Local consistency as a reduction between constraint satisfaction problems

Víctor Dalmau
victor.dalmau@upf.edu
Universitat Pompeu Fabra
Barcelona, Spain

Jakub Opršal
j.oprsal@bham.ac.uk
University of Birmingham
Birmingham, UK

ABSTRACT

We study the use of local consistency methods as reductions between constraint satisfaction problems (CSPs), and promise version thereof, with the aim to classify these reductions in a similar way as the algebraic approach classifies gadget reductions between CSPs. This research is motivated by the requirement of more expressive reductions in the scope of promise CSPs. While gadget reductions are enough to provide all necessary hardness in the scope of (finite domain) non-promise CSP, in promise CSPs a wider class of reductions needs to be used.

We provide a general framework of reductions, which we call *consistency reductions*, that covers most (if not all) reductions recently used for proving NP-hardness of promise CSPs. We prove some basic properties of these reductions, and provide the first steps towards understanding the power of consistency reductions by characterizing a fragment associated to arc-consistency in terms of polymorphisms of the template. In addition to showing hardness, consistency reductions can also be used to provide feasible algorithms by reducing to a fixed tractable (promise) CSP, for example, to solving systems of affine equations. In this direction, among other results, we describe the well-known Sherali–Adams hierarchy for CSP in terms of a consistency reduction to linear programming.

CCS CONCEPTS

• **Theory of computation → Constraint and logic programming**; **Problems, reductions and completeness**.

KEYWORDS

constraint satisfaction problem, Datalog, Karp reduction, polymorphism

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No 101034413. Jakub Opršal was also supported by the UK EPSRC grant EP/R034516/1. Victor Dalmau was supported by the MICIN under grants PID2019-109137GB-C22 and PID2022-138506NB-C22, and the Maria de Maeztu program (CEX2021-001195-M)..

LICS '24, July 8–11, 2024, Tallinn, Estonia

ACM ISBN 979-8-4007-0660-8/24/07
https://doi.org/10.1145/3661814.3662068

ACM Reference Format:
Víctor Dalmau and Jakub Opršal. 2024. Local consistency as a reduction between constraint satisfaction problems. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24), July 8–11, 2024, Tallinn, Estonia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3661814.3662068

ACKNOWLEDGMENTS

A part of this paper is based on an unpublished note by Marcin Wrochna [51] which contains the characterisation of the arc-consistency reduction and several observations and statements that served as a ground for research presented in this paper.

The second author would also like to thank Anuj Dawar, Joanna Ochremiak, Adam Ó Conghaile, Antoine Mottet, and Manuel Bodirsky for inspiring discussions. We would also like to thank the reviewers of this paper for careful reading and their insightful remarks, and all who read a previous version of this paper and provided useful feedback.

1 INTRODUCTION

Is it possible to find a mathematical invariant that characterises when one computational problem reduces to another by a polynomial-time *(Karp) reduction*? This question is far beyond our current understanding as it would imply a characterisation of polynomial-time solvable problems, possibly resolving the P vs. NP problem. The present paper is a contribution to the effort to provide a partial answer to this question by characterising certain *well-structured efficient reductions* between structured problems.

A characterisation of a subclass of log-space reductions between *constraint satisfaction problems (CSPs)* is the core of a theory referred to as *the algebraic approach* to the CSP. The development of this theory started with the work of Jeavons, Cohen, and Gyssens [35]; Bulatov, Jeavons, and Krokhin [19], and after 20 years of active research provided the celebrated *dichotomy theorem* proven by Bulatov [20] and Zhuk [53] after being conjectured by Feder and Vardi [29]. The scope of the theory and the dichotomy theorem are finite-template CSPs. Finite-template CSPs can be formulated as deciding whether there is a homomorphism from a given structure to a fixed finite (relational) structure (called *template*), or alternatively as deciding whether the template satisfies a primitive-positive sentence; the atoms occurring in this sentence are called *constraints*. By varying the template one can encode a large family of computational problems, including SAT (and its variants), graph k-colouring, or solving systems of linear equations over a finite field. Many of these problems naturally arise in various settings. A general theorem [19] characterises when one such problem can be reduced to another using a simple *gadget reduction* by the means of the *polymorphisms* of the template (see also Theorem 2.4 below). In the context of CSPs

and this paper, gadget reductions are those reductions that replace each constraint of the input instance with a 'gadget' consisting of several constraints over the output language (see Section 2.3). This is a special case of a more general concept in complexity theory where one only requires that every bit of the output depends on a bounded number of bits of the input. The dichotomy theorem can be formulated as follows: For each finite template, either its CSP can be proven to be NP-complete via a gadget reduction from graph 3-colouring [19], or there is a polynomial time algorithm solving it [20; 53].

The motivation of the present paper stems from two failures of gadget reductions. Firstly, their inability to sufficiently explain the tractability side of the dichotomy; we would like to see that all tractable CSPs (assuming P ≠ NP) reduce to a small class of problems that are tractable for an obvious reason (e.g., solving systems of linear equations over finite fields). This is not the case for gadget reductions, and a lot of effort has been put into understanding the hardest (w.r.t. gadget reductions) tractable CSPs (see, e.g., Barto, Brady, Bulatov, Kozik, and Zhuk [6], or Barto, Bodor, Kozik, Mottet, and Pinsker [5]). In this paper, we suggest an alternative approach that might make this work unnecessary.

Secondly, gadget reductions fail to explain NP-hardness in a slightly more general setting of *promise constraint satisfaction problems*. In the promise version of the problem, each instance comes with two versions of each constraint, a *stronger* and a *weaker* one. The goal is then decide between two (disjoint, but not complementary) cases: all stronger constraints can be satisfied, or not even the weaker constraints can be satisfied. A prominent problem described in this scope is *approximate graph colouring* which asks, e.g., to decide between graphs that are 3-colourable and those that are not even 6-colourable. More formally, a template of a promise CSP consists of a pair of structures, such that the first structure maps homomorphically to the second. The goal is to decide, given as input a third structure, whether the input maps homomorphically to the first structure of the template, or it does not map homomorphically even to the second. A systematic study of promise CSPs has been started by Austrin, Guruswami, and Håstad [3] where a certain promise version of SAT was considered, and continued in a series of papers including generalisations of the algebraic approach by Brakensiek and Guruswami [16] and Barto, Bulín, Krokhin, and Opršal [7]. In particular, [7] characterises gadget reductions in terms of (generalised) *polymorphisms* in a similar way as [19] characterises gadget reductions in the scope of (classic) CSPs (see Theorem 2.4 for a formal statement). Many of the NP-hardness results in the promise setting [e.g., 27; 34; 3; 38; 30; 52; 18; 4; 43] cannot be adequately explained by gadget reductions. These hardness results usually rely on variations of the PCP theorem of Arora and Safra [1] which is better suited for a different, more analytical, approximation version of CSPs. Indeed, new reductions for promise CSPs have been called for by Barto, Bulín, Krokhin, and Opršal [7], and by Barto and Kozik [10]. The goal is to find a class of reductions that would explain all of these hardness results, and subsequently extend the hardness to other interesting promise CSPs, e.g., the approximate graph colouring.

Our contributions

The primary objective of the present paper is to offer a novel perspective on the complexity landscape of CSPs and promise CSPs that is focused on reductions rather than algorithms or algebraic hardness conditions. We aim to identify a class of efficient reductions that addresses both the tractability side of CSPs and the hardness side of promise CSPs. To this end, the suggested class of reductions should have the following properties:

- *robustness*, informally meaning that the notion of reduction remains invariant under minor variations in its definition.
- *expressiveness*, that is, it should significantly extend the capabilities of gadget reductions.
- *characterisability*, meaning that it must be possible to characterise, in some terms similar to the algebraic characterisation of gadget reductions, whether a given (promise) CSP reduces to another given (promise) CSP.

We introduce a new general class of reductions between (promise) CSPs, referred to as *k-consistency reductions* or *Datalog$^\cup$ reductions*, and supplement it with several results that showcase the robustness and expressiveness of this family of reductions. Furthermore, we take initial steps towards its characterisability. A fully fledged theory for k-consistency reductions would have several overreaching consequences and, hence, it falls beyond the scope of the present paper, which aims solely to initiate it.

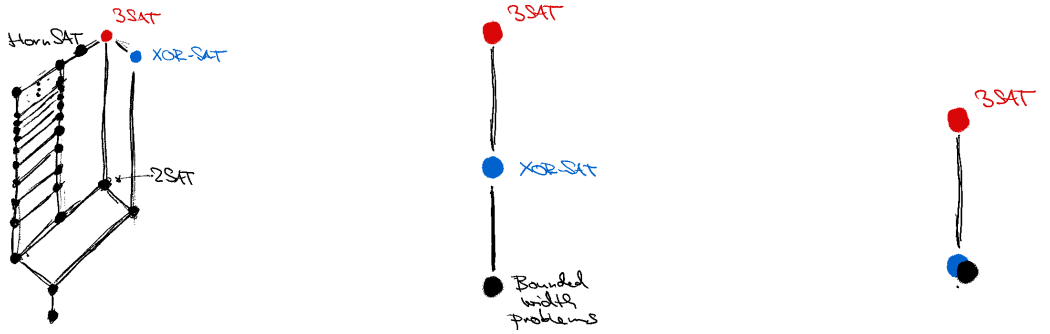
Robustness. There are two key properties of a robust class of reductions in the scope of (promise) CSPs: *monotonicity* with respect to the homomorphism preorder, which we require in order to preserve the structure of the problem, and *closure under composition*, which is a natural requirement that is essential in order to chain reductions together. Most of the reductions used in the scope of promise CSPs are monotone, but not all compose; in particular, the reductions provided by Barto and Kozik [10] do not compose (see Example 3.8).

