with variable set V. Such an instance contains at most $2^{m\cdot|V|^p}$ distinct constraints. Each such constraint contains at most $m \cdot |V|^p$ disjuncts so the instance I can be solved in

$$O((m \cdot |V|^p)^{2^{m\cdot|V|^p}} \cdot poly(\|I\|)) = O(2^{2^{m\cdot|V|^p}\cdot\log(m\cdot|V|^p)} \cdot poly(\|I\|))$$

time by enumerating all possible choices of one disjunct out of every disjunctive constraint. The satisfiability of the resulting sets of constraints can be checked in polynomial time due to our initial assumptions. How does such an enumerative approach compare to a branching search algorithm? In the worst case, a branching algorithm without heuristic aid will go through all of these cases so the bound above is valid for such algorithms. Analyzing the time complexity of branching algorithms equipped with powerful heuristics is a very different (and presumably very difficult) problem.

Assume instead that we have an instance I of CSP($\mathcal{B}^{\vee k}$) with variable set V. There are at most $m \cdot |V|^p$ different disjuncts which leads to at most $\sum_{i=0}^{k}(m|V|^p)^i \leq k \cdot (m|V|^p)^k$ distinct constraints. We can thus solve instances with $|V|$ variables in $O(k^{k\cdot(m|V|^p)^k} \cdot poly(\|I\|)) = O(2^{k\cdot\log k\cdot(m|V|^p)^k} \cdot poly(\|I\|))$ time.

Finally, let $I = (V, C)$ be an instance of CSP($\mathcal{B}^{\vee=}$) with variable set V. We analyze the size of C: given the variable set V, there are $|V|^p$ variable sequences of length p and there are 2^m different disjunctive relations over $\mathcal{B}$. Thus, there are at most $2^m \cdot |V|^p$ distinct constraints in C and each such constraint has length at most m. Non-deterministic guessing gives that instances of this kind can be solved in

$$O(m^{2^m\cdot|V|^p} \cdot poly(\|I\|)) = O(2^{2^m\cdot|V|^p\cdot\log m} \cdot poly(\|I\|))$$

time. This may appear to be surprisingly slow but this is mainly due to the fact that we have not imposed any additional restrictions on the set $\mathcal{B}$ of basic relations. Hence, assume that the relations in $\mathcal{B}$ are PD. Given two relations $R_1, R_2 \in \mathcal{B}^{\vee=}$, it is now clear that $R_1 \cap R_2$ is a relation in $\mathcal{B}^{\vee=}$, i.e. $\mathcal{B}^{\vee=}$ is closed under intersection. Let $I = (V, C)$ be an instance of CSP($\mathcal{B}^{\vee=}$). For any sequence of variables $(x_1, \ldots, x_p)$, we can assume that there is at most one constraint $R(x_1, \ldots, x_p)$ in C. This implies that we can solve CSP($\mathcal{B}^{\vee=}$) in $O(m^{|V|^p} \cdot poly(\|I\|)) = O(2^{|V|^p\log m} \cdot poly(\|I\|))$ time. Combining everything so far we obtain the following upper bounds.

Lemma 3. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and let $m = |\mathcal{B}|$. Then*

- CSP($\mathcal{B}^{\vee\omega}$) *is solvable in* $O(2^{2^{m\cdot|V|^p}\cdot\log(m\cdot|V|^p)} \cdot poly(\|I\|))$ *time,*
- CSP($\mathcal{B}^{\vee k}$) *is solvable in* $O(2^{k\cdot\log k\cdot(m|V|^p)^k} \cdot poly(\|I\|))$ *time,*

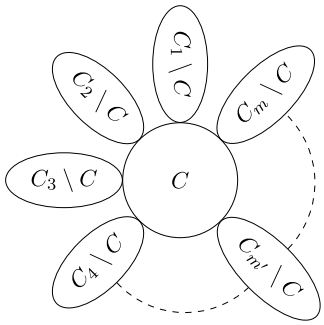

Fig. 1. A sunflower $\{C_1, \ldots, C_{m'}, \ldots, C_m\}$ with a heart C and m pairwise disjoint petals.

- CSP($\mathcal{B}^{\vee=}$) *is solvable in* $O(2^{2^{m\cdot|V|^p}\cdot \log m} \cdot poly(\|I\|))$ *time, and*
- CSP($\mathcal{B}^{\vee=}$) *is solvable in* $O(2^{|V|^p \log m} \cdot poly(\|I\|))$ *time if* $\mathcal{B}$ *is PD.*

A bit of fine-tuning is often needed when applying highly general results like Lemma 3 to concrete problems. For instance, Renz and Nebel [54] show that the RCC-8 problem can be solved in $O(c^{\frac{|V|^2}{2}})$ for some (unknown) $c > 1$. This problem can be viewed as CSP($\mathcal{B}^{\vee=}$) where $\mathcal{B}$ contains JEPD binary relations and $|\mathcal{B}| = 8$. Lemma 3 implies that CSP($\mathcal{B}^{\vee=}$) can be solved in $O(2^{3|V|^2})$ which is significantly slower if $c < 8^2$. However, it is well known that $\mathcal{B}$ is closed under converse. Let $I = (\{x_1, \ldots, x_n\}, C)$ be an instance of CSP($\mathcal{B}^{\vee=}$). Since $\mathcal{B}$ is closed under converse, we can always assume that if $R(x_i, x_j) \in C$, then $i \leq j$. Thus, we can solve CSP($\mathcal{B}^{\vee=}$) in $O(m^{\frac{|V|^2}{2}} \cdot poly(\|I\|)) = O(2^{\frac{|V|^2}{2} \log m} \cdot poly(\|I\|))$ time. This figure matches the bound by Renz and Nebel better when c is small.

3.2. Sparsification

The complexity of the algorithms proposed in Section 3 is dominated by the number of constraints. An idea for improving these running times is therefore to reduce the number of constraints within instances. One way of accomplishing this is by using *sparsification* [36]. This method was originally used for the k-SAT problem with the aim of proving that k-SAT instances with only a linear number (in $|V|$) constraints are still NP-complete and, in fact, that the ETH is still true for such instances. Recall from Section 2.3 that the ETH states that 3-SAT is not solvable in subexponential time. Sparsification can intuitively be described as the process of picking a disjunct that appears in a relatively large number of constraints, and create two instances I_1 and I_2, corresponding to the case where this disjunct is either true or false. In I_1 we can safely remove all constraints where this disjunct appears, and in I_2 all such constraints contain at least one less disjunct. We can then check the satisfiability of I by answering yes if and only if I_1 or I_2 is satisfiable. By repeating this process, we end up with a sequence of instances $I_1, \ldots, I_k$ such that at least one of $I_1, \ldots, I_k$ is satisfiable if and only if the original instance is satisfiable.

To concretize this idea, a *sunflower* is defined to be a set of clauses $\{C_1, \ldots, C_m\}$, containing the same number of disjuncts, such that $C_1 \cap \ldots \cap C_m \neq \emptyset$. Here, we tacitly view a clause C_i as a set of literals, and with this interpretation, the above condition states that the clauses have at least one literal in common. The clause $C_1 \cap \ldots \cap C_m = C$ is the *heart* of the sunflower and the clauses $C_1 \setminus C, \ldots, C_m \setminus C$ the *petals* of the sunflower. This structure is visualized in Fig. 1. By searching after a sunflower $C_1, \ldots, C_m$ where m is as large as possible we obtain the two instances I_1 and I_2 corresponding to the case where we branch on either the heart or the petals, and thus reducing either the number of constraints or the number of disjuncts in constraints. Sunflowers and related structures are important in combinatorics and there are several connections with central problems in computer science, cf. Alon et al. [1] or Jukna [40, Sec. 6]. For a more thorough and formal introduction to sparsification see Chapter 16.3 in Flum and Grohe [28]. Analyzing such a seemingly simple recursive strategy as described above is by no means trivial and we will not present the details. The analysis can be found in Impagliazzo et al. [36].

We will now use sparsification for solving infinite-domain CSPs. We need a few additional definitions. A *family of k-sets* $(U, \mathcal{C})$ consists of a finite set U (the *universe*) and a collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ where $S_i \subseteq U$ and $|S_i| \leq k$, $1 \leq i \leq m$. A *hitting set* for $\mathcal{C}$ is a set $C \subseteq U$ such that $C \cap S_i \neq \emptyset$ for each $S_i \in \mathcal{C}$. Let $\sigma(\mathcal{C})$ be the set of all hitting sets of $\mathcal{C}$. $\mathcal{T}$ is a *restriction* of $\mathcal{C}$ if for each $S \in \mathcal{C}$ there is a $T \in \mathcal{T}$ with $T \subseteq S$. If $\mathcal{T}$ is a restriction of $\mathcal{C}$, then $\sigma(\mathcal{T}) \subseteq \sigma(\mathcal{C})$. We then have the following result.[1]

[1] We remark that Impagliazzo et al. [36] use a slightly different terminology.

Theorem 4 *(Impagliazzo et al. [36]). For all $\varepsilon > 0$ and positive k, there is a constant K and an algorithm that, given a family of k-sets $(U, \mathcal{C})$ where $|U| = n$, produces a list of $t \leq 2^{\varepsilon \cdot n}$ restrictions $\mathcal{T}_1, \ldots, \mathcal{T}_t$ of $\mathcal{C}$ so that $\sigma(\mathcal{C}) = \bigcup_{i=1}^{t} \sigma(\mathcal{T}_i)$ and so that for each $\mathcal{T}_i$, $|\mathcal{T}_i| \leq Kn$. Furthermore, the algorithm runs in time $poly(n) \cdot 2^{\varepsilon \cdot n}$.*

Lemma 5. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and let $m = |\mathcal{B}|$. Then $\mathrm{CSP}(\mathcal{B}^{\vee k})$ is solvable in $O(2^{(\varepsilon + K \log k) \cdot |V|^p \cdot m} \cdot poly(\|I\|))$ time for every $\varepsilon > 0$, where K is a constant depending only on ε and k.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{CSP}(\mathcal{B}^{\vee k})$ with $C = \{c_1, \ldots, c_m\}$. To avoid unnecessary notation, we view each constraint $c = (R_1(\mathbf{x}_1) \vee \cdots \vee R_n(\mathbf{x}_n))$ as a set $\{R_1(\mathbf{x}_1), \ldots, R_n(\mathbf{x}_n)\}$ in this proof. Note that I has a solution if and only if there exists a set $X \subseteq \mathrm{Disj}(I)$ such that

1. (V, X) is satisfiable and
2. $X \cap c_i \neq \emptyset$, $1 \leq i \leq m$, i.e. X is a hitting set of C.