First, we describe our suggested class of reductions as logical interpretations in a monotone fragment of the fixed-point logic, namely in the logic of Datalog programs endowed with an operator that allows to take a disjoint union of relations. We call them *Datalog$^\cup$ reductions*.

It is relatively easy to show that this reduction has the first two properties. In particular, we show that Datalog$^\cup$ reductions compose (Theorem 3.7). Moreover, we show that Datalog$^\cup$ reductions encapsulate gadget reductions, which is a generalisation of a result of Atserias, Bulatov, and Dawar [2, Theorem 5] to the promise setting, although we use disjoint unions instead of parameters.

Theorem 1 (Theorem 3.9 informally). *Every gadget reduction is expressible as a Datalog$^\cup$ reduction up to homomorphic equivalence.*

Secondly, we provide a combinatorial counterpart of Datalog$^\cup$ reductions. This alternative construction has two ingredients: gadget reductions and the local consistency algorithm for (promise) CSP. Local consistency is ubiquitous in constraint programming (see for example [46]). Intuitively, the consistency algorithm (with a parameter k) iteratively adds all newly implied constraints that can be derived by simultaneously considering k variables. From a theoretical standpoint, it has been more common to consider a decision version (which we call *consistency test*) that merely outputs



(a) gadget reductions **(b) Datalog$^\cup$ reductions** **(c) polynomial-time reductions**

Figure 1: Boolean CSPs ordered by three classes of reductions with examples of problems belonging to some classes.

'no' if the set of derived constraints contains an unsatisfiable constraint and 'yes' otherwise. More specifically, several papers have investigated which CSPs are solvable by such algorithms, leading to a complete characterisation in the non-promise case by Barto and Kozik [9]. Datalog programs, too, can be used to solve (promise) CSPs by turning them into a decision algorithm choosing a nullary goal predicate. It was soon established by Kolaitis and Vardi [37] that the consistency test and Datalog programs have the same power in solving non-promise CSPs. This equivalence extends to the promise setting (set $\mathscr{A}$ to be the union of positive and negative instances, and B to be the first structure of the template in the statement of [37, Theorem 4.8]).

Getting back to reductions, we refer to the combination of a gadget reduction and the local consistency algorithm with parameter k as the *k-consistency reduction*. We show that, indeed, consistency reductions and Datalog reductions have the same power.

Theorem 2 (Theorem 3.12 informally). *A promise CSP reduces to a second promise CSP via a Datalog$^\cup$ reduction if and only if it reduces via the k-consistency reduction.*

Alternatively, the preceding theorem can be viewed as an extension of the well-known equivalence between the decision versions of consistency and Datalog to the realm of reductions. The proof, however, faces two primary technical challenges. Firstly, we are dealing here with the full-fledged version (instead of the Boolean version) of consistency and Datalog. And, more importantly, it is additionally necessary to understand how Datalog interpretations interact with gadget reductions.

Expressiveness. We present several examples illustrating that k-consistency reductions effectively tackle, at least to some extent, the aforementioned shortcomings of gadget reductions.

Firstly, it is conceivable and consistent with our current understanding of the complexity of promise CSPs, that a promise CSP is NP-hard if and only if it allows a k-consistency reduction from an NP-hard CSP, e.g., graph 3-colouring. For example, it is relatively easy to check that the reductions of Barto and Kozik [10] are covered by k-consistency reductions, and consequently many results about NP-hardness of promise CSPs can be explained by a k-consistency reduction including the hardness of 5 colouring 3-colourable graphs [7] and the NP-hardness in [27; 3; 38; 30; 4; 43]. Furthermore, the reduction used by Wrochna and Živný [52] to provide NP-hardness of colouring k-colourable graphs with $\binom{k}{\lfloor k/2 \rfloor} - 1$ colours is a Datalog$^\cup$ reduction from approximate graph colouring which is NP-hard by a previous result of Huang [34]. The result of Huang is one of the few exceptions; we currently do not know whether it can be also explained by a k-consistency reduction from graph 3-colouring.

Second, several well-known hierarchies of relaxation algorithms for promise CSPs fall within our framework. A comprehensive look at hierarchies is given in Section 4 where we describe several hierarchies in terms of k-consistency reductions to a fixed problem. This includes the *bounded width hierarchy* (i.e., the hierarchy of promise CSPs expressible by Datalog), and the *Sherali–Adams hierarchy* of linear programming for promise CSPs. Furthermore, we show that the Sherali–Adams hierarchy of (promise) CSPs coincides with the hierarchy of (promise) CSPs reducible to linear programming by a k-consistency reduction (Theorem 4.3). A different framework to study hierarchies of relaxations of algorithms was recently introduced by Ciardo and Živný [24]. The Ciardo-Živný hierarchy is always subsumed by the hierarchy of k-consistency reductions to a fixed problem, and in some cases, e.g., the Sherali–Adams hierarchy, both hierarchies align perfectly.

Since consistency reductions contain, as a particular case, gadget reductions, it follows that any class of (promise) CSPs closed under consistency reductions can be, at least theoretically, described in terms of polymorphisms. This is a very desirable feature as polymorphisms lie at the heart of the algebraic approach. As a direct consequence of our findings it follows that both the class of promise CSPs solvable by the consistency test and the class of promise CSPs solvable by some level of the Sherali–Adams relaxation are closed under gadget reductions. While the former result had previously been established [7, Lemma 7.5], the latter result, in its full generality, is new. Prior to our work it had only been known for the particular case of non-promise CSPs where it can be derived by combining the results of Thapper and Živný [49] and Barto and Kozik [9].

Finally, we get to the tractability side of CSPs. In the Boolean case (i.e., for templates with two elements) it is known that every tractable CSP is reducible via a gadget reduction to one of three fundamental tractable problems (HornSAT, 2SAT, and XOR-SAT)

[47; 14]. If we replace gadget reductions by Datalog$^\cup$ reductions then every tractable Boolean CSP can be reduced to a single problem, namely, solving systems of linear equations over $\mathbb{Z}_2$ (XOR-SAT). See also Figure 1 comparing gadget, Datalog$^\cup$, and general polynomial-time reductions on Boolean CSPs.[1]

However, a finite-template tractable CSP does not necessarily reduce to XOR-SAT via a Datalog$^\cup$ reduction (when template is allowed to have more than two elements). Yet, this seems to be caused by the fact that that the problem has the ability to 'express' equations modulo a prime different from 2. This leads us to believe that a finite-template tractable CSP can perhaps reduce via a Datalog$^\cup$ reduction to solving systems of linear equations over $\mathbb{Z}_n$ for some integer n (which is divisible by all the primes p such that equations mod p are 'expressible' by the problem). Given that, in turn, this problem reduces via a Datalog$^\cup$ reduction to solving systems of linear equations over integers, we put forth the following conjecture, supported by evidence presented in Section 4.

Conjecture 1 (Conjecture 4.10). *Every finite template CSP either allows a Datalog$^\cup$ reduction from 3-colouring (and hence it is NP-hard), or is reducible to solving systems of affine equations over $\mathbb{Z}$ via a Datalog$^\cup$ reduction (and hence it is in P).*

The CSPs that are conjectured to be NP-hard in the above conjecture include all CSPs hard under the Bulatov–Zhuk dichotomy (since gadget reductions are subsumed by Datalog$^\cup$ reductions due to Theorem 1). Naturally, this implies that the two classes coincide unless P = NP. A direct proof (without assuming P ≠ NP) that every CSP allowing a Datalog$^\cup$ reduction from 3-colouring also allows a gadget reduction from 3-colouring is not known. We have decided to express the dichotomy using Datalog$^\cup$ reductions instead of gadget reductions because this formulation generalizes better to promise CSPs: It is consistent with our knowledge that the hardness condition given in the conjecture covers all NP-complete finite-template promise CSPs, while there are NP-hard promise CSPs (e.g., 5-colouring 3-colourable graphs) whose hardness can be shown via a Datalog$^\cup$ reduction but not via a gadget reduction from 3-colouring. Likewise, the tractability side of our conjecture gives an algorithm which generalises in a straight-forward way to promise CSPs unlike the algorithms of Bulatov and Zhuk.

Characterisability. We describe a characterisation of a special case of k-consistency reduction, referred to as *arc-consistency reduction* (which is obtained by replacing the k-consistency algorithm in the k-consistency reduction by a weaker *arc-consistency*), in terms of a certain transformation ω (see Definition 5.4) of *polymorphisms* of the templates (see, e.g., [7, Section 2.2] for the necessary standard definitions).

Theorem 3 (Theorem 5.5 informally). *A promise CSP with polymorphisms $\mathscr{A}$ reduces to a promise CSP with polymorphisms $\mathscr{B}$ via the arc-consistency reduction if and only if there is a 'weak' minion homomorphism $\omega(\mathscr{B}) \to \mathscr{A}$.*

In comparison, the above-mentioned theorem of Barto, Bulín, Krokhin, and Opršal [7] claims that there is a gadget reduction between the two problems if and only if there is a minion homomorphism $\mathscr{B} \to \mathscr{A}$.

We note that a different characterisation of the arc-consistency reduction was recently provided by Mottet [42, Theorem 12].