We will now apply Lemma 5 on the family of k-sets (U, C) where $U = \mathrm{Disj}(I)$: choose some $\varepsilon > 0$ and let $\{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$ be the resulting set of restrictions. Note that each $(V, \mathcal{T}_i)$ can be viewed as an instance of $\mathrm{CSP}(\mathcal{B}^{\vee k})$ under the convention of viewing disjunctions as sets.

We claim the following: there exists a $1 \leq i \leq t$ such that $\mathcal{T}_i$ is satisfiable if and only if I is satisfiable. Assume that I is satisfiable. Then there exists a hitting set $X \subseteq \mathrm{Disj}(I)$ of C such that (V, X) is satisfiable. Hence, $X \in \sigma(C)$. This implies that there exists a $1 \leq i \leq t$ such that $X \in \sigma(\mathcal{T}_i)$ since $\sigma(C) = \bigcup_{i=1}^{t} \sigma(\mathcal{T}_i)$. Since (V, X) is satisfiable, $(V, \mathcal{T}_i)$ is satisfiable, too.

Assume instead that there exists a $(V, \mathcal{T}_i)$, $1 \leq i \leq t$, such that $(V, \mathcal{T}_i)$ is satisfiable. Let s be a solution to $\mathcal{T}_i$. Let $X = \{R(\mathbf{x}) \in \mathrm{Disj}(I) \mid s \text{ satisfies } R(\mathbf{x})\}$ and note that (V, X) is satisfiable and X is a hitting set of $\mathcal{T}_i$. The set $\mathcal{T}_i$ is a restriction of C so for every $c \in C$, there exists a $T \in \mathcal{T}_i$ such that $T \subseteq c$. It follows that X is a hitting set for (V, C) which implies that s is a solution to (V, C).

We conclude that in order to prove that I is satisfiable, it is sufficient to find a satisfiable instance $(V, \mathcal{T}_i)$. Each instance $(V, \mathcal{T}_i)$ contains at most $K \cdot |U| \leq K \cdot |V|^p \cdot m$ distinct constraints, where K is a constant depending on ε and k, and can therefore be solved in time $O(poly(\|I\|) \cdot k^{K \cdot |V|^p \cdot m})$ by exhaustive search as in Section 3.1. This gives a total running time of

$$poly(|V|^p \cdot m) \cdot 2^{\varepsilon \cdot |V|^p \cdot m} + 2^{\varepsilon \cdot |V|^p \cdot m} \cdot k^{K \cdot |V|^p \cdot m} \cdot poly(\|I\|) \in$$
$$O(2^{\varepsilon \cdot |V|^p \cdot m} \cdot 2^{K \cdot |V|^p \cdot m \cdot \log k} \cdot poly(\|I\|)) = O(2^{(\varepsilon + K \log k) \cdot |V|^p \cdot m} \cdot poly(\|I\|))$$

since $t \leq 2^{\varepsilon \cdot n}$. □

This procedure can be implemented using only polynomial space, just as the methods presented in Section 3.1. This follows from the fact that the restrictions $\mathcal{T}_1, \ldots, \mathcal{T}_t$ of $\mathcal{C}$ can be computed one after another with polynomial delay [17, Theorem 5.15]. Although this running time still might seem excessively slow observe that it is significantly more efficient than the $2^{k \cdot \log k \cdot (m|V|^p)^k}$ algorithm for $\mathrm{CSP}(\mathcal{B}^{\vee k})$ in Lemma 3. However, in Theorem 6, Theorem 7, and Theorem 8 in Section 4.1 we will be able to improve upon this running time even further, by directly enumerating the hitting sets corresponding to the disjuncts of an instance, rather than reverting to backtracking algorithms as in Lemma 5. As we will demonstrate in Theorem 13, these bounds can also be strengthened for certain $\mathrm{CSP}(\mathcal{B}^{\vee k})$ problems, by using an idea influenced by sparsification.

4. Improved upper bounds

In this section, we show that it is possible to obtain markedly better upper bounds than the ones presented in Section 3. In Section 4.1 we consider algorithms for $\mathrm{CSP}(\mathcal{B}^{\vee \omega})$ based on *structure enumeration*, and in Section 4.2, we consider algorithms for $\mathrm{CSP}(\mathcal{B}^{\vee \omega})$ and $\mathrm{CSP}(\mathcal{B}^{\vee k})$ based on *domain enumeration*.

4.1. Structure enumeration

We begin by presenting a general algorithm for $\mathrm{CSP}(\mathcal{B}^{\vee \omega})$ based on the idea of enumerating all variable assignments that are implicitly described in instances. As in the case of Section 3 we assume that $\mathcal{B}$ is a set of basic relations such that $\mathrm{CSP}(\mathcal{B})$ is solvable in $O(poly(\|I\|))$ time.

Theorem 6. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and let $m = |\mathcal{B}|$. Then $\mathrm{CSP}(\mathcal{B}^{\vee \omega})$ is solvable in $O(2^{m|V|^p} \cdot poly(\|I\|))$ time.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{CSP}(\mathcal{B}^{\vee \omega})$. Let $S = \mathrm{Disj}(I)$ and note that $|S| \leq m|V|^p$. For each subset S_i of S first determine whether S_i is satisfiable. Due to the initial assumption this can be done in $O(poly(\|I\|))$ time since this set of disjuncts can be viewed as an instance of $\mathrm{CSP}(\mathcal{B})$. Next, check whether S_i satisfies I by, for each constraint in C, determine

whether at least one disjunct is included in S_i. Each such step can determined in time $O(poly(\|I\|))$ time. The total running time for this algorithm is therefore in $O(2^{m|V|^p} \cdot poly(\|I\|))$. □

The advantage of this approach compared to the branching algorithm in Section 3 is that enumeration of variable assignments is much less sensitive to instances with a large number of constraints. At this point, it may be interesting to discuss what is actually meant by 'a large number of constraints'. Assume we have a set $\mathcal{B} = \{B_1, \ldots, B_m\}$ of p-ary basic relations. Let us consider CSP($\mathcal{B}^{\vee 2}$) instances with $|V|^{2p}$ constraints. The number of constraints is thus polynomially bounded in the number of variables. Theorem 6 shows that we solve such instances in $O(2^{m|V|^p} \cdot poly(\|I\|))$ time. A backtracking algorithm, on the other hand, needs $O(2^{|V|^{2p}} \cdot poly(\|I\|))$ time if we reason in the same way as in Section 3.1, i.e. we need to choose one disjunct out of every constraint and we need to try all possibilities in the worst case. Obviously, $2^{|V|^{2p}} > 2^{m|V|^p}$ even for quite small $|V|$ and this indicates that structure enumeration beats branching algorithms even when the number of constraints are polynomially bounded in the number of variables.

We can speed up this result even further by making additional assumptions on the set $\mathcal{B}$. This allows us to enumerate smaller sets of constraints than in Theorem 6.

Theorem 7. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and let $m = |\mathcal{B}|$. Then* CSP($\mathcal{B}^{\vee\omega}$) *solvable in* $O(2^{|V|^p \cdot \log m} \cdot poly(\|I\|))$ *time if $\mathcal{B}$ is JEPD.*

Proof. Let $I = (V, C)$ be an instance of CSP($\mathcal{B}^{\vee\omega}$). Observe that every basic relation has the same arity p since $\mathcal{B}$ is JEPD. Let F be the set of functions from $|V|^p$ to $\mathcal{B}$ and for every $f \in F$, we let $S_f = \{B_j(\mathbf{x}) \mid \mathbf{x} \in V^p, f(\mathbf{x}) = B_j\}$. The set S_f contains the constraints that are specified by the function f so it contains one constraint for each tuple in V^p. The size of the set is polynomially bounded in (V, C) since p is a fixed constant that only depends on the choice of basic relations. We begin by proving two claims.

Claim 1. I is satisfiable if and only if there exists an $f \in F$ such that $(V, C \cup S_f)$ is satisfiable. If I is not satisfiable, then there trivially is no $f \in F$ such that $(V, C \cup S_f)$ is satisfiable. Assume instead that I has a solution s. Arbitrarily choose a tuple $(x_1, \ldots, x_p) \in V^p$. Since $\mathcal{B}$ is JEPD, the tuple $(s(x_1), \ldots, s(x_p))$ is a member of exactly one $B \in \mathcal{B}$. Thus, for every tuple $(x_1, \ldots, x_p) \in V^p$, there exists a unique $B \in \mathcal{B}$ such that $(s(x_1), \ldots, s(x_p)) \in B$. Define the function $g : V^p \to \mathcal{B}$ such that it returns this relation. By definition, g is a member of F. The function s is a solution to the CSP instance (V, S_f) due to the choice of f and this implies that s is a solution to the instance $(V, C \cup S_f)$, too.

Claim 2. If (V, S_f) is satisfiable for some $f \in F$, then we can check in polynomial time whether $(V, C \cup S_f)$ is satisfiable or not. Let s be a solution to (V, S_f). Arbitrarily choose a constraint $c = (c_1 \vee \cdots \vee c_k) \in C$. Consider $c_1 = B_i(\mathbf{x}_1)$ where $B_i \in \mathcal{B}$. There is a constraint $B_j(\mathbf{x}_1)$ in S_f by the construction of S_f. If $i = j$, then s satisfies the disjunct c_1 and thus the constraint c. If $i \neq j$, then s does not satisfy c_1 since $\mathcal{B}$ is PD. Otherwise, check the next disjunct and so on. If no disjunct $c_1, \ldots, c_k$ passes the test, then $C \cup S_f$ is not satisfiable. By repeating this process for all constraints in C, we can check whether $(V, C \cup S_f)$ is satisfiable or not. This can be done in polynomial time in the size of (V, C) since the size of the set S_f is polynomially bounded in the size of (V, C), as we noted in the beginning of the proof.

Consider the following algorithm for solving CSP($\mathcal{B}^{\vee\omega}$):

1. $ans := false$
2. for every $f \in F$ do the following
3. compute S_f
4. if (V, S_f) is satisfiable then
5. if $(V, C \cup S_f)$ is satisfiable then $ans := true$
6. return ans

We first verify that the algorithm is correct. If (V, C) is not satisfiable, then $(V, C \cup S_f)$ is not satisfiable for any choice of $f \in F$ and the algorithm will answer $false$. If (V, C) is satisfiable, then there exists an $f \in F$ such that $(V, C \cup S_f)$ is satisfiable by Claim 1 and the algorithm answers $true$. Note here that (V, S_f) is satisfiable, too, so the algorithm will indeed perform the test in Line 5.

We continue by analyzing its time complexity. Computing S_f takes polynomial time in the size of (V, C) since p and $|\mathcal{B}|$ are fixed constants that only depends on the choice of $\mathcal{B}$. Checking whether (V, S_f) is satisfiable or not takes polynomial time since CSP($\mathcal{B}$) is a polynomial-time solvable problem. Finally, checking whether $(V, C \cup S_f)$ is satisfiable or not takes polynomial time due to Claim 2. The set F contains $|\mathcal{B}|^{|V|^p} = 2^{|V|^p \log m}$ functions and these functions can be incrementally computed with negligible overhead. We conclude that the algorithm runs in $O(2^{|V|^p \cdot \log m} \cdot poly(\|I\|))$ time. □

Let us reconsider the RCC-8 example from Section 3.1 and let $\mathcal{B}$ denote the corresponding set of eight basic relations. We know (from Renz and Nebel [54]) that CSP($\mathcal{B}^{\vee=}$) is solvable in $O(c^{\frac{|V|^2}{2}})$ time for some $c > 1$, and we obtained the concrete bound $O(2^{\frac{3|V|^2}{2}} \cdot poly(\|I\|))$ time by utilizing a simple branching algorithm. Theorem 7(1) gives that CSP($\mathcal{B}^{\vee\omega}$) is solvable in

$O(2^{3|V|^2} \cdot poly(\|I\|))$ time. We can once again exploit the fact that $\mathcal{B}$ is closed under converse and instead of enumerating all functions from V^2 to $\mathcal{B}$ (as in the proof of Theorem 7), we assume that $V = \{x_1, \ldots, x_n\}$ and we merely enumerate the functions from $\{(x_i, x_j) \mid 1 \leq i < j \leq n\}$ to $\mathcal{B}$. This gives us the time bound $O(2^{\frac{3|V|^2}{2}} \cdot poly(\|I\|))$, i.e. we can solve $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ as fast as the severely restricted problem $\mathrm{CSP}(\mathcal{B}^{\vee=})$. This indicates that there may be more efficient algorithms for $\mathrm{CSP}(\mathcal{B}^{\vee=})$.

If the set of basic relations $\mathcal{B}$ are PD but not JE, then we get a slightly slower algorithm for $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$.

Theorem 8. *Let $\mathcal{B}$ be a set of basic relations with arity p and let $m = |\mathcal{B}|$. Then $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ is solvable in $O(2^{|V|^p \cdot \log(m+1)} \cdot poly(\|I\|))$ time if $\mathcal{B}$ is PD.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$. We introduce a symbol $\top$ for indicating that we do not care about the exact relation between the variables in a variable tuple. Let F' be the set of functions from $|V|^p$ to $\mathcal{B} \cup \{\top\}$ and for every $f \in F'$ let $S_f = \{B_j(\mathbf{x}) \mid \mathbf{x} \in V^p, f_i(\mathbf{x}) = B_j \neq \top\}$.

We say that a function $f \in F'$ is *compatible* if $f(\mathbf{x}) = B \neq \top$ for at least one disjunct $B(\mathbf{x})$ in each constraint in C. We begin by proving an auxiliary result: I is satisfiable if and only if there exists a compatible $f \in F'$ such that (V, S_f) is satisfiable. Assume there exists an $f \in F'$ such that (V, S_f) has a solution s. The fact that f is compatible implies that at least one disjunct in each constraint in C is satisfied by s. Thus, (V, C) is satisfiable.

Assume instead that (V, C) has the solution s. Let the set S contain one disjunct that is satisfied by s from each constraint in C. Define the function $f : V^p \to \mathcal{B} \cup \{\top\}$ such that $f(\mathbf{x}) = B$ if $B(\mathbf{x}) \in S$ and $f(\mathbf{x}) = \top$ otherwise. Note that f is a well-defined function since it cannot be the case (due to PD) that $B(\mathbf{x})$ and $B'(\mathbf{x})$ are simultaneously in S if $B \neq B'$. Also note that f is compatible since the solution s satisfies at least one disjunct in each constraint.

Consider the following algorithm for solving $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$:

1. $ans := false$
2. for every compatible $f \in F'$ do the following
3. compute S_f
4. if (V, S_f) is satisfiable then $ans := true$
5. return ans

The correctness of the algorithm was verified above. We continue by analyzing its time complexity. Computing S_f takes polynomial time in the size of (V, C) since p and $|\mathcal{B}|$ are fixed constants that only depends on the choice of $\mathcal{B}$. Checking whether (V, S_f) is satisfiable or not takes polynomial time since $\mathrm{CSP}(\mathcal{B})$ is a polynomial-time solvable problem. The set F contains $(|\mathcal{B}|+1)^{|V|^p} = 2^{|V|^p \log(m+1)}$ functions and these functions can be incrementally computed with negligible overhead. Furthermore, checking whether a function $f \in F'$ is compatible or not can be done in polynomial time. We conclude that the algorithm runs in $O(2^{|V|^p \cdot \log(m+1)} \cdot poly(\|I\|))$ time. □

4.2. Domain enumeration

A fundamental problem with structure enumeration is that the number of instances to be enumerated increases rapidly with the number of variables. This phenomenon is particularly noticeable if the basic relations have high arity: if the arity of the basic relations $\{B_1, \ldots, B_m\}$ is p, then we need to consider between $2^{m|V|^p}$ instances (in the general case) and $2^{\log m \cdot |V|^p}$ instances (in the JEPD case). We will suggest an alternative enumeration method in this section, *domain enumeration*, that offers a partial solution to the problems with structure enumeration. This section contains four parts: we begin by presenting the method and giving temporal reasoning examples in Sections 4.2.1 and 4.2.2, respectively. We continue by elaborating upon the method in Sections 4.2.3 and 4.2.4.

4.2.1. Basics

A possible solution to the problem outlined above is to enumerate domain elements instead — a method that is analogous to the basic algorithm for solving finite-domain CSPs. This approach presents certain difficulties, though:

1. there needs to exist some finite selection of elements that guarantees that solvable instances have solutions restricted to these elements,
2. the elements need to be representable in some suitable way, and
3. we need an efficient method for verifying whether a variable assignment using these elements is a solution or not.

We concretize these requirements in the next theorem.

Theorem 9. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and $m = |\mathcal{B}|$. Assume there exist functions $t, u : \mathbb{N} \to \mathbb{N}$ such that for arbitrary $n > 0$*

1. *there exist finite sets* $S_1^n, \ldots, S_{a_n}^n$ *for some* $a_n > 0$ *such that for every solvable instance* $I = (V, C)$ *of* CSP($\mathcal{B}$) *with* $|V| = n$, *there exists a solution* $f : V \to S_i^n$ *for some* $1 \leq i \leq a_n$,
2. *the set* $\{S_i^n \mid 1 \leq i \leq n\}$ *can be generated in* $t(n)$ *time, and*
3. *it can be verified in* $u(\|I\|)$ *time whether a function* $f : V \to S_i^{|V|}$ *is a solution to a given instance* $I = (V, C)$ *of* CSP($\mathcal{B}^{\vee\omega}$).

Let $b_i = \max\{|S_1^i|, \ldots, |S_{a_i}^i|\}$. *Then* CSP($\mathcal{B}^{\vee\omega}$) *is solvable in* $O(t(|V|) + a_{|V|} \cdot 2^{|V| \log b_{|V|}} \cdot u(\|I\|) \cdot poly(\|I\|))$ *time.*

Proof. Let $I = (V, C)$ be an arbitrary instance of CSP($\mathcal{B}^{\vee\omega}$). If I has a solution, then there is a solution $f : V \to S_i^{|V|}$ for some $1 \leq i \leq a_{|V|}$ by condition (1). Condition (2) allows us to compute the set $\mathcal{S} = \{S_i^n \mid 1 \leq i \leq n\}$. For each $S \in \mathcal{S}$, we generate every function from V to S and check whether it is a solution or not — there is a method for this by condition (3). Generating the set $\mathcal{S}$ takes $t(|V|)$ time by (2). Given an $S \in \mathcal{S}$, there are at most $(b_{|V|})^{|V|} = 2^{|V| \cdot \log b_{|V|}}$ functions from V to S, and the size of $\mathcal{S}$ is at most $a_{|V|}$ by (1). Checking whether such a function is a solution or not can be done in $u(\|I\|)$ time by (3). Taken together, it follows that CSP($\mathcal{B}^{\vee\omega}$) is solvable in $O(t(|V|) + a_{|V|} \cdot 2^{|V| \log b_{|V|}} \cdot u(\|I\|) \cdot poly(\|I\|))$ time. □

A basic requirement for structure enumeration is that CSP($\mathcal{B}$) is in P (or, at least, does not have too high time complexity). Observe that this is irrelevant in domain enumeration since it is sufficient to check whether concrete variable assignments are solutions or not.

4.2.2. Two examples from temporal reasoning

Let $\mathcal{T} = \{<, >, =\}$ denote the JEPD order relations on $\mathbb{Q}$. The CSP problem for $\mathcal{T}^{\vee=}$ is often referred to as the *time point algebra* and it has been intensively studied within the temporal reasoning community. It was realized quite early that CSP($\mathcal{T}$) is tractable [63] and, soon after, that CSP($\mathcal{T}^{\vee=}$) is tractable [62], too. It is also well-known that CSP($\mathcal{T}^{\vee\omega}$) is NP-complete. This follows from general results by Broxvall et al. [16] but it was known earlier: it can, for instance, quite easily be inferred from the original NP-hardness proof for Allen's algebra [63].

We now recall that Theorem 7 implies that CSP($\mathcal{T}^{\vee\omega}$) can be solved in $O(2^{|V|^2 \cdot \log 3} \cdot poly(\|I\|))$ time. We improve this bound using domain enumeration as follows.

Theorem 10. CSP($\mathcal{T}^{\vee\omega}$) *is solvable in* $O(2^{|V| \log |V|} \cdot poly(\|I\|))$ *time.*

Proof. Let $I = (V, C)$ be an arbitrary instance of CSP($\mathcal{T}^{\vee\omega}$). If I has a solution, then we claim that there is a solution $f : V \to \{1, \ldots, |V|\}$. To see this, let $f' : V \to \mathbb{Q}$ be an arbitrary solution to I. Assume $\{f'(v) \mid v \in V\} = \{a_1, \ldots, a_p\}$ where $a_1 < a_2 < \cdots < a_p$. Define $f : V \to \{1, \ldots, |V|\}$ such that $f(v) = i$ if and only if $f'(v) = a_i$. We see that f is a solution to I since $f(v) < f(v')$ if and only if $f'(v) < f'(v')$, $f(v) = f(v')$ if and only if $f'(v) = f'(v')$, and $f(v) > f(v')$ if and only if $f'(v) > f'(v')$.