Organisation of the paper

We present basic definitions and notation in Section 2. This is followed by the definition of our framework and formulation of the basic theorems of our theory in Section 3. Section 4 describes several hierarchies in terms of k-consistency reductions, and discusses evidence for our conjecture about tractability of finite-template CSPs. In Section 5, we describe the characterisation of the arc-consistency reduction. And finally, in Section 6, we outline a few directions for future research, and possible applications of the theory developed in the present paper.

Detailed proofs of the claims made here, and several additional examples can be found in the appendix of the full version of this paper [25]. The cited version is organised analogously to this paper; in particular the content of the main part of [25] coincides with this paper.

2 PRELIMINARIES

Many of the constructions described in this paper can be defined using several different languages, e.g., an instance of a CSP can be viewed as a list of constraints, a logical formula, or a relational structure. In this paper, the structural perspective takes precedence.

Sets are generally denoted by capital letters, and integers by lower-case letters. We often do not explicitly specify that such symbols represent sets, or integers when it is clear from the context, e.g., when the symbol appears in an index. We denote by $[n]$ the set of the first n positive integers, and by A^X the set of all functions $f\colon X \to A$. We use the notation $\pi^{-1}(y)$ for the set of preimages of y under π, i.e., the set $\{x \in X \mid \pi(x) = y\}$. We will denote entries in a tuple a by lower indices, i.e., $a = (a_1, \dots, a_k)$, and occasionally view a tuple $a \in A^k$ as a function $a\colon [k] \to A$. The identity map on X is denoted by 1_X.

2.1 Constraint satisfaction problems

To settle on the playing field, we formally define the constraint satisfaction problem (CSP), and its promise variant. We refer to Barto, Krokhin, and Willard [11] for a deeper exposition of the algebraic theory of CSPs, and to Krokhin and Opršal [39] for more background and examples of promise CSPs. We use the standard notation and definitions for promise CSPs as in Barto, Bulín, Krokhin, and Opršal [7].

The *uniform CSP* is a decision problem whose instance consists of variables $v, w, \dots$ each of which is allowed to attain values from some domain and constraints, each of which involves several variables. The goal is to decide whether there is an assignment of values to variables that simultaneously satisfies all constraints. In the present paper, we are concerned with fixed-template CSP which restricts the possible constraints in the input. This is formalised as a homomorphism problem between relational structures where the target structure is fixed. To allow for different domains for distinct variables we allow structures to be typed (multisorted). The multisorted framework does not increase the complexity of the

[1] Fig. 1a is due to Bodirsky and Vucaj [14]; and Fig. 1c is due to Bulatov [20] and Zhuk [53] and assumes P ≠ NP.

proofs, merely the complexity of notation. Moreover, as noted in [10, Section B.4], it provides a natural setting for the theory since it places *(layered) label cover problem*, whose promise version is used as an intermediate step in reductions (both in this paper and in the algebraic characterisation of gadget reductions [7, Theorem 3.12]), into the framework of CSPs. In addition, several of the key concepts introduced here (e.g., the extension of Datalog with disjoint unions) are more naturally expressed in the multisorted setting.

Definition 2.1. A *(multisorted) relational signature* consists of a set of types (sorts) $\{t, s, \dots\}$ and a set of relational symbols $\{R, S, \dots\}$, where each symbol has a fixed *arity* which is formally a tuple of types. We denote this tuple by $\mathrm{ar}_R, \mathrm{ar}_S, \dots$, and often say simply that R is of arity k where k is the length of ar_R.

A *relational structure* $\mathbf{A}$ of such a signature is a tuple

$$\mathbf{A} = (A_t, A_s, \dots; R^{\mathbf{A}}, S^{\mathbf{A}}, \dots)$$

where A_t, A_s, etc. are sets and $R^{\mathbf{A}}$, $S^{\mathbf{A}}$, etc. are relations on these sets of the corresponding arity, e.g., $R^{\mathbf{A}} \subseteq A_{\mathrm{ar}_R(1)} \times \cdots \times A_{\mathrm{ar}_R(k)}$ assuming that ar_R has length k.

We will usually assume that both the signature and the structures are finite, i.e., the signature contains only finitely many types and symbols, and all of the sets A_t, etc. are finite. We call any $a \in A_t$ an *element of* $\mathbf{A}$ *of type* t (we use sort and type interchangeably), and we do not consider two elements of $\mathbf{A}$ to be equal unless they have the same type. For example, if $\mathbf{A}$ has two domains $A_t = A_s = \{0, 1\}$, we will view it as a structure with four elements. We will write $x \in \mathbf{A}$ for 'x is an element of $\mathbf{A}$'. All structures in this paper are typed relational structures, and signature of such structures is given implicitly.

A *homomorphism* between two structures is a mapping that maps elements of one structure to elements of the second structure that preserves types and all relations. A necessary requirement for a homomorphism to exist is that both structures have the same signature. Formally, a *homomorphism* from a structure $\mathbf{A}$ to a structure $\mathbf{B}$ is a collection f of maps $f_t \colon A_t \to B_t$, one for each type t, such that for each relational symbol R and $(a_1, \dots, a_k) \in R^{\mathbf{A}}$, we have

$$(f_{\mathrm{ar}_R(1)}(a_1), \dots, f_{\mathrm{ar}_R(k)}(a_k)) \in R^{\mathbf{B}}.$$

We write $f(a)$ instead of $f_t(a)$ if the type t of a is clear from the context, we write $f \colon \mathbf{A} \to \mathbf{B}$ if f is a homomorphism, and $\mathbf{A} \to \mathbf{B}$ if such a homomorphism exists.

The *(typed) constraint satisfaction problem with template* $\mathbf{A}$ is a decision problem whose goal is: given a structure $\mathbf{X}$, decide whether $\mathbf{X} \to \mathbf{A}$. We denote this problem by $\mathrm{CSP}(\mathbf{A})$. Before we define promise CSP, recall that a promise problem is a generalisation of a decision problem where the *yes* and *no* instances form disjoint, but not necessarily complementary, sets.

Definition 2.2. A pair of structures $\mathbf{A}, \mathbf{A}'$ with $\mathbf{A} \to \mathbf{A}'$ is called a *promise template*. We assume that $\mathbf{A}$ in a promise template is finite unless explicitly specified otherwise. Given such a template, a *(typed) promise constraint satisfaction problem* is a promise problem whose goal is, on an input $\mathbf{X}$ which is a structure (of the same signature), to output *yes* if $\mathbf{X} \to \mathbf{A}$, and *no* if $\mathbf{X} \not\to \mathbf{A}'$ (since $\mathbf{A} \to \mathbf{A}'$, these two cases are disjoint). We denote this problem by $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$.

A *constraint* of an instance $\mathbf{X}$ of $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ is an expression of the form $(v_1, \dots, v_k) \in R^{\mathbf{X}}$ which we formally view as a pair $((v_1, \dots, v_k), R)$ where $v_1, \dots, v_k$ are elements of X and R is a relational symbol such that the expression is satisfied. The tuple $(v_1, \dots, v_k)$ is called the *scope* of the constraint.

Alternatively, one could define a promise CSP as a promise *search* problem where the goal would be, given $\mathbf{X}$ that is promised to map to $\mathbf{A}$ via a homomorphism, to find a homomorphism $\mathbf{X} \to \mathbf{A}'$. It is currently not known whether these two versions of promise CSPs are of equivalent complexity. This paper focuses on the decision version of promise CSP, although a few results (mostly with some caveats) apply also to the search version. Finally, note that $\mathrm{CSP}(\mathbf{A})$ is $\mathrm{PCSP}(\mathbf{A}, \mathbf{A})$, and therefore every result about promise CSPs is applicable to CSPs.

Finally, we will denote by $\mathbf{B}^X$ the *X-fold power* of a structure $\mathbf{B}$, i.e., a structure whose t-th domain is B_t^X for each type t, and the relations are defined as follows: $(b_1, \dots, b_k) \in R^{\mathbf{B}^X}$ if and only if $(b_1(i), \dots, b_k(i)) \in R^{\mathbf{B}}$ for all $i \in X$.

2.2 Reductions

In this paper we deal with the simplest form of a reduction between promise problems. By a *reduction* we mean a (usually efficiently computable) function between the instances of the two problems which maps positive instances to positive instances and negative instances to negative instances. In the case of promise CSPs, a reduction from $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ is a mapping ψ between the corresponding categories of structures such that (1) if $\mathbf{X} \to \mathbf{A}$ then $\psi(\mathbf{X}) \to \mathbf{B}$, and (2) if $\mathbf{X} \not\to \mathbf{A}'$ then $\psi(\mathbf{X}) \not\to \mathbf{B}'$. The item (1) is usually referred to as *completeness*, and the item (2) as *soundness*. We will usually use and prove the soundness in its converse form: if $\psi(\mathbf{X}) \to \mathbf{B}'$ then $\mathbf{X} \to \mathbf{A}'$.[2]

Reductions are more general than algorithms: every decision algorithm can be viewed as a reduction to a problem with two admissible instances *yes* and *no* where *yes* is the positive instance. We express this problem as a 'trivial CSP' whose template is a structure in a very degenerate signature Ω which allows only two structures: Ω has no types and has a single nullary relational symbol C. There are only two Ω-structures: $\bot$ and $\top$. The structure $\bot$ is defined as the structure with empty nullary relation, and $\top$ is the structure with the non-empty nullary relation.