The set $\{1, \ldots, |V|\}$ has cardinality $|V|$ and it can be computed in $O(|V| \cdot \log(|V|))$ time. In other words, $a_{|V|} = 1$, $b_{|V|} = |V|$, and t, u are polynomials. Theorem 9 gives that CSP($\mathcal{T}^{\vee\omega}$) can be solved in

$$\begin{aligned} &O(t(|V|) + a_{|V|} \cdot 2^{|V| \log b_{|V|}} \cdot u(\|I\|) \cdot poly(\|I\|)) \\ &= O(poly(|V|) + 1 \cdot 2^{|V| \log |V|} \cdot poly(\|I\|)) \\ &= O(2^{|V| \log |V|} \cdot poly(\|I\|)) \end{aligned}$$

since $|V| \leq \|I\|$. □

As our second example, we consider CSPs for *branching time* temporal reasoning. Here, we will use domain enumeration in a more substantial way that in the previous example. The branching time model has been used in, for instance, planning [23] and the analysis and verification of concurrent systems [27]. Let F be the forest containing all oriented, finite trees where the indegree of each node is at most one and let D_F be the nodes in F. We then define the following four relations on F. Arbitrarily choose $x, y \in D_F$.

1. $x =_F y$ if and only if there is a path from x to y and a path from y to x,
2. $x <_F y$ if and only if and there is a path from x to y but no path from y to x,
3. $x >_F y$ if and only if there is a path from y to x but no path from x to y, and
4. $x \|_F y$ if and only if there is no path from x to y and no path from y to x.

These four basic relations are known as the *point algebra for branching time*. We let $\mathcal{P} = \{=_F, <_F, >_F, \|_F\}$ and we note that $\mathcal{P}$ is JEPD. The problem CSP($\mathcal{P}^{\vee=}$) is in P [31] while the problem CSP($\mathcal{P}^{\vee\omega}$) is NP-complete [15].

Example 11. Let $I = (V, C)$ be an instance of CSP($\mathcal{P}^{\vee\omega}$) where $V = \{x_1, x_2, x_3, x_4, x_5\}$ and C contains the constraints

$$\{x_1 <_F x_4, x_5 \|_F x_4, x_3 \leq_F x_5, x_5 \leq_F x_3, x_2 \|_F x_5, x_1 <>_F x_2, x_1 <>_F x_5\},$$

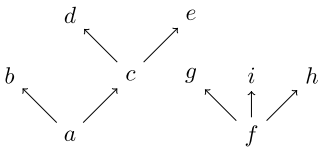

Fig. 2. The forest in Example 11.

where $x_i \leq_F x_j$ is an abbreviation of $(x_i <_F x_j) \vee (x_i =_F x_j)$ and $x_i <>_F x_j$ an abbreviation of $(x_i <_F x_j) \vee (x_i >_F x_j)$. This instance is satisfiable by e.g. the function $f(x_1) = a$, $f(x_2) = b$, $f(x_3) = d$, $f(x_4) = e$ and $f(x_5) = d$, where a, b, d, e are the points in the forest in Fig. 2. But if we let $f'(x) = f(x)$ for $x \in \{x_1, x_3, x_4, x_5\}$ and $f'(x_2) = g$, then f' is not satisfying assignment since the constraint $x_1 <>_F x_2$ is not satisfied by the partial order in Fig. 2.

From a formal viewpoint, we need to work with the structure F and view solutions as functions from variables to D_F. It is, however, quite impractical to work with the large and opaque structure F directly. It is easier to use the following observation: an instance (V, C) of CSP($\mathcal{P}^{\vee\omega}$) has a solution if and only if there exist an oriented forest T with the property that

1. the indegree of each node in T is at most one and
2. the number of nodes in T equals $|V|$,

such that the relations in C are satisfied by T (according to the interpretation of the basic relations given above). In particular, Theorem 9 is still applicable but we do not have to explicitly give unique names to all elements in D_F and invent algorithms that work with this representation.

We know from Theorem 7 that CSP($\mathcal{P}^{\vee\omega}$) can be solved in $O(2^{|V|^2 \cdot \log 4} \cdot poly(\|I\|)) = O(2^{2|V|^2} \cdot poly(\|I\|))$ time. We will now improve upon this result. Let $\tau(n)$ denote the number of unlabeled trees on n vertices. Otter [51] has shown that there exist constants C, α such that $\lim_{n\to\infty} \frac{\tau(n)}{C\alpha^n n^{-5/2}} = 1$ where $C > 0.53$ and $\alpha < 2.96$.

Theorem 12. CSP($\mathcal{P}^{\vee\omega}$) *is solvable in* $O(2^{|V|+\log(\tau(|V|))+|V|\log|V|} \cdot poly(\|I\|))$ *time.*

Proof. In this proof, we will utilize Theorem 9 so we need to define the constants $a_1, a_2, \ldots, b_1, b_2, \ldots$, the sets $S^n_1, \ldots, S^n_{a_n}$ for arbitrary n, and the functions t and u. We will use the alternative representation of solutions that we outlined after Example 11 so the sets $S^n_1, \ldots, S^n_{a_n}$ will be concrete forests and not subsets of D_F.

Given some $n > 0$, we first estimate the number of directed forests with n nodes where each node has indegree at most one. To enumerate all forests instead of trees, we can enumerate all unlabeled trees with $n+1$ vertices and only consider the trees where the extra vertex is connected to all other vertices. By removing this vertex we obtain a forest with n vertices (which implies that $b_n = n$). Hence, there are at most $2^n\tau(n+1)$ directed forests with n nodes. The factor 2^n stems from the observation that each forest contains at most n edges, where each edge has two possible directions. We then filter out the directed forests containing a tree where the indegree of any vertex is more than one, and we let $S^n_1, \ldots, S^n_{a_n}$ denote these forests. It follows that we can upper bound a_n with $2^n\tau(n+1)$.

Next, we need a way to compute the set of all directed forests where each node has indegree at most one. The only non-constructive argument above is the generation of all directed labeled trees with n nodes. However, these can be efficiently enumerated (with polynomial delay) as demonstrated by Wright et al. [66]. Thus, $t(n) = 2^n\tau(n+1) \cdot poly(n)$.

Finally, we need a way of checking whether a function $f : V \to S_i^{|V|}$ is a solution to an instance (V, C) of CSP($\mathcal{P}^{\vee\omega}$). Since $S_i^{|V|}$ is a forest, we can directly use the definitions of the basic relations in $\mathcal{P}$ when verifying this condition. This can be done in polynomial time so the function u is some polynomial.

Putting the pieces together with the aid of Theorem 9, we see that CSP($\mathcal{P}^{\vee\omega}$) is solvable in time

$$O(2^{|V|}\tau(|V|+1) \cdot poly(|V|) + 2^{|V|}\tau(|V|+1) \cdot 2^{|V|\log|V|} \cdot poly(\|I\|))$$
$$= O(2^{|V|+\log(\tau(|V|))+|V|\log|V|} \cdot poly(\|I\|)) \qquad \square$$

A simpler algorithm is obtained if we enumerate all labeled trees (by, for instance, using *Prüfer sequences* [53]) instead of the unlabeled trees. However, there are n^{n-2} such trees on n vertices according to Cayley's formula. This implies that the resulting algorithm runs in $O(2^{|V|+2|V|\log|V|} \cdot poly(\|I\|))$ time. This is substantially slower than the algorithm in Theorem 12 since $\log \tau|V| \leq (1+\epsilon)|V|$ (for arbitrary $\epsilon > 0$) when $|V|$ is sufficiently large.

4.2.3. Bounded disjunctions

This section contains a more efficient method for solving CSP($\mathcal{B}^{\vee k}$) when k is a fixed constant. In particular, such problems are interesting when studying finite constraint languages due to Lemma 2. The idea is to construct a number of k-SAT

instances with the property that at least one of them is satisfiable if and only if the original instance has a solution. More or less similar ideas have been used frequently in the literature and examples include algorithms for k-SAT [22], algorithms for combinatorial optimization [60, Sec. 8], and derandomization of probabilistic CSP algorithms [50]. One may also see certain similarities to the sparsification method that we presented in Section 3.2: sparsification is also based on the idea of transforming a single CSP instance into a set of CSP instances with advantageous properties. In the statement of the following theorem, let c_k denote an arbitrary real number $c_k < 1$ such that there exists a deterministic algorithm solving k-SAT in $O(2^{c_k \cdot |V|})$ time.

Theorem 13. *Let $\mathcal{B}$ be a set of basic relations with maximum arity p and $m = |\mathcal{B}|$. Assume that the following holds for every $n > 0$.*

1. *there exist finite sets $S_1^n, \ldots, S_{a_n}^n$ such that for every solvable instance $I = (V, C)$ of $\mathrm{CSP}(\mathcal{B})$, there exists a solution $f : V \to S_i^n$ for some $1 \leq i \leq a_n$ and*
2. *the set $\{S_i^n \mid 1 \leq i \leq n\}$ can be generated in $u(n)$ time.*

Let $b_i = \max\{|S_1^i|, \ldots, |S_{a_i}^i|\}$. Then $\mathrm{CSP}(\mathcal{B}^{\vee k})$ is solvable in $O(u(|V|) + a_{|V|} \cdot 2^{|V|(\log b_{|V|} - 1 + \log(c_{kp}))} \cdot poly(\|I\|))$ time.

Proof. Let $I = (V, C)$ be an arbitrary instance of $\mathrm{CSP}(\mathcal{B}^{\vee k})$. Assume $V = \{x_1, \ldots, x_s\}$. If I has a solution, then there is a solution $f : V \to S_i^{|V|}$ for some $1 \leq i \leq a_{|V|}$ by Condition (1). Thus, we begin by computing the set $\mathcal{S} = \{S_i^n \mid 1 \leq i \leq n\}$. This is possible due to Condition (2). We assume, without loss of generality, that $|S|$ is even for every $S \in \mathcal{S}$ and, for simplicity, we additionally assume that $S = \{1, \ldots, 2t\}$ for some $t \geq 1$. For each $S \in \mathcal{S}$, we construct a set of k-SAT instances $F_1, \ldots, F_p$ such that there exists (at least) one F_i that is satisfiable if and only if there is a solution $f : V \to S$ to I. We describe this construction next.

Arbitrarily choose a vector $\mathbf{z} = (z_1, \ldots, z_{|V|})$ where $z_i \in \{1, 3, 5, \ldots, 2t-1\}$, $1 \leq i \leq |V|$. We let $F_{\mathbf{z}}$ denote the k-SAT instance associated with the vector $\mathbf{z}$. The instance $F_{\mathbf{z}}$ contains variable set $V' = \{x'_1, \ldots, x'_s\}$ where we interpret variable x_i as follows: if x'_i is false, then variable x_i has value z_i and, otherwise, x_i has value $z_i + 1$. Arbitrarily choose a constraint in C. For simplicity, we assume that the constraint has maximal arity kp and that it equals $R(x_1, \ldots, x_{kp})$. For each tuple

$$\mathbf{r} \in \{z_1, z_1+1\} \times \{z_2, z_2+1\} \times \cdots \times \{z_{kp}, z_{kp}+1\}$$

that is not a member of the set

$$R \cap (\{z_1, z_1+1\} \times \{z_2, z_2+1\} \times \cdots \times \{z_{kp}, z_{kp}+1\}),$$

add the clause that 'forbids' this assignment to the variables, given the interpretation of variables described above. Note that this clause has arity kp, too. Do this for all constraints in C. It follows that F is satisfiable if and only if there exists a satisfying solution $f : V \to \{1, \ldots, 2t\}$ to I such that $f(x_1) \in \{z_1, z_1+1\}$, $f(x_2) \in \{z_2, z_2+1\}$, and so on.

By choosing all possible vectors $\mathbf{z}$, we end up with $(2t/2)^{|V|} = (b_{|V|}/2)^{|V|}$ kp-SAT instances such that at least one of them is satisfiable if and only if I has a solution. We need to verify the time complexity of this procedure. Note first that computing $F_{\mathbf{z}}$ can be done in polynomial time since the number of assignments that are forbidden by a constraint is at most 2^p, and p is a fixed constant. Finally, the time needed for verifying the satisfiability of $F_{\mathbf{a}}$ is $O(2^{c_{kp} \cdot |V|})$, and computing the set $\mathcal{S}$ takes $u(|V|)$ time due to condition (2). It follows that

$$\begin{aligned} u(|V|) + (b_{|V|}/2)^{|V|} \cdot 2^{c_{kp} \cdot |V|} &= u(|V|) + 2^{|V| \log(b_{|V|}/2)} \cdot 2^{|V| \log(c_{kp})} \\ &= u(|V|) + 2^{|V|(\log b_{|V|} - 1 + \log(c_{kp}))} \end{aligned}$$

which concludes the proof. □

The change in time complexity may seem minimal in comparison with Theorem 9. However, note that

$$2^{|V| \log b_{|V|}} = 2^{|V|} \cdot 2^{|V|(\log b_{|V|} - 1)}$$

so there is an exponential speed-up even if we do not take the negative term $\log c_{kp}$ into account. We remind the reader that the bounded length of disjunctions is vital for this method to work. If the length is unbounded, then there may be an exponential number of assignments that must be excluded by adding clauses to $F_{\mathbf{z}}$. This implies that the time needed for constructing $F_{\mathbf{z}}$ adds an exponential factor to the complexity figure in Theorem 13.

We will now turn our attention towards finite temporal constraint languages. Let us first consider total-ordered time. The computational complexity of such CSP problems has been intensively studied in the literature. In a breakthrough result, Bodirsky and Kára [13] have determined the complexity of $\mathrm{CSP}(\Gamma)$ for all such Γ and their result shows that $\mathrm{CSP}(\Gamma)$ is either tractable or NP-complete. It is well known that the first-order theory of $(\mathbb{Q}, <)$ admits quantifier elimination [13,33]. Hence, we can exploit Lemma 2 and Theorem 13 to obtain the following corollary.

Corollary 14. *Let Γ be a finite constraint language that is first-order definable in $(\mathbb{Q}, <)$. If $\mathrm{CSP}(\Gamma)$ is NP-complete, then it is solvable in time $O(2^{|V|(\log |V| - 1 - s_\Gamma)} \cdot poly(\|I\|))$ where $0 \leq s_\Gamma \leq 1$ is a constant that only depends on the choice of Γ. Otherwise, $\mathrm{CSP}(\Gamma)$ is polynomial-time solvable.*

Unfortunately, we cannot give a similar result for branching time since branching time does not admit quantifier elimination [8, Section 4.2] (so Lemma 2 is not applicable) and there is no complexity classification available. However, there are closely connected constraint languages on trees that have this property. Examples include the *triple consistency problem* with important applications in bioinformatics [14]: here we have both quantifier elimination and a complexity classification [11].

4.2.4. Improved domain enumeration

In the proof of Theorem 9, we compute a set $\{S_1, \ldots, S_n\}$ of finite variable domains and then consider *all* possible functions $V \to S_1$, $V \to S_2$, ..., $V \to S_n$. There are obviously cases where we do not need to enumerate all functions and this may lead to improved complexity figures. We demonstrate this by considering *equality* languages. An equality language is a set of relations definable through first-order formulas over the structure $(D, =)$. Such languages are of fundamental interest in complexity classifications for infinite domain CSPs, since a classification of CSP problems based on first-order definable relations over some fixed structure typically includes the classification of equality constraint language CSPs.

Let $\mathcal{E} = \{=, \neq\}$ over some countably infinite domain D. Note that $\mathcal{E}^{\vee\omega}$ is a sublanguage of $\mathcal{T}^{\vee\omega}$ so $\mathrm{CSP}(\mathcal{E}^{\vee\omega})$ can be solved in $O(2^{|V|\log|V|} \cdot poly(\|I\|))$ time by Theorem 10 (which, in turn, is based on Theorem 9). We will now improve upon this bound but first we need some additional machinery. A *partition* of a set X with n elements is a pairwise disjoint set $\{X_1, \ldots, X_m\}$, $m \leq n$ such that $\bigcup_{i=1}^{m} X_i = X$. A set X with n elements has B_n partitions, where B_n is the n-th *Bell number*. Let $L(n) = \frac{0.792n}{\ln(n+1)}$. It is known that $B_n < L(n)^n$ [5] and that all partitions can be enumerated in $O(nB_n)$ time [25,59].

Theorem 15. $\mathrm{CSP}(\mathcal{E}^{\vee\omega})$ *is solvable in* $O(|V|2^{|V|\cdot\log L(|V|)} \cdot poly(\|I\|))$ *time.*

Proof. Let $I = (V, C)$ be an instance of $\mathrm{CSP}(\mathcal{E}^{\vee\omega})$. For every partition $S_1 \cup \ldots \cup S_n$ of V we interpret the variables in S_i as being equal and having the value i, i.e. a constraint $(x = y)$ holds if and only if x and y belong to the same set and $(x \neq y)$ holds if and only if x and y belong to different sets. Then check in $poly(\|I\|)$ time if this partition satisfies I using the above interpretation. The complexity of this algorithm is therefore $O(|V|L_{|V|} \cdot poly(\|I\|)) \subseteq O(|V|L(|V|)^{|V|} \cdot poly(\|I\|)) = O(|V|2^{|V|\cdot\log L(|V|)} \cdot poly(\|I\|))$. □

The approach taken in Theorem 15 can be viewed as an opposite extreme of Theorem 9: here, we only consider *one* function per set of possible values.

It is well known that equality constraint languages admit quantifier elimination [12]. Hence, we can use Lemma 2 to extend Theorem 15 to cover arbitrary equality constraint languages.

Corollary 16. *Let* Γ *be a finite set of relations first-order definable over* $(D, =)$. *Then* $\mathrm{CSP}(\Gamma)$ *is solvable in* $O(|V|2^{|V|\cdot\log L(|V|)} \cdot poly(\|I\|))$ *time.*

Recall that $\mathcal{T}^{\vee k}$ (and consequently $\mathcal{E}^{\vee k}$) can be solved in time $O(2^{|V|(\log|V|-1-s_k)} \cdot poly(\|I\|))$ where $0 \leq s_k \leq 1$. This bound is beaten by Corollary 16 whenever $\log L(|V|) < (\log|V| - 1 - s_k)$ and this occurs even for fairly small $|V|$ since

$$\log L(|V|) \leq \log(0.792|V|) - \log(\ln|V|) \leq \log|V| - \log 1.26 - \log(\ln|V|).$$

5. Lower bounds

The algorithms presented in Section 4 give improved upper bounds (compared to the bounds given in Section 3) for many constraint satisfaction problems. It is natural to also ask, given reasonable complexity theoretical assumptions, how much room there is for improvement. Even though providing systematic lower bounds appears to be a challenging problem, non-trivial lower bounds can be given in certain cases. Such results are typically obtained by reducing a problem, which is believed to have a particular lower bound, to the problem in question. The reduction needs to have certain properties in order to be useful: basically, the reduction is not allowed to blow up the parameter that we are interested in too much. Since we measure time complexity in the number of variables, we need reductions that introduce only a small number of additional variables.

This section is divided into two parts. Section 5.1 contains lower bounds for $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ and $\mathrm{CSP}(\mathcal{B}^{\vee k})$ based on the (strong) exponential time hypothesis, and Section 5.2, where we obtain lower bounds for Allen's interval algebra based on the Chromatic Number problem.

5.1. Lower bounds based on (S)ETH

We begin by providing a general lower bound for $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ (Theorem 17) and we immediately observe (Corollary 18) that this reduction is useful for analyzing $\mathrm{CSP}(\mathcal{B}^{\vee k})$ when $k \geq 3$, too. We continue by refining our results in Theorem 19: if $\mathcal{B}$ is JEPD and contains the equality relation, then there is a stronger lower bound for $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ than the one given in Theorem 17. This result is *not* useful for studying $\mathrm{CSP}(\mathcal{B}^{\vee k})$ since it introduces disjunctive constraints with many disjuncts.

Theorem 17. *Let $\mathcal{B} = \{R_1, R_2, \ldots, R_m\}$, $m > 1$, be a set of nonempty p-ary basic relations such that $R_1 \cap R_2 = \emptyset$. If the SETH holds, then $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ cannot be solved in $O(2^{\delta|V|})$ time for any $\delta < 1$.*

Proof. If the SETH holds then SAT cannot be solved in $O(2^{\delta|V|})$ time for any $\delta < 1$. We provide a polynomial-time many-one reduction from SAT to $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ which only increases the number of variables by a constant (that only depends on the choice of $\mathcal{B}$) — hence, if $\mathrm{CSP}(\mathcal{B}^{\vee\omega})$ is solvable in $O(2^{\delta|V|})$ time for some $\delta < 1$ then SAT is also solvable in $O(2^{\delta|V|})$ time, contradicting the original assumption. We begin by constructing a useful gadget. Consider the following CSP instance:

$$I_1 : R_1(u_1, \ldots, u_p) \wedge R_2(v_1, \ldots, v_p).$$

This instance is satisfiable since both R_1 and R_2 are non-empty relations. Consider instead the instance

$$I_2 : R_1(z_1, u_2, \ldots, u_p) \wedge R_2(z_1, v_2, \ldots, v_p).$$

In this case, the instance can be either satisfiable or not satisfiable. If it is not satisfiable, then one may note that every solution f to instance I_1 has the property $f(u_1) \neq f(v_1)$. If I_2 is satisfiable, then we can continue the process of identifying variables until we reach a non-satisfiable instance

$$I_3 : R_1(\underbrace{z_1, \ldots, z_1}_{k \text{ times}}, u_{k+1}, \ldots, u_p) \wedge R_2(\underbrace{z_1, \ldots, z_1}_{k \text{ times}}, v_{k+1}, \ldots, v_p).$$