Definition 2.3. The *trivial CSP* is the CSP with template $\bot$. Note that $\bot$ is a positive instance of this CSP (since $\bot \to \bot$), and $\top$ is a negative instance (since $\top \not\to \bot$).

We assume that all (promise) CSPs considered in this paper have negative instances — this can be always ensured by adding a nullary constraint that cannot be satisfied.

2.3 Gadget reductions

Gadget reductions are the more traditional reductions in the realm of finite-template CSPs and promise CSPs. They have been classified by the algebraic approach. We refer to Krokhin, Opršal, Wrochna, and Živný [40, Section 4.1] for a detailed exposition.

[2] In the case of a reduction between the search version of (promise) CSPs, it would be necessary to additionally require that this converse is witnessed by an efficiently computable function that, given a homomorphism $\psi(\mathbf{X}) \to \mathbf{B}'$, outputs a homomorphism $\mathbf{X} \to \mathbf{A}'$.

A *gadget replacement* is a transformation of a Π-structure $\mathbf{X}$ to a Σ-structure $\gamma(\mathbf{X})$ that replaces every element $v \in \mathbf{X}$ by a fixed Π-structure that only depends on the type of v, and replaces every constraint $(v_1, \dots, v_k) \in R^{\mathbf{X}}$ by another Σ-structure that only depends on the relation R identifying some of the vertices of this gadget with the elements of the gadgets introduced by replacing $v_1, \dots, v_k$ (see also [39, Definition 2.9]).[3] If such a gadget replacement yields a reduction, we call it a *gadget reduction*. Gadget reductions are characterised using *polymorphisms and minion homomorphisms* [7, Definition 2.15] (see also Definitions 5.2 and 5.3).

Theorem 2.4 (Barto, Bulín, Krokhin, and Opršal [7, Theorem 3.1]). $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ *reduces to* $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *by a gadget reduction if and only if there is a minion homomorphism* $\mathrm{Pol}(\mathbf{B}, \mathbf{B}') \to \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$.

The gadget reduction provided by the above theorem in presence of a minion homomorphism between the polymorphisms of the templates is called a *universal gadget reduction* (it should be noted that the reduction depends on the structures $\mathbf{A}$ and $\mathbf{B}$).

3 LOCAL CONSISTENCY REDUCTIONS

In this section, we define the class of Datalog$^{\cup}$ and consistency reductions, we discuss some basic properties of these reductions, and provide formal statements of Theorems 1 and 2. We define the reductions from the logical perspective, followed by the combinatorial version and the equivalence of both definitions.

3.1 Logical reductions

We first describe the class of reductions we consider in this paper in terms of logic, or more precisely Datalog. Since our signatures are multisorted this means that all logic formulae need to respect the types: We assume that every variable $x_1, x_2, \dots$ has an associated type and define an atomic Π-formula (where Π is a relational signature) to be (1) $R(x_1, \dots, x_n)$ where R is a Π-symbol, and the type of x_i is $\mathrm{ar}_R(i)$ for all $i \in [k]$, or (2) an equality $x_1 = x_2$ where x_1 and x_2 are of the same type.

A *Datalog program* ϕ with input signature Π is constituted by a relational signature Δ satisfying $\Pi \subseteq \Delta$ (meaning that it has the same types as Π and each symbol of Π appears in Δ with the same arity) along with a finite collection of rules that are traditionally written in the form

$$t_0 \leftarrow t_1, \dots, t_r$$

where $t_0, \dots, t_r$ are atomic Δ-formulae. In such a rule t_0 is called the *head* and $t_1, \dots, t_r$ the *body* of the rule. Moreover, we require that neither the symbols in Π nor the equality appear in the head of any rule. The symbols in Π are called *input symbols*, or *EDBs* (standing for *extensional database predicates*, while all the other symbols would be *IDBs*, *intensional database predicates*). Furthermore, one of the Δ-symbols is designed as *output*. [4] A Datalog program receives as input a Π-structure $\mathbf{A}$ and produces a relation with the same arity of the output predicate using standard fix-point semantics. That is, let $\mathbf{B}$ be the Δ-structure computed in the following way.

(1) Start with setting $R^{\mathbf{B}} = R^{\mathbf{A}}$ if $R \in \Pi$ and $R^{\mathbf{B}} = \emptyset$ otherwise.
(2) If there is a rule $t_0 \leftarrow t_1, \dots, t_r$ and some assignment $h\colon X \to A$, where X are the variables occurring in the rule, such that the atomic predicates $t_1, \dots, t_r$ hold in $\mathbf{B}$ after the substitution then include $(h(x_1), \dots, h(x_k))$ in $R^{\mathbf{B}}$ where $t_0 = R(x_1, \dots, x_k)$. Repeat until we get to a fixed point.
(3) Output the relation $R^{\mathbf{B}} \subseteq A_{\mathrm{ar}_R(1)} \times \cdots \times A_{\mathrm{ar}_R(k)}$ where R is the output predicate.

We will denote the output of such a Datalog program by $\phi(\mathbf{A})$. Commonly in the context of CSPs, the output of a Datalog program is a nullary predicate, so that such a program outputs either *true*, or *false* — we call such Datalog programs *Datalog sentences*. The *arity* of a Datalog program is the arity of the output predicate, and the *width* of a Datalog program is the maximal number of variables in a rule.

We define *Datalog interpretations* as a particular case of logical interpretations. This definition hinges on interpretation of Datalog programs as logical formulae. More precisely, a Datalog program ϕ whose output is a relation with arity k is seen as a formula with k free variables, so that $\mathbf{A} \models \phi(a_1, \dots, a_k)$ if and only if $(a_1, \dots, a_k) \in \phi(\mathbf{A})$.

Definition 3.1. Fix two relational signatures Σ and Π. A *Datalog interpretation* ϕ mapping Π-structures to Σ-structures (usually referred to as a Datalog interpretation of Σ into Π) consist of: a Datalog program ϕ_t with input signature Π for each Σ-type t, and a Datalog program ϕ_R also with input signature Π for each Σ-symbol R, such that, for each symbol R of arity k, the arity of ϕ_R is the concatenation of the arities of $\phi_{\mathrm{ar}_R(1)}, \dots, \phi_{\mathrm{ar}_R(k)}$. Such an interpretation is said to be of *width at most* k if all ϕ_t and ϕ_R are of width at most k.

The application of such a Datalog interpretation ϕ to a Π-structure $\mathbf{A}$ produces the Σ-structure

$$\phi(\mathbf{A}) = (\phi_t(\mathbf{A}), \phi_s(\mathbf{A}), \dots; \phi_R(\mathbf{A}), \phi_S(\mathbf{A}), \dots),$$

where, for each Σ-symbol R, $\phi_R(\mathbf{A})$ is interpreted as a relation of the required arity k in the natural way, i.e., it consists of all tuples

$$((a_1, \dots, a_{j_1}), \dots, (a_{j_{k-1}+1}, \dots, a_{j_k})) \in \phi_{\mathrm{ar}_R(1)}(\mathbf{A}) \times \cdots \times \phi_{\mathrm{ar}_R(k)}(\mathbf{A}),$$

such that $(a_1, \dots, a_{j_k}) \in \phi_R(\mathbf{A})$.

Unless there is a clash of notation (in which case we will specify the output predicate explicitly) we shall be using D_t and R as output predicate of ϕ_t and ϕ_R respectively.

Since we do not allow parameters, nor inequality in Datalog interpretations, we obtain *monotone* transformations in the following sense.

Lemma 3.2. *Let* ϕ *be a Datalog interpretation, and* $\mathbf{A}$ *and* $\mathbf{B}$ *structures. If* $\mathbf{A} \to \mathbf{B}$*, then* $\phi(\mathbf{A}) \to \phi(\mathbf{B})$.

[3] Gadgets and gadget replacements may be formalised elegantly in a categorical language: We view a signature Π as a category whose objects are all types and all symbols of Π, and contains a morphism $p_i\colon R \to \mathrm{ar}_R(i)$ for each symbol R of arity k and $i \in [k]$. Hence a Π-structure can be viewed as a functor from Π to the category of sets. A *gadget* is then a contravariant functor from Π to Σ-structures. Every such gadget then defines a pair of adjoint functors (see, e.g., Pultr [45]); *gadget replacement* is the left adjoint, and the right adjoint corresponds (up to homomorphic equivalence) to *pp-powers* [7, see Definition 4.7].

[4] In database theory, Datalog programs very often have several output predicates (and define structures instead of relations). Such program can be viewed as a special case of a Datalog interpretation which we define later.

Datalog interpretations are powerful when used as reductions; they can reduce a substantial class of CSPs, including 2SAT and Horn-3SAT, to the trivial problem. More precisely, it can be observed that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ reduces to $\mathrm{CSP}(\bot)$ by a Datalog reduction if and only if it is expressible in Datalog, i.e., whenever there is a Datalog sentence which is *true* in all negative instances, and *false* in all positive instances of $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$. This is exemplified in the following example.

Example 3.3 (2-colouring). We shall define a Datalog interpretation ϕ that provides a reduction $\mathrm{CSP}(\mathbf{K}_2) \le \mathrm{CSP}(\bot)$. To define such interpretation, we only need ot provide a nullary Datalog program ϕ_C which is based on a well-known program solving 2-colouring (see, e.g., [37, p. 14]):

$$\begin{aligned} P(x,y) &\leftarrow E(x,y) \\ P(x,y) &\leftarrow E(y,x) \\ P(x,y) &\leftarrow P(x,z), P(z,w), P(w,y) \\ C &\leftarrow P(x,x) \end{aligned}$$

It is straightforward to check that C is derived if and only if the input graph has an odd cycle. Consequently, we get $\phi(\mathbf{G}) = \top$ if and only if $\mathbf{G}$ is not 2-colourable. Since $\top$ is the negative instance of $\mathrm{CSP}(\bot)$, this concludes that ϕ is a reduction between the two problems.

Further examples of Datalog interpretations providing tractability of some (promise) CSPs can be obtained through a framework of Ciardo and Živný [24]; more precisely, it may be observed that the tensor construction introduced in [24] can be also expressed as a Datalog interpretation. For more details about tensor hierarchies, see the full version [25, Appendix D.1].

Let us continue with an example of a reduction used to provide NP-hardness of the approximate graph colouring.

Example 3.4 (Line digraph construction). The *line digraph construction* δ, described below, was used by Wrochna and Živný [52] to provide hardness of some cases of approximate graph colouring. They obtained these results by iterating (i.e., composing) δ to reduce from $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_{2^{\Omega(k^{1/3})}})$ for large enough k (known to be NP-hard by a result of Huang [34]) to $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_{b(k)})$, where $b(k) = \binom{k}{\lfloor k/2 \rfloor} - 1$, for all $k \ge 4$. This result improved the state-of-the-art for all $k > 5$ and matched it for $k = 5$. The same reduction is also used by Guruswami and Sandeep [32] to provide conditional hardness of approximate graph colouring under d-to-1 conjecture, and by Ciardo and Živný [23] to provide a negative result about solvability of approximate graph colouring by a combination of Sherali–Adams with affine integer programming.

The *line digraph construction* is a Datalog interpretation from digraphs to digraphs that changes the domain of an input digraph. It can be defined by a Datalog interpretation $\delta = (\delta_V; \delta_E)$ where δ_V is the program

$$V'(x,y) \leftarrow E(x,y)$$

with output V', and δ_E is the program

$$E'(x,y,y,z) \leftarrow E(x,y), E(y,z)$$

with output E'. The application on a digraph $\mathbf{G}$ then yields a digraph $\delta(\mathbf{G})$ whose vertices are edges of $\mathbf{G}$, and two such 'vertices' are connected by an edge if they are incident, i.e., of the form (u,v) and (v,w).

The monotone version of Datalog interpretations that we defined above cannot fully emulate gadget reductions. In particular, it cannot express taking disjoint unions of domains and relations. Atserias, Bulatov, and Dawar [2] use parameters in Datalog interpretations to express these disjoint unions up to a finite number of exceptions — we note that allowing parameters (or inequality) would yield reductions that are not monotone. Instead, we extend Datalog interpretations with disjoint unions. Extending logical interpretations to allow taking disjoint unions of definable relations is not completely unusual (see, e.g., [15, Definition 3.1]). Here, we take an advantage from the multisorted approach to provide a technical solution that helps us to track that the relations are well-defined.

Definition 3.5. Assume that Π and Σ are two relational signatures. A *union gadget* υ is defined by a pair of mappings (d, r), where d maps Π-types to Σ-types and r maps Π-symbols to Σ-symbols, such that for each Π-symbol R, we have $\mathrm{ar}_{r(R)} = d \circ \mathrm{ar}_R$.

Such a union gadget can be then applied on a structure $\mathbf{A}$ with disjoint domains[5] to produce a Σ-structure $\upsilon(\mathbf{A})$ whose t-th domain is the (disjoint) union of A_i's with $d(i) = t$, and similarly, $S^{\upsilon(\mathbf{A})} = \bigcup_{R \in r^{-1}(S)} R^{\mathbf{A}}$ for each Σ-symbol S.

It is not difficult to observe that disjoint unions are also expressible as gadget replacements.

Definition 3.6. A *Datalog$^\cup$ reduction* is a composition $\upsilon \circ \phi$ where υ is a union gadget and ϕ a Datalog interpretation. We write

$$\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \le_{\mathsf{DL}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$$

if there exists a Datalog$^\cup$ reduction between the two problems.

Note that a Datalog$^\cup$ reduction between two single-sorted CSPs may use a multisorted structure as an intermediate step even though we are reducing between two single-sorted CSPs.

The fact that Datalog$^\cup$ reductions compose follows from observing that Datalog interpretations compose, and that they can be permuted with disjoint unions, which also compose. This makes the relation $\le_{\mathsf{DL}}$ (on promise CSPs) transitive, and hence it defines a preorder.

Theorem 3.7. *Let ϕ and χ be two Datalog$^\cup$ reductions such that the output signature of ϕ coincides with the input signature of χ. Then there is a Datalog$^\cup$ reduction ψ such that $\psi(\mathbf{A})$ and $\chi\phi(\mathbf{A})$ are isomorphic for all structures $\mathbf{A}$.*

Example 3.8. The *k-reductions* [39, see p. 48] used by Barto and Kozik [10] do not compose unlike Datalog$^\cup$ reductions. These weaker reductions can be described briefly as a combination of Datalog interpretations without recursion and gadget reductions. We denote by $\mathbf{P}_n$ the oriented path with n edges, i.e., the digraph $(\{0, \dots, n\}; \{(i, i+1) \mid i < n\})$. It can be observed that: (1) $\mathrm{CSP}(\mathbf{P}_2)$ reduces to $\mathrm{CSP}(\mathbf{P}_1)$ by a gadget reduction which is a special case of a k-reduction, (2) $\mathrm{CSP}(\mathbf{P}_1)$ reduces to $\mathrm{CSP}(\bot)$ by a k-reduction, and (3) $\mathrm{CSP}(\mathbf{P}_2)$ does not reduce to $\mathrm{CSP}(\bot)$ by a k-reduction (this is since $\mathrm{CSP}(\mathbf{P}_2)$ does not have *finite duality* [21, Definition 3]).

[5] We can always enforce disjoint domains by replacing an element a of type t with a pair $(a; t)$.

Clearly, every Datalog interpretation is a Datalog$^\cup$ reduction (compose with the trivial union gadget). We show that every gadget replacement can be also expressed as a Datalog$^\cup$ reduction up to homomorphic equivalence.

Theorem 3.9. *For every gadget γ there is a Datalog$^\cup$ reduction ψ such that, for all $\mathbf{A}$, $\gamma(\mathbf{A})$ and $\psi(\mathbf{A})$ are homomorphically equivalent.*

3.2 Combinatorial reductions

The combinatorial counterpart of Datalog$^\cup$ reductions is based on the local consistency algorithm for CSPs and the universal gadget reduction. Consistency reductions are defined by the two templates and a single parameter k. Let us assume that we are trying to reduce $\mathrm{PCSP}(\mathbf{A}, *)$ to $\mathrm{PCSP}(\mathbf{B}, *)$; the second part of the templates are irrelevant for the definition of the reduction, but play an important role for the soundness which we do not characterise here.

The k-consistency reduction is a combination of two procedures: *Enforcing k-consistency* which is possibly the most intuitive approach to solving CSPs, and it is used in many CSPs algorithms (including, e.g., Zhuk's polynomial-time algorithm [53]), and *the universal gadget replacement* which is based on a standard way to reduce from label cover to $\mathrm{CSP}(\mathbf{B})$ using a gadget consisting of powers of $\mathbf{B}$ (see, e.g., [7, Section 3.3]).

Let $\mathbf{X}$ and $\mathbf{A}$ be structures of the same signature and let $K \subseteq X$. A *partial homomorphism* from K to $\mathbf{A}$ is a mapping $f\colon K \to A$ that preserves types and relations, i.e., it satisfies the definition of a homomorphism when all variables are quantified in K instead of X. Intuitively, a partial homomorphism is simply a partial solution to the instance defined on the given set K. We denote by $\binom{X}{\leq k}$ the set of all at most k-element subsets of X.

Definition 3.10 (k-consistency reduction). Let $k \geq 1$ and let $\mathbf{A}$ be a Π-structure and $\mathbf{B}$ be a Σ-structure.[6] The k-consistency reduction maps every Π-structure $\mathbf{X}$ to a Σ-structure $\mathbf{Y}$ constructed by the following procedure:

(1) For each $K \in \binom{X}{\leq k}$, let $\mathcal{F}_K$ be the set of all partial homomorphisms from K to $\mathbf{A}$.
(2) Ensure that for each $L \subset K \in \binom{X}{\leq k}$, the sets $\mathcal{F}_K$ and $\mathcal{F}_L$ are *consistent*, i.e., remove from $\mathcal{F}_L$ all $h\colon L \to A$ that do not extend to a $g \in \mathcal{F}_K$, and remove from $\mathcal{F}_K$ all g whose restriction $g|_L$ is not in $\mathcal{F}_L$.
(3) Repeat (2), while anything changes.
(4) For each $K \in \binom{X}{\leq k}$, include in $\mathbf{Y}$ a copy of $\mathbf{B}^{\mathcal{F}_K}$ whose elements are denoted by $(K; b)$ where $b\colon \mathcal{F}_K \to B$.
(5) For each $L \subseteq K \in \binom{X}{\leq k}$ and $b\colon \mathcal{F}_L \to B$, identify the elements $(K; b \circ \rho)$ and $(L; b)$ where $\rho\colon \mathcal{F}_K \to F_L$ is the restriction $\rho(g) = g|_L$.