We thus have the following satisfiable instance

$$I_4 : R_1(\underbrace{z_1, \ldots, z_1}_{k-1 \text{ times}}, u_k, \ldots, u_p) \wedge R_2(\underbrace{z_1, \ldots, z_1}_{k-1 \text{ times}}, v_k, \ldots, v_p)$$

and we can continue the process of identifying variables by introducing a fresh variable z_2 and arrive at the instance

$$I_5 : R_1(\underbrace{z_1, \ldots, z_1}_{k-1 \text{ times}}, z_2, u_{k+1}, \ldots, u_p) \wedge R_2(\underbrace{z_1, \ldots, z_1}_{k-1 \text{ times}}, z_2, v_{k+1}, \ldots, v_p)$$

Just as in the case when we introduced z_1, this instance may or may not be satisfiable. If it is not satisfiable, then I_4 is satisfiable and every solution f satisfies $f(u_k) \neq f(v_k)$. Otherwise, we can continue the process described above. In the end, we will end up with a satisfiable instance

$$I_* : R_1(\underbrace{z_1, \ldots, z_1}_{k_1 \text{ times}}, \underbrace{z_2, \ldots, z_2}_{k_2 \text{ times}}, \ldots, \underbrace{z_m, \ldots, z_m}_{k_m \text{ times}}, y, u_{k_1+\cdots+k_m+2}, \ldots, u_p) \wedge$$
$$R_2(\underbrace{z_1, \ldots, z_1}_{k_1 \text{ times}}, \underbrace{z_2, \ldots, z_2}_{k_2 \text{ times}}, \ldots, \underbrace{z_m, \ldots, z_m}_{k_m \text{ times}}, y', v_{k_1+\cdots+k_m+2}, \ldots, v_p)$$

such that every solution f satisfies $f(y) \neq f(y')$. Note that the property PD guarantees that the process above will stop at some point since $R_1(\mathbf{x}) \wedge R_2(\mathbf{y})$ is not satisfiable when the variable vectors $\mathbf{x}$ and $\mathbf{y}$ are identical. We abbreviate the resulting instance $R_1(\mathbf{z}, y, \mathbf{u}) \wedge R_2(\mathbf{z}, y', \mathbf{v})$ and let $K = k_1 + \cdots + k_m + 2$. Let f_* be an arbitrary solution to I_*.

We are now ready to present the reduction. Let $I = (V, C)$ be an instance of SAT, where V is a set of variables and C a set of clauses. First observe that since $m \geq 2$ and since $\mathcal{B}$ is PD, $\mathcal{B}$ must be defined over a domain with two or more elements. Introduce the variables $z_1, \ldots, z_m, v_K, \ldots, v_p, u_K, \ldots, u_p$. Given a variable x, define

$$\phi(x) = R_1(\mathbf{z}, x, \mathbf{u})$$

and

$$\phi(\neg x) = R_2(\mathbf{z}, x, \mathbf{v}).$$

For every clause $(\ell_1 \vee \ldots \ell_k) \in C$, create the constraint $(\phi(\ell_1) \vee \ldots \vee \phi(\ell_k))$. We prove that the resulting instance J is satisfiable if and only if I is satisfiable.

Assume first that I has a solution $f : V \to \{0, 1\}$. We construct a solution $g : V \cup \{z_1, \ldots, z_m, v_K, \ldots, v_p, u_K, \ldots, u_p\} \to D$ to J as follows. First let $g(z_i) = f_*(z_i)$ $(1 \leq i \leq m)$, $g(u_i) = f_*(u_i)$ $(K \leq i \leq p)$, and $g(v_i) = f_*(v_i)$ $(K \leq i \leq p)$. Furthermore, let $g(x) = f_*(y)$ if $f(x) = 1$ and let $g(x) = f_*(y')$ if $f(x) = 0$.

Arbitrarily choose a clause $C = (\ell_1 \vee \cdots \vee \ell_m)$ in C and recall that there is a corresponding constraint $C' = (\phi(\ell_1) \vee \cdots \vee \phi(\ell_m))$ in J. Assume without loss of generality that ℓ_1 is satisfied by f. If $\ell_1 = x_1$, then $f(x_1) = 1$ and the corresponding relation in C' is $R_1(\mathbf{z}, x, \mathbf{u})$. Note that this relation (and thus the constraint C') is indeed satisfied by g, If $\ell_1 = \neg x_1$, then $f(x_1) = 0$ and the corresponding relation in C' is $R_2(\mathbf{z}, x, \mathbf{v})$. Once again, the constraint C' is satisfied by g, and J is satisfiable.

Assume instead that J has a solution f. This solution makes at least one disjunct in each constraint satisfied so we let the set S contain exactly one satisfied disjunct from each constraint. The set S cannot simultaneously contain the constraints $R_1(\mathbf{z}, x, \mathbf{u})$ and $R_1(\mathbf{z}, x, \mathbf{v})$ for any variable $x \in V$. Thus we can construct a solution g for I as follows: if $R_1(\mathbf{z}, x, \mathbf{u}) \in S$, then

$g(x) = 1$ and $g(x) = 0$ otherwise. Since there is a one-to-one correspondence between clauses in I and the disjunctive constraints in J, it follows that g must be a satisfying assignment to I. □

If the SAT instance I in the proof of Theorem 17 has clauses of length at most k, then the resulting CSP instance J is an instance of CSP($\mathcal{B}^{\vee k}$). The ETH immediately gives us the following result.

Corollary 18. *Let $\mathcal{B} = \{R_1, R_2, \ldots, R_m\}$, $m > 1$, be a set of nonempty p-ary basic relations such that $R_1 \cap R_2 = \emptyset$. If the ETH holds, then* CSP($\mathcal{B}^{\vee k}$), $k \geq 3$, *cannot be solved in $O(2^{\delta_k |V|})$ time.*

Theorem 17 and Corollary 18 have a wide range of applicability since the only restriction on the set $\mathcal{B}$ is that it contains two distinct relations R_1, R_2 such that $R_1 \cap R_2 = \emptyset$. There are significantly better lower bounds if we impose additional restrictions on the set $\mathcal{B}$. By assuming that the relations are JEPD and that we have access to $=$ (as usual, we let $=$ denote the equality relation on a given domain), we can view several variables as a single variable and thus obtain stronger lower bounds. Similar techniques have been used when proving lower bounds by, for instance, Traxler [61] and Gutin and Wahlström [30].

Theorem 19. *Let $\mathcal{B} = \{=, R_1, \ldots, R_m\}$ be a set of binary JEPD relations over a countably infinite domain. If the SETH holds, then* CSP($\mathcal{B}^{\vee\omega}$) *cannot be solved in $O(c^{|V|})$ time for any $c > 1$.*

Proof. First observe that the binary inequality relation $\neq$ over D can be defined as $\bigcup_{i=1}^{m} R_i$ since $\mathcal{B}$ is JEPD. In the proof we therefore use $\neq$ as an abbreviation for $\bigcup_{i=1}^{m} R_i$. Let $I = (V, C)$ be an instance of SAT with variables $V = \{x_1, \ldots, x_n\}$ and the set of clauses C. Let K be an integer such that $K > \log c$. Assume without loss of generality that n is a multiple of K. We will construct an instance of CSP($\mathcal{B}^{\vee\omega}$) with $\frac{n}{K} + 2^K = \frac{n}{K} + O(1)$ variables. First, introduce 2^K fresh variables $v_1, \ldots, v_{2^K}$ and make them different by imposing $\neq$ constraints. Second, introduce $\frac{n}{K}$ fresh variables $y_1, \ldots, y_{\frac{n}{K}}$, and for each $i \in \{1, \ldots, \frac{n}{K}\}$ impose the constraint

$$(y_i = v_1 \vee y_i = v_2 \vee \cdots \vee y_i = v_{2^k}).$$

Let $V_1, \ldots, V_{\frac{n}{K}}$ be a partition of V such that each $|V_i| = K$. We will represent each set V_i of Boolean variables by one y_i variable over D. To do this we will interpret each auxiliary variable z_i as a K-ary Boolean tuple. Let $h : \{v_1, \ldots, v_{2^K}\} \to \{0, 1\}^K$ be an injective function which assigns a Boolean K-tuple for every variable v_i. Let g_+ be a function from $\{1, \ldots, K\}$ to subsets of $\{v_1, \ldots, v_{2^K}\}$ such that $v_i \in g(j)$ if and only if the j-th element in $h(v_i)$ is equal to 1. Define g_- in the analogous way. Observe that $|g_+(j)| = |g_-(j)| = 2^{K-1}$ for each $j \in \{1, \ldots, K\}$.

For the reduction, let $(\ell_{i_1} \vee \ldots \vee \ell_{i_{n'}})$, $\ell_{i_j} = x_{i_j}$ or $\ell_{i_j} = \neg x_{i_j}$, be a clause in C. We assume that $n' \leq n$ since the clause contains repeated literals otherwise. For each literal ℓ_{i_j} let $V_{i'} \subseteq V$ be the set of variables such that $x_{i_j} \in V_{i'}$. Each literal ℓ_{i_j} is then replaced by

$$\bigvee_{z \in g_+(i_j)} y_{i'} = z$$

if $\ell_{i_j} = x_{i_j}$, and with

$$\bigvee_{z \in g_-(i_j)} y_{i'} = z$$

if $\ell_{i_j} = \neg x_{i_j}$. This reduction can be done in polynomial time since a clause with n' literals is replaced by a disjunctive constraint with $n' 2^{K-1}$ disjuncts (since K is a constant depending only on c). It follows that SAT can be solved in

$$O(c^{\frac{n}{K} + O(1)} \cdot poly(\|I\|)) = O(2^{(\frac{n}{K} + O(1)) \cdot \log c} \cdot poly(\|I\|)) = O(2^{\delta \cdot n} \cdot poly(\|I\|))$$

for some $\delta < 1$, since $K > \log c$. Thus, the SETH does not hold. □

As an illustrative use of the theorem we see that the temporal problem CSP($\mathcal{T}^{\vee\omega}$) is solvable in $O(2^{|V| \log |V|} \cdot poly(\|I\|))$ time but not in $O(c^{|V|})$ time for any $c > 1$ if the SETH holds. Lower bounds can also be obtained for the branching time problem CSP($\mathcal{P}^{\vee\omega}$) since there is a trivial reduction from CSP($\mathcal{T}$)$^{\vee\omega}$ which does not increase the number of variables: simply add a constraint $(x < y \vee x > y \vee x = y)$ for every pair of variables in the instance. Similarly, the equality constraint satisfaction problem CSP($\mathcal{E}^{\vee\omega}$) is not solvable in $O(c^{|V|})$ time for any $c > 1$ either, unless the SETH fails. Hence, even though the algorithms that were presented in Section 4 might appear to be quite simple, there is very little room for improvement.

Table 1
The thirteen basic relations in Allen's interval algebra. The endpoint relations $x^s < x^e$ and $y^s < y^e$ that are valid for all relations have been omitted.