We denote the output $\mathbf{Y}$ by $\kappa_k^{\mathbf{A},\mathbf{B}}(\mathbf{X})$.

In a nutshell the k-consistency reduction is a concatenation of the k-consistency algorithm (steps (1), (2), and (3)) and the universal gadget reduction (steps (4) and (5)). Here, in steps (1), (2), and (3) we are using the standard formulation of the k-consistency algorithm as a winning strategy in the existential k-pebble game. Note that these steps can be turned into a decision algorithm for CSPs by outputting *no* if one of the sets $\mathcal{F}_K$ (and consequently each of them) is empty, and outputting *yes* if all $\mathcal{F}_K$'s are non-empty; this coincides with [9, Algorithm 1 on p.5]. We shall refer to this decision algorithm as the *k-consistency test* to distinguish it from the k-consistency algorithm corresponding to the application of steps (1), (2), and (3). In turn, they should not be confused with the k-consistency reduction defined above. Steps (4) and (5) correspond to the universal gadget reduction; these two steps are a standard way of reducing from Label Cover to another CSP used in the algebraic approach [7, Section 3.3].

[6] It is useful but not necessary to assume that $k \geq m$, where m is the maximal arity of the relations of $\mathbf{A}$, since the procedure below ignores all constraints of arity bigger than k.

Example 3.11. Barto and Kozik [10] provided a sufficient condition for the existence of an efficient reduction between two promise CSPs, which is a weaker version of the k-consistency reduction. Many known NP-hard promise CSPs can be explained by a reduction from graph 3-colouring using this sufficient condition. For example, the approximate hypergraph colouring (first proved to be NP-hard by Dinur, Regev, and Smyth [27]) allows a k-consistency reduction from $\mathrm{CSP}(\mathbf{K}_3)$ as shown by Wrochna [50]. The same approach can also provide the NP-hardness results in [3; 38; 30; 52; 18]. We return to this sufficient condition in the full version [25, Appendix C.2].

The main result of this section is that Datalog$^\cup$ reductions and consistency reductions have the same power in the scope of promise CSPs. In fact we prove a refined version of the statement that relates the parameters of the two reductions.

Theorem 3.12. *Fix $k \geq 1$ and let $\mathbf{A}, \mathbf{A}'$ and $\mathbf{B}, \mathbf{B}'$ be two promise templates. The following are equivalent:*

(1) There exists a Datalog interpretation ϕ of width k and a gadget γ such that $\gamma \circ \phi$ is a reduction from $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$.
(2) $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{k\text{-cons}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$.

We prove the theorem by showing that the k-consistency reduction is essentially a composition of a *canonical Datalog interpretation* and a *universal gadget*. This claim immediately gives one of the two implications. The other implication is proved by showing that both the canonical Datalog interpretation and the universal gadget have a certain universal property. Loosely speaking, the canonical Datalog interpretation of width k can be used in place of any other Datalog interpretation of width k, and the universal gadget can be used in place of any other gadget. This general idea has a lot of subtle details that are dealt with in the technical part of the paper, e.g., what if the output signatures of the Datalog interpretation does not agree with the output signature of the canonical Datalog program?

As a direct corollary of the above theorem, we get that k-consistency reductions are equivalent to Datalog reductions in the following sense.

Corollary 3.13. $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{\mathrm{DL}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *if and only if there exists $k \geq 1$ such that* $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{k\text{-cons}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$.

Proof. By definition, every Datalog$^\cup$ reduction is a composition of a Datalog interpretation ϕ and a union gadget υ, which is equivalently expressed as a gadget reduction. Hence Theorem 3.12 applies

for k equal to the width of ϕ. Consequently, $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{k\text{-cons}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$.

If we assume that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{k\text{-cons}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ for some $k \geq 1$, then the same theorem provides a Datalog interpretation ϕ and a gadget γ whose composition gives a reduction. Since both ϕ and γ are equivalent to Datalog$^\cup$ reductions (the latter by Theorem 3.9), we get that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{\mathsf{DL}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ by composition (Theorem 3.7). □

Example 3.14. Building on Example 3.11, note that the NP-hardness of $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_5)$ follows by a k-consistency reduction from $\mathrm{CSP}(\mathbf{K}_3)$ since

$$\mathrm{CSP}(\mathbf{K}_3) \leq_{k\text{-cons}} \mathrm{PCSP}(\mathbf{H}_2, \mathbf{H}_{17480}) \leq_{\mathsf{DL}} \mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_5)$$

where $\mathrm{PCSP}(\mathbf{H}_2, \mathbf{H}_{17480})$ is the approximate hypergraph colouring problem mentioned in Example 3.11. The first reduction is the one described in Example 3.11 (it is expressible as a Datalog$^\cup$ reduction by the above corollary), and the second reduction is provided by a combination of [7, Theorem 6.5] and Theorem 3.9. Consequently, since Datalog$^\cup$ reductions compose, we get that $\mathrm{CSP}(\mathbf{K}_3) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbf{K}_3, \mathbf{K}_5)$.

Naturally, we could ask for a concrete Datalog interpretation (together with the associated union gadget) that provides the above reduction. Such an interpretation is far from obvious: First, the Datalog$^\cup$ reduction from hypergraph colouring to graph colouring is an unfolding of the gadget reduction provided by the algebraic approach, and hence quite complex to write down explicitly. Second, [10, Theorem 5.1] does not provide an explicit bound on k for the k-consistency reduction from graph colouring to hypergraph colouring (although this bound could be reconstructed from the proof). We also do not know what is the smallest k, s.t., $\mathrm{CSP}(\mathbf{K}_3) \leq_{k\text{-cons}} \mathrm{CSP}(\mathbf{K}_3, \mathbf{K}_5)$.

4 HIERARCHIES AND CSP ALGORITHMS

Several hierarchies of algorithms are studied for the tractability of CSPs and promise CSPs. The most prominent are arguably the local consistency hierarchy, the Sherali–Adams hierarchy [48], and the Lasserre hierarchy of semi-definite programming relaxations. Furthermore, Ciardo and Živný [24] introduced a framework to create more hierarchies through a tensor construction. There is ongoing research aiming to characterise which problems are solvable by some level of such a hierarchy (e.g., Ciardo and Živný [23]).

In this section we provide an alternative look at these hierarchies through the lenses of the k-consistency reduction. In particular, we associate a hierarchy to every (promise) CSP, where the k-th level of the hierarchy consists of all (promise) CSPs that reduce to it by the k-consistency reduction. As we will show below, by choosing adequately the initial (promise) CSP, we recover several of the above hierarchies. In particular, the bounded width hierarchy is the 'consistency hierarchy' associated to the trivial problem, and the Sherali–Adams hierarchy is the 'consistency hierarchy' associated to linear programming. One of the benefits of expressing hierarchies in this way is that, by Theorems 3.9 and 3.7, we immediately get that the consistency hierarchy of a fixed problem is closed under gadget reductions, which in particular means that there is a possibility to describe membership in this hierarchy by the means of polymorphisms. We note that this is not always clear for the analogous tensor hierarchies. Finally, we consider new hierarchies not previously studied, namely, the hierarchy associated to solving systems of linear equations over integers, or over a finite (Abelian) group.

In this section, we work with infinite template CSPs, e.g., we express linear programming as a CSP with domain $\mathbb{Q}$. This creates some difficulties since, if $\mathbf{B}$ is an infinite structure in Definition 3.10, then the output of the k-consistency reduction will be an infinite instance as well. Nevertheless, if $\mathbf{B}$ is one of the infinite templates considered in this section, it is always possible to produce an equivalent, but finite, instance. We provide proofs of this statement for each template separately in the corresponding subsection.

4.1 Bounded width

As a warm-up, we start with the hierarchy of promise CSPs with bounded width. This hierarchy corresponds to the smallest (easiest) class, w.r.t., Datalog$^\cup$ reductions, of promise CSPs. In the above sense, it can be described as the k-consistency hierarchy associated to the trivial problem $\mathrm{CSP}(\bot)$. Traditionally, the problems belonging to this hierarchy are said to be of *bounded width* — a (promise) CSP is said to be of *width at most k* if the k-consistency test solves the problem, or equivalently if it is expressible by a Datalog sentence of width k (as we mentioned before, the proof of the corresponding theorem for standard CSPs by Kolaitis and Vardi [37] applies to the promise setting). We show that this property is also equivalent to having a k-consistency reduction to the trivial problem.

Theorem 4.1. $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ *has width k if and only if it reduces to* $\mathrm{CSP}(\bot)$ *by the k-consistency reduction.*

Proof. If $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ is expressible by a Datalog formula of width k, there is Datalog interpretation ϕ of width k that reduces from $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ to $\mathrm{PCSP}(\bot)$. Consequently, by Theorem 3.12, we get the required k-consistency reduction.

If $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \leq_{k\text{-cons}} \mathrm{CSP}(\bot)$ and $\mathbf{X}$ is accepted by the k-consistency test, then $\kappa_k^{\mathbf{A},\bot}(\mathbf{X}) = \bot$ since $\bot^D = \bot$ for any $D \neq \emptyset$, and hence there are no constraints. Consequently, we get that $\mathbf{X} \to \mathbf{A}'$ by the soundness of the k-consistency reduction. □

While the previous result is somewhat expected there is more than meets the eye as the statement implies that bounded width promise CSP problems are closed under gadget reductions. Although this has been previously shown [7, Lemma 7.5], a direct proof requires a bit of a case analysis.

We also note without a proof that this hierarchy corresponds to the k-consistency hierarchy associated to Horn-SAT.

4.2 Sherali–Adams

There are several slightly different ways to define the Sherali–Adams (SA) relaxation for CSPs and promise CSPs, e.g., [49; 24; 22; 8]. Although they might differ in minor technical details, they are all based on the scheme introduced by Sherali and Adams [48] to generate increasingly tight relaxations of a linear program. In all cases, a CSP instance is turned into an instance of linear programming whose tightness can be adjusted using a parameter k.

Linear programming can be viewed as a fixed template CSP though with infinite domain and infinitely many relations: Namely,

the domain is $\mathbb{Q}$,[7] and the relations are all relations defined by affine inequalities, e.g., of the form

$$a_1x_1 + \cdots + a_nx_n \leq b$$

for some $a_1, \ldots, a_n, b \in \mathbb{Q}$. In an instance of linear programming, the relations are usually given as the tuples of their coefficients. Nevertheless, all of these relations are in fact expressible (more precisely, definable by a *primitive positive formula*) using the following three types of inequalities: $x \leq y$, $x_1 + x_2 = y$, and $y = 1$. With some care, the length of these primitive positive definitions becomes proportionate to the binary encoding of the coefficients. This means that up to some simple gadget reductions, we can define the template of linear programming, denoted $\mathbf{Q}_{\text{conv}}$, as the structure with these three relations.

In this section we shall show that the Sherali–Adams hierarchy coincides with the k-consistency hierarchy associated to $\mathbf{Q}_{\text{conv}}$. While a weaker version of our results (where the hierarchies are only required to interleave) holds for any natural variant of the SA relaxation, we can additionally prove that both hierarchies align perfectly if we choose it suitably. Our definition agrees with the $\text{SA}(k, k)$ system of Thapper and Živný [49, Section 3.1] assuming that k is at least the maximal arity of the constraints.

In plain words, the goal of the k-th level of Sherali–Adams relaxation is to find a collection of probability distributions on partial solutions on each of the subsets of variables of size at most k that have consistent marginals.

Definition 4.2 (k-th level of Sherali–Adams relaxation). Fix $k \geq 1$ and a structure $\mathbf{A}$. The k-th level of Sherali–Adams relaxation with template $\mathbf{A}$ is the mapping that given a structure $\mathbf{X}$ of the same signature as $\mathbf{A}$ produces a linear program in the following way: For each $K \in \binom{X}{\leq k}$, let $\mathcal{F}_K$ be the set of all partial homomorphisms from K to $\mathbf{A}$. The k-th level of Sherali–Adams relaxation, denoted by $\text{SA}^k(\mathbf{X})$, is the following linear program with variables $x_{K,f} \in [0, 1]$ for each $K \in \binom{X}{\leq k}$ and $f \in \mathcal{F}_K$:

$$\sum_{f \in \mathcal{F}_K} x_{K,f} = 1 \qquad \text{for each } K \in \tbinom{X}{\leq k},$$

$$\sum_{f \in \mathcal{F}_K, f|_L = g} x_{K,f} = x_{L,g} \quad \text{for each } L \subset K \in \tbinom{X}{\leq k}, \text{ and } g \in \mathcal{F}_L.$$

Observe that $\text{SA}^k(\mathbf{X})$ has a 0-1 solution if there is a homomorphism $h\colon \mathbf{X} \to \mathbf{A}$; assign $x_{f,K} = 1$ if $f = h|_K$ and $x_{f,K} = 0$ otherwise. This observation provides completeness of the Sherali–Adams relaxation. We say that the k-th level of the Sherali–Adams relaxation *solves* $\text{PCSP}(\mathbf{A}, \mathbf{A}')$ if it is also sound in the sense that if $\mathbf{X} \not\to \mathbf{A}'$, for some instance $\mathbf{X}$ of the promise CSP, then the $\text{SA}^k(\mathbf{X})$ does not have a feasible solution. Having settled on the definition, we formulate the main theorem of this subsection.

Theorem 4.3. $\text{PCSP}(\mathbf{A}, \mathbf{A}')$ *is solvable by the k-th level of the Sherali–Adams relaxation if and only if it reduces to* $\text{CSP}(\mathbf{Q}_{conv})$ *by the k-consistency reduction.*

Let us briefly sketch the proof. The theorem is proven by a step-by-step comparison of the k-consistency reduction and the SA^k. We first note that the system $\mathcal{F}_K$ constructed in SA^k is precisely equivalent to step (1) of the k-consistency procedure. We then show that steps (2) and (3) of the k-consistency are not needed since any solution to the linear program SA^k yields a consistent system; indeed, it is not hard to check that if $x_{K,f}$ is a solution to SA^k, then

$$\mathcal{F}'_K = \{f \in \mathcal{F}_K \mid x_{K,f} > 0\}$$

is a consistent system. This means that replacing $\mathcal{F}_K$ in the construction of the SA^k system with the sets obtained by enforcing k-consistency, i.e., at the end of step (3) of the procedure, does not change the output of SA^k. Next, we show that we can replace the equations in SA^k with the universal gadget for linear programming (as in steps (4) and (5) of the k-consistency) without increasing (or decreasing) the power of the relaxation, i.e., we show that $\text{SA}^k(\mathbf{X})$ has a feasible solution if and only if $\kappa_k^{\mathbf{A},\mathbf{Q}_{\text{conv}}}(\mathbf{X}) \to \mathbf{Q}_{\text{conv}}$. For the 'only if' direction, we can construct, from a feasible solution of $\text{SA}^k(\mathbf{X})$, a homomorphism $h\colon \kappa_k^{\mathbf{A},\mathbf{Q}_{\text{conv}}}(\mathbf{X}) \to \mathbf{Q}_{\text{conv}}$ by letting

$$h([K; q]) = \sum_{f \in \mathcal{F}_K} x_{f,K} q(f) \tag{4.1}$$

where $K \in \binom{X}{\leq k}$ and $q\colon \mathcal{F}_K \to \mathbb{Q}$. It is relatively straightforward to check that h is indeed a homomorphism. For the 'if' direction, starting with a homomorphism $h\colon \kappa_k^{\mathbf{A},\mathbf{Q}_{\text{conv}}}(\mathbf{X}) \to \mathbf{Q}_{\text{conv}}$, we can define a solution to the Sherali–Adams system by

$$x_{K,f} = h([K; e_f]) \tag{4.2}$$

where $e_f\colon \mathcal{F}_K \to \mathbb{Q}$ is defined by $e_f(f) = 1$ and $e_f(g) = 0$ if $g \neq f$. Again, checking that the $x_{K,f}$'s, thus defined, is a feasible solution of $\text{SA}^k(\mathbf{X})$ is rather straightforward.

In the detailed proof [25, Appendix D], we show a more general statement that uses abstract properties of linear programming and its polymorphisms rather than formalising the above argument directly. This argument can be generalised to provide analogues of Theorem 4.3 for other hierarchies. In particular, it can proven that the hierarchy of a *conic minion* $\mathscr{M}$ [24, Definition 3.6] coincides with k-consistency hierarchy for the corresponding promise version of label cover (i.e., the problem sometimes denoted by $\text{PMC}_{\mathscr{M}}$ [7, Definition 3.10]) assuming k is greater or equal than the arity of any relation of the template.

Example 4.4. Ciardo and Živný [23] prove that $\text{PCSP}(\mathbf{K}_c, \mathbf{K}_d)$ cannot be solved by any level of a hierarchy of combined LP and AIP for any $d \geq c \geq 3$ (which we call *LP+AIP hierarchy*). The hierarchy was introduced by Brakensiek, Guruswami, Wrochna, and Živný [17], and can be described as a tensor hierarchy of a conic minion, and hence as a k-consistency hierarchy of a fixed problem for which an analogue of Theorem 4.3 is valid.

The proof that approximate graph colouring is not solved by this hierarchy relies on the line digraph reduction from Example 3.4. First, Ciardo and Živný show that $\text{PCSP}(\mathbf{K}_c, \mathbf{K}_d)$ is not solved by the k-th level of the hierarchy for any $c \geq (k^2 + k)/2$ (we assume $k \geq 2$). Then they use the line digraph construction to reduce a general case to this case.

The reduction from a general case can be explained using our framework as follows: Observe that the line digraph construction δ

[7] Although maybe a more natural choice of the domain would be $\mathbb{R}$, we prefer $\mathbb{Q}$ since it is countable, and hence there is less discussion about encoding of coefficients and solutions.

provides a reduction

$$\mathrm{PCSP}(\mathbf{K}_{b(c)}, \mathbf{K}_{b(d)}) \leq_{\mathrm{DL}} \mathrm{PCSP}(\mathbf{K}_c, \mathbf{K}_d)$$

where $b(n) = \binom{n}{\lfloor n/2 \rfloor}$ [40, Lemma 4.16]. Furthermore, it is easy to observe that for each Datalog interpretation ϕ of width k, $\phi \circ \delta$ is expressible as a Datalog interpretation of width $2k$ (by inspection of our proof of Theorem 3.7). Hence, if $\mathrm{PCSP}(\mathbf{K}_c, \mathbf{K}_d)$ is solvable by the k-th level of the hierarchy, then $\mathrm{PCSP}(\mathbf{K}_{b(c)}, \mathbf{K}_{b(d)})$ is solvable by the $2k$-th level of the hierarchy by Theorem 3.12. Chaining these reductions, we obtain that if $\mathrm{PCSP}(\mathbf{K}_c, \mathbf{K}_d)$ is solvable by the k-th level, then $\mathrm{PCSP}(\mathbf{K}_{b^n(c)}, \mathbf{K}_{b^n(d)})$ is solvable by the $2^n k$-th level. Since b increases exponentially (more precisely $b(c) \geq 2^c/c$), we eventually get that $b^n(c) \geq ((2^n k)^2 + 2^n k)/2$, and we may apply that $\mathrm{PCSP}(\mathbf{K}_{b^n(c)}, \mathbf{K}_{b^n(d)})$ is not solved by the $2^n k$-th level to finish the prove.