Basic relation		Example	Endpoints
x precedes y	p	xxx	$x^e < y^s$
y preceded by x	p^{-1}	yyy	
x meets y	m	xxxx	$x^e = y^s$
y met-by x	m^{-1}	yyyy	
x overlaps y	o	xxxx	$x^s < y^s < x^e$,
y overl.-by x	o^{-1}	yyyy	$x^e < y^e$
x during y	d	xxx	$x^s > y^s$,
y includes x	d^{-1}	yyyyyyy	$x^e < y^e$
x starts y	s	xxx	$x^s = y^s$,
y started by x	s^{-1}	yyyyyyy	$x^e < y^e$
x finishes y	f	xxx	$x^e = y^e$,
y finished by x	f^{-1}	yyyyyyy	$x^s > y^s$
x equals y	$\equiv$	xxxx	$x^s = y^s$,
		yyyy	$x^e = y^e$

5.2. Lower bounds based on Chromatic Number

The results in Section 5.1 show that the (S)ETH can be used for obtaining lower bounds for problems such as CSP($\mathcal{B}^{\vee\omega}$) and CSP($\mathcal{B}^{\vee k}$). Unfortunately, it is not obvious how to obtain lower bounds for CSP($\mathcal{B}^{\vee=}$) using this assumption. In this section, we present lower bounds for *Allen's interval algebra* using a conjecture concerning the time complexity of computing the chromatic number of graphs. The bound will not be as strong as the ones obtained by using the (S)ETH and it does not seem to (easily) generalize to other CSP($\mathcal{B}^{\vee=}$) problems.

We first recapitulate the basics of Allen's interval algebra. Allen's algebra is a well-known formalism for temporal reasoning where one considers relations between intervals of the form $[x, y]$, where $x, y \in \mathbb{R}$ is the starting and ending point, respectively. Let *Allen* be the $2^{13} = 8192$ possible unions of the set of the thirteen relations in Table 1. For convenience we write constraints such as $(\mathsf{p} \vee \mathsf{m})(x, y)$ as $x\{\mathsf{p}, \mathsf{m}\}y$, using infix notation and omitting explicit disjunction signs. The problem CSP(*Allen*) is NP-complete and all tractable fragments have been identified [44].

Given an instance $I = (V, C)$ of CSP(*Allen*) we first create two fresh variables x_i^s and x_i^e for every $x \in V$, intended to represent the startpoint and endpoint of the interval x. Then observe that a constraint $x\{r_1, \ldots, r_m\}y \in C$, where each r_i is a basic relation, can be represented as a disjunction of temporal constraints over x^s, x^e, y^s and y^e by using the definitions of each basic relation in Table 1. Applying Theorem 10 to the resulting instance gives the following result.

Corollary 20. CSP(*Allen*) *is solvable in* $O(2^{2|V|(1+\log|V|)} \cdot poly(\|I\|))$ *time.*

We will now relate CSP(*Allen*) to the Chromatic Number problem, i.e. the problem of computing the number of colors needed to color a given graph.

Theorem 21. *If* CSP(*Allen*) *can be solved in* $O(\sqrt{c}^{|V|})$ *time for some* $c < 2$*, then* Chromatic Number *can be solved in* $O((c+\epsilon)^{|V|})$ *time for arbitrary* $\epsilon > 0$.

Proof. We first present a polynomial-time many-one reduction from k-Colourability to CSP(*Allen*) which introduces k fresh variables. Given an undirected graph $G = (\{v_1, \ldots, v_n\}, E)$, introduce the variables $z_1, \ldots, z_k$ and $v_1, \ldots, v_n$, and:

1. impose the constraints $z_1\{\mathsf{m}\}z_2\{\mathsf{m}\}\ldots\{\mathsf{m}\}z_k$,
2. for each v_i, $1 \le i \le n$, add the constraints $v_i\{\equiv, \mathsf{s}^{-1}\}z_1$, $v_i\{\mathsf{p}, \mathsf{m}, \mathsf{f}^{-1}, \mathsf{d}^{-1}\}z_j$ $(2 \le j \le k-1)$, and $v_i\{\mathsf{p}, \mathsf{m}, \mathsf{f}^{-1}\}z_k$,
3. for each $(v_i, v_j) \in E$, add the constraint $v_i\{\mathsf{s}, \mathsf{s}^{-1}\}v_j$.

Consulting Table 1, we see that for each v_i, it holds that its right endpoint must equal the right endpoint of some z_i, and its left endpoint must equal the left endpoint of z_1. Thus, there are exactly k possible choices for the right endpoint of v_i. If there is an edge (v_i, v_j), then we have the constraint $v_i\{\mathsf{s}, \mathsf{s}^{-1}\}v_j$ which ensures that the right endpoints of the corresponding variables differ. It follows that the resulting instance has a solution if and only if G is k-colorable. Hence, there is a polynomial-time Turing reduction from Chromatic Number to CSP(*Allen*) by combining binary search (that will evaluate $\log n$ Allen instances) with the reduction above (recall that $O(\log n \cdot c^n) \subseteq O((c+\epsilon)^n)$ for every $\epsilon > 0$). Observe that if $k = n$ then the reduction introduces n fresh variables, which is where the constant $\sqrt{c}$ in the expression $O(\sqrt{c}^{|V|})$ stems from. □

The exact complexity of Chromatic Number has been analyzed and discussed in the literature. Björklund et al. [6] have shown that the problem is solvable in $2^{|V|} \cdot poly(\|I\|)$ time. Impagliazzo and Paturi [35] poses the following question: 'Assuming SETH, can we prove a $2^{n-o(n)}$ lower bound for Chromatic Number?'. Hence, an $O(\sqrt{c}^{|V|})$, $c < 2$, algorithm for CSP(*Allen*) would also be a major breakthrough for Chromatic Number.

6. Research directions

The study of infinite-domain CSP time complexity is still in its infancy, and there is a large amount of open questions that need to be addressed. We present a small selection below.

Analysis of algorithms. We have investigated several novel algorithms for solving disjunctive CSP problems, which, with respect to worst-case time complexity, are much more efficient than e.g. backtracking algorithms without heuristics. These bounds can likely be improved, but, due to the lower bounds in Section 5, probably not to a great degree. Despite this, algorithms for solving infinite domain constraint satisfaction problems are in practice used in many non-trivial applications. In light of this the following research direction is particularly interesting: *how to formally analyze the time complexity of branching algorithms equipped with (powerful) heuristics?* In the case of finite-domain CSPs and, in particular, DPLL-like algorithms for the k-SAT problem there are numerous results to be found in the literature, cf. the survey by Vsemirnov et al. [64]. This is not the case for infinite-domain CSPs, even though there is a considerable amount of empirical evidence that infinite-domain CSPs can be efficiently solved by such algorithms, so one ought to be optimistic about the chances of actually obtaining non-trivial bounds. Yet, sharp formal analyses appear to be virtually nonexistent in the literature.

Quantified constraint satisfaction. In our article, we have limited ourselves to infinite-domain CSP problems included in NP. A natural generalization of the CSP problem is to consider instances which are allowed to also contain universally quantified variables, in addition to existentially quantified variables. This problem is in general known as the *quantified constraint satisfaction problem* (QCSP). For finite domains this problem is included in PSPACE, and this is also known to hold for many well-studied languages over infinite domains. For example, QCSP problems over equality constraint languages and temporal constraint languages are in general PSPACE-complete [9]. Would it be possible to exploit the algorithms for CSP($\mathcal{E}^{\vee\omega}$) and CSP($\mathcal{T}^{\vee\omega}$) in order to obtain upper bounds for their QCSP counterparts?

Upper bounds. A natural step is to obtain upper bounds for spatial formalisms such as RCC-5 or RCC-8. We have encountered structure enumeration for formalisms with binary basic relations $\mathcal{B}$ several times during the course of the article: the problem CSP($\mathcal{B}^{\vee\omega}$) can be solved in $c^{|V|^2}$ where the constant c depends on $\mathcal{B}$. This bound is clearly applicable to RCC-5 and RCC-8 and it raises the question whether domain enumeration may lead to improved algorithms or not. A starting point may be to analyze the complexity of the point algebra for *partially ordered* time since the relations in this algebra is expressible in both RCC-5 and RCC-8. One possibility here is to construct an algorithm based on the algorithm for branching time in Section 4.2.2. It is known that there are slightly more than $2^{n^2/4}$ partial orders on n nodes [52]. Thus, this approach will not immediately lead to a significantly faster algorithm than the $c^{|V|^2}$ time algorithm based on structure enumeration, even if we have a polynomial delay algorithm for enumerating partial orders. It may, though, exist a domain enumeration algorithm running in $(c')^{|V|^2}$ time with $c' < c$. Such a speed-up may still be considered important.

Lower bounds. Another obvious research direction is to strengthen the lower bounds in Section 5 even further. The probably most challenging problem here is to obtain stronger lower bounds for CSP($\mathcal{B}^{\vee=}$). It appears that the (strong) exponential time hypothesis is not so useful since the use of disjunctions seems essential. We have seen that the conjecture for Chromatic Number was useful when studying Allen's algebra, even though the achieved bound is weaker than those obtained for more expressive disjunctive constraints. Unfortunately, it is not obvious how to generalize the Allen result to other problems of the type CSP($\mathcal{B}^{\vee=}$).

It would also be interesting to prove stronger lower bounds for CSP($\mathcal{B}^{\vee k}$) for some concrete choices of $\mathcal{B}$ and k. As an example, consider the temporal problem CSP($\mathcal{T}^{\vee 4}$). From Corollary 18 we see that CSP($\mathcal{T}^{\vee 4}$) is not solvable in $O(2^{s_4|V|})$ time for some $s_4 < \log 1.6$, assuming the ETH holds, since the currently best deterministic algorithm for 4-SAT runs in $O(1.6^{|V|})$ time [57]. On the other hand, if CSP($\mathcal{T}^{\vee 4}$) is solvable in $O(\sqrt{c}^{|V|})$ time for some $c < 2$, then Chromatic Number can be solved in $O((c+\epsilon)^{|V|})$ time for arbitrary $\epsilon > 0$. This can be proven similar to the reduction in Theorem 21 but by making use of temporal constraints instead of interval constraints. Hence, for certain choices of $\mathcal{B}$ and k it might be possible to improve upon the general bounds given in Section 5.

Acknowledgements

Peter Jonsson is partially supported by Swedish Research Council (VR) under Grant 621-2012-3239. Victor Lagerkvist is supported by the National Graduate School in Computer Science (CUGS), Sweden, and the DFG-funded project "Homogene Strukturen, Bedingungserfüllungsprobleme, und topologische Klone" (Project number 622397).

References

[1] N. Alon, A. Shpilka, C. Umans, On sunflowers and matrix multiplication, Comput. Complex. 22 (2) (2013) 219–243.

[2] C. Bäckström, P. Jonsson, All PSPACE-complete planning problems are equal but some are more equal than others, in: Proceedings of the 4th Annual Symposium on Combinatorial Search (SOCS-2011), 2011.
[3] C. Bäckström, P. Jonsson, A refined view of causal graphs and component sizes: SP-closed graph classes and beyond, J. Artif. Intell. Res. 47 (2013) 575–611.
[4] B. Bauslaugh, The complexity of infinite H-coloring, J. Comb. Theory, Ser. B 61 (2) (1994) 141–154.
[5] D. Berend, T. Tassa, Improved bounds on Bell numbers and on moments of sums of random variables, Probab. Math. Stat. 30 (2) (2010) 185–205.
[6] A. Björklund, T. Husfeldt, M. Koivisto, Set partitioning via inclusion–exclusion, SIAM J. Comput. 39 (2) (2009) 546–563.
[7] M. Bodirsky, Constraint satisfaction problems with infinite templates, in: N. Creignou, P.G. Kolaitis, H. Vollmer (Eds.), Complexity of Constraints, in: Lecture Notes in Computer Science, vol. 5250, Springer, Berlin, Heidelberg, 2008, pp. 196–228.
[8] M. Bodirsky, Complexity classification in infinite-domain constraint satisfaction, Mémoire d'habilitation à diriger des recherches, Université Diderot – Paris 7, 2012, Available at arXiv:1201.0856.
[9] M. Bodirsky, H. Chen, Collapsibility in infinite-domain quantified constraint satisfaction, in: Proceedings of Computer Science Logic 2006 (CSL 2006), Springer, Berlin, Heidelberg, 2006, pp. 197–211.
[10] M. Bodirsky, M. Hils, Tractable set constraints, J. Artif. Intell. Res. 45 (2012) 731–759.
[11] M. Bodirsky, P. Jonsson, T.V. Pham, The complexity of phylogeny constraint satisfaction, in: 33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17–20, 2016, Orléans, France, 2016, pp. 20:1–20:13.
[12] M. Bodirsky, J. Kára, The complexity of equality constraint languages, Theory Comput. Syst. 43 (2) (2008) 136–158.
[13] M. Bodirsky, J. Kára, The complexity of temporal constraint satisfaction problems, J. ACM 57 (2) (2010) 9:1–9:41.
[14] M. Bodirsky, J.K. Mueller, The complexity of rooted phylogeny problems, Log. Methods Comput. Sci. 7 (4) (2011).
[15] M. Broxvall, P. Jonsson, Point algebras for temporal reasoning: algorithms and complexity, Artif. Intell. 149 (2) (2003) 179–220.
[16] M. Broxvall, P. Jonsson, J. Renz, Disjunctions, independence, refinements, Artif. Intell. 140 (1/2) (2002) 153–173.
[17] C. Calabro, The Exponential Complexity of Satisfiability Problems, PhD thesis, University of California, San Diego, CA, USA, 2009.
[18] J. Chen, B. Chor, M. Fellows, X. Huang, D.W. Juedes, I.A. Kanj, G. Xia, Tight lower bounds for certain parameterized NP-hard problems, Inf. Comput. 201 (2) (2005) 216–231.
[19] J. Chen, X. Huang, I. Kanj, G. Xia, Strong computational lower bounds via parameterized complexity, J. Comput. Syst. Sci. 72 (8) (2006) 1346–1367.
[20] Y. Chen, M. Grohe, An isomorphism between subexponential and parameterized complexity theory, SIAM J. Comput. 37 (4) (2007) 1228–1258.
[21] D. Cohen, P. Jeavons, P. Jonsson, M. Koubarakis, Building tractable disjunctive constraints, J. ACM 47 (5) (2000) 826–853.
[22] E. Dantsin, A. Goerdt, E. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, O. Raghavan, U. Schöning, A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search, Theor. Comput. Sci. 289 (2002) 69–83.
[23] T.L. Dean, M.S. Boddy, Reasoning about partially ordered events, Artif. Intell. 36 (3) (1988) 375–399.
[24] R. Dechter, Constraint Processing, Elsevier/Morgan Kaufmann, 2003.
[25] B. Djokic, M. Miyakawa, S. Sekiguchi, I. Semba, I. Stojmenovic, A fast iterative algorithm for generating set partitions, Comput. J. 32 (3) (June 1989) 281–282.
[26] T. Drakengren, M. Bjareland, Reasoning about action in polynomial time, Artif. Intell. 115 (1) (1999) 1–24.
[27] E. Emerson, J. Srinivasan, Branching time temporal logic, in: J. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.), Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Springer-Verlag, New York, 1989, pp. 123–172.
[28] J. Flum, M. Grohe, Parameterized Complexity Theory, Texts in Theoretical Computer Science, An EATCS Series, Springer, 2006.
[29] S. Gaspers, Exponential Time Algorithms – Structures, Measures, and Bounds, VDM, 2010.
[30] G. Gutin, M. Wahlström, Tight lower bounds for the workflow satisfiability problem based on the strong exponential time hypothesis, Inf. Process. Lett. 116 (3) (2016) 223–226.
[31] R. Hirsch, Expressive power and complexity in algebraic logic, J. Log. Comput. 7 (3) (1997) 309–351.
[32] R. Hirsch, A finite relation algebra with undecidable network satisfaction problem, Bull. Interest Group Pure Appl. Log. 7 (4) (1999) 547–554.
[33] W. Hodges, A Shorter Model Theory, Cambridge University Press, New York, NY, USA, 1997.
[34] R. Impagliazzo, R. Paturi, On the complexity of k-SAT, J. Comput. Syst. Sci. 62 (2) (2001) 367–375.
[35] R. Impagliazzo, R. Paturi, Exact complexity and satisfiability, in: G. Gutin, S. Szeider (Eds.), Parameterized and Exact Computation, in: Lecture Notes in Computer Science, vol. 8246, Springer International Publishing, 2013, pp. 1–3.
[36] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity?, J. Comput. Syst. Sci. 63 (4) (2001) 512–530.
[37] P. Jonsson, C. Bäckström, A unifying approach to temporal constraint reasoning, Artif. Intell. 102 (1) (1998) 143–155.
[38] P. Jonsson, V. Lagerkvist, Upper and lower bounds on the time complexity of infinite-domain CSPs, in: Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP-2015), 2015, pp. 183–199.
[39] P. Jonsson, V. Lagerkvist, G. Nordh, B. Zanuttini, Complexity of SAT problems, clone theory and the exponential time hypothesis, in: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2013), 2013, pp. 1264–1277.
[40] S. Jukna, Extremal Combinatorics – With Applications in Computer Science, Texts in Theoretical Computer Science, Springer, 2001.
[41] P.C. Kanellakis, G.M. Kuper, P.Z. Revesz, Constraint query languages, J. Comput. Syst. Sci. 51 (1) (1995) 26–52.
[42] I. Kanj, S. Szeider, On the subexponential time complexity of CSP, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-2013), 2013.
[43] M. Koubarakis, Dense time and temporal constraints with $\neq$, in: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), 1992, pp. 24–35.
[44] A. Krokhin, P. Jeavons, P. Jonsson, Reasoning about temporal relations: the tractable subalgebras of Allen's interval algebra, J. ACM 50 (5) (September 2003) 591–640.
[45] G. Ligozat, Qualitative Spatial and Temporal Reasoning, Wiley-ISTE, 2011.
[46] B. Liu, J. Jaffar, Using constraints to model disjunctions in rule-based reasoning, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI-96, Portland, Oregon, August 4–8, 1996, vol. 2, 1996, pp. 1248–1255.
[47] D. Lokshtanov, D. Marx, S. Saurabh, Lower bounds based on the exponential time hypothesis, Bull. Eur. Assoc. Theor. Comput. Sci. 3 (105) (2013).
[48] K. Marriott, P. Moulder, P.J. Stuckey, A. Borning, Solving disjunctive constraints for interactive graphical applications, in: Proc. 7th International Conference on Principles and Practice of Constraint Programming (CP-2001), 2001, pp. 361–376.
[49] D. Marx, On the optimality of planar and geometric approximation schemes, in: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS-2007), 2007, pp. 338–348.
[50] R.A. Moser, D. Scheder, A full derandomization of Schöning's k-SAT algorithm, in: Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC-2011), 2011, pp. 245–252.
[51] R. Otter, The number of trees, Ann. Math. 49 (3) (1948) 583–599.
[52] H.J. Prömel, Counting unlabeled structures, J. Comb. Theory, Ser. A 44 (1987) 83–93.
[53] H. Prüfer, Neuer beweis eines satzes über permutationen, Arch. Math. Phys. 27 (1918) 742–744.
[54] J. Renz, B. Nebel, Efficient methods for qualitative spatial reasoning, J. Artif. Intell. Res. 15 (1) (2001) 289–318.

[55] F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006.
[56] R. Santhanam, S. Srinivasan, On the limits of sparsification, in: Proceeding of the 39th International Colloquium on Automata, Languages, and Programming (ICALP-2012), 2012, pp. 774–785.
[57] U. Schöning, A probabilistic algorithm for k-SAT based on limited local search and restart, Algorithmica 32 (4) (2002) 615–623.
[58] K. Stergiou, M. Koubarakis, Backtracking algorithms for disjunctions of temporal constraints, Artif. Intell. 120 (1) (2000) 81–117.
[59] I. Stojmenović, An optimal algorithm for generating equivalence relations on a linear array of processors, BIT Numer. Math. 30 (3) (1990) 424–436.
[60] J. Thapper, Aspects of a Constraint Optimization Problem, PhD thesis, Linköping University, Linköping, Sweden, 2010.
[61] P. Traxler, The time complexity of constraint satisfaction, in: Proceeding of the Third International Workshop on Parameterized and Exact Computation (IWPEC-2008), 2008, pp. 190–201.
[62] P. van Beek, Approximation algorithms for temporal reasoning, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-1989), 1989, pp. 1291–1296.
[63] M.B. Vilain, H.A. Kautz, Constraint propagation algorithms for temporal reasoning, in: Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86), 1986, pp. 377–382.
[64] M. Vsemirnov, E. Hirsch, E. Dantsin, S. Ivanov, Algorithms for SAT and upper bounds on their complexity, J. Math. Sci. 118 (2) (2003) 4948–4962.
[65] G. Woeginger, Exact algorithms for NP-hard problems: a survey, in: M. Juenger, G. Reinelt, G. Rinaldi (Eds.), Combinatorial Optimization – Eureka! You Shrink!, 2000, pp. 185–207.
[66] R. Alan Wright, L. Bruce Richmond, A.M. Odlyzko, B.D. McKay, Constant time generation of free trees, J. Comput. Syst. Sci. 15 (2) (1986) 540–548.