Our argument applies to any hierarchy which can be described using k-consistency reductions, which includes all tensor hierarchies of conic minions for which an argument is given in the full version of [23]. We believe that the argument presented here is simpler than the one presented in [23], and moreover, it still holds if one replaces δ with any Datalog$^\cup$ reduction (since the width of the composition of two Datalog interpretations is always bounded by the product of the widths).

Finally, we have the following immediate corollary of Theorems 4.3, 3.9, and 3.12, which we believe has not been shown before in the scope of promise CSPs. The analogous statement in the non-promise setting follows from the characterisation of Sherali–Adams for CSPs [9; 49]. Again, the same can be proven for any hierarchy of a conic minion, and in particular for the Lasserre hierarchy.

Corollary 4.5. *Let* $\mathbf{A}, \mathbf{A}'$ *and* $\mathbf{B}, \mathbf{B}'$ *be two promise templates such that* $\mathrm{Pol}(\mathbf{B}, \mathbf{B}') \to \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$. *If* $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *is solvable by some level of Sherali–Adams hierarchy, then so is* $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$.

An important consequence of the above corollary is that a characterisation of the applicability of the Sherali–Adams algorithm in the scope of promise CSPs could be theoretically described by an algebraic condition on polymorphisms.

4.3 Hierarchies of groups

Solving systems of equations over a (finite) Abelian group is a well-known CSP that cannot be solved by neither the k-consistency nor the Sherali–Adams relaxations. We start with formally defining group CSPs.[8]

Definition 4.6. Let $\mathbb{G}$ be an Abelian group. We use $\mathrm{CSP}(\mathbb{G})$ to denote the following problem: given a systems of equations of the form

$$a_1 x_1 + \cdots + a_n x_n = b \tag{4.3}$$

where $a_1, \ldots, a_n \in \mathbb{Z}$ and $b \in \mathbb{G}$, decide whether it has a feasible solution. This problem can be formulated as the CSP with template $\mathbf{G}$ whose domain is G and relations are $x_1 + x_2 = y$ and $y = b$ for each $b \in \mathbb{G}$. Moreover, it is enough to consider b from a fixed generating set[9] of $\mathbb{G}$ in the relations of the second form.

Similarly, if $\mathbb{G}$ is a finite non-Abelian group, we let $\mathrm{CSP}(\mathbb{G})$ to be the CSP with template $\mathbf{G}$ defined in the same way as above (note that the equational definition is more tricky in this case).

For simplicity, we restrict our investigations to cyclic groups. For several reasons, little generality is loss due to this restriction. First, solving systems of equations over a non-Abelian finite group $\mathbb{G}$, and hence $\mathrm{CSP}(\mathbb{G})$, is NP-complete. Further, every finite Abelian group $\mathbb{G}$ is a product of cyclic groups, and finally, the infinite cyclic group $\mathbb{Z}$ provides a uniform algorithm, the so-called basic affine integer relaxation, for solving $\mathrm{CSP}(\mathbb{G})$ where $\mathbb{G}$ is an Abelian group. In particular, for every finite Abelian group $\mathbb{G}$, $\mathrm{CSP}(\mathbb{G})$ is reducible via a gadget reduction to $\mathrm{CSP}(\mathbb{Z})$ since $\mathbf{G}$ has an *alternating polymorphism* [7, Section 7.3].

In the rest of the section, we assume that $\mathbb{G}$ is a cyclic group generated by an element denoted by 1 (i.e., it is either isomorphic to the group $\mathbb{Z}_n$ of addition modulo $n \in \mathbb{Z}$, or to $\mathbb{Z}$ itself). We introduce the following relaxation.

Definition 4.7. ($\mathbb{G}$-affine k-consistency relaxation) Fix $k \geq 1$ and a structure $\mathbf{A}$. The $\mathbb{G}$-affine k-consistency relaxation with template $\mathbf{A}$ is the mapping that, given a structure $\mathbf{X}$ of the same signature as $\mathbf{A}$, produces a system of equations over $\mathbb{G}$ in the following way: Run the k-consistency to compute sets $\mathcal{F}_K$, i.e., execute steps (1), (2), and (3) in Definition 3.10. Consider the same system of linear equations as in Definition 4.2 except that the variables $x_{K,f}$ are now valued over $\mathbb{G}$.

Recall that it can be decided in polynomial time whether the resulting system of equations has a feasible solution. Also, it can be easily verified that it has a feasible solution whenever there is an homomorphism $h\colon \mathbf{X} \to \mathbf{A}$. In particular, the assignment $x_{K,h|_K} = 1$ extends to a solution of the system by assigning 0 to all other variables.

The $\mathbb{Z}$-affine k-consistency relaxations (for varying k) are similar to relaxations considered by Berkholz and Grohe [13, see Section 2.2]; the system of Berkholz and Grohe is obtained by skipping consistency enforcement, i.e., steps (2) and (3). Even weaker hierarchy of AIP was consider by Ciardo and Živný [24].

The following lemma shows that the $\mathbb{G}$-affine k-consistency relaxation is equivalent to the k-consistency reduction to $\mathrm{CSP}(\mathbb{G})$.

Lemma 4.8. *Let* $\mathbb{G}$ *be a cyclic group. The* $\mathbb{G}$*-affine* k*-consistency relaxation solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ *if and only if* $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ *reduces to* $\mathrm{CSP}(\mathbb{G})$ *by the* k*-consistency reduction.*

Proof sketch. We only need to show that replacing the linear system in Definition 4.7 by steps (4) and (5) of the k-consistency reduction does not yield a more powerful reduction. This is argued in a similar way as in the sketch of the proof of Theorem 4.3. Since $\mathbb{G}$ is either $\mathbb{Z}_n$, or $\mathbb{Z}$, we can assume a ring structure on $\mathbb{G}$. This allows us to use analogous definitions for h and x as in the case of linear programming above, i.e., as in Equations (4.1) and (4.2) replacing $\mathbb{Q}$ by $\mathbb{G}$ in both definitions. □

[8] We note that what our definition of a group CSP differs from Berkholz and Grohe [12; 13]; for Abelian groups this difference does not matter, but the Berkholz–Grohe CSP of a non-Abelian finite group is polynomial time solvable while ours is NP-complete (see Goldmann and Russell [31]).

[9] A set $A \subseteq G$ is said to *generate* a group $\mathbb{G}$ if there is no proper subgroup of $\mathbf{G}$ that contains A.

Obstructions to consistency reductions and a conjecture. In this subsection we shall explore the role of group CSPs as 'obstacles' for k-consistency reductions. A seminal example is the fact that k-consistency does not solve $\mathrm{CSP}(\mathbb{G})$ for any non-trivial finite Abelian group $\mathbb{G}$ [29, Theorem 30], or equivalently, $\mathrm{CSP}(\mathbb{G}) \not\leq_{\mathsf{DL}} \mathrm{CSP}(\bot)$. It then follows that, e.g., $\mathrm{CSP}(\mathbf{K}_3) \not\leq_{\mathsf{DL}} \mathrm{CSP}(\bot)$ since, say, $\mathrm{CSP}(\mathbb{Z}_3)$ reduces to $\mathrm{CSP}(\mathbf{K}_3)$ by a gadget reduction and Datalog$^\cup$ reductions are transitive by Theorem 3.7. The following proposition allows us to push this approach a bit further.

Proposition 4.9. $\mathrm{CSP}(\mathbb{Z}_p) \not\leq_{\mathsf{DL}} \mathrm{CSP}(\mathbb{Z}_q)$ *for any primes* $p \neq q$.

Proof. We start with a k-consistent but unsolvable instance $\mathbf{X}$ of $\mathrm{CSP}(\mathbb{Z}_p)$, i.e., with a system of equations modulo p which is not solvable, but is accepted by the k-consistency test. Such a system is known to exist (see, e.g., [29; 2]). Observe that, after each step in the k-consistency procedure, the sets $\mathcal{F}_K$ are always affine subspaces of $\mathbb{Z}_p^K$ (to begin with they are defined by linear equations, and in each step, we intersect with a projection of another affine subspace). Since the system is consistent, the subspaces are non-empty.

Further observe that the maps $\pi\colon \mathcal{F}_K \to \mathcal{F}_L$ defined by restriction to L are affine, and hence the size of the preimage of an element $f \in \mathcal{F}_L$ under π does not depend on f — it only depends on the dimension of the kernel of π. Hence the uniform probability on $\mathcal{F}_K$, i.e., $\lambda\colon \mathcal{F}_K \to \mathbb{Q}$ defined as $\lambda(f) = 1/p^d$ where d is the dimension of $\mathcal{F}_K$, solves the corresponding Sherali–Adams linear program. Since p and q are coprime, we can interpret the expression $1/p^d$ as an element of $\mathbb{Z}_q$. Consequently, this assignment is a feasible solution of the $\mathbb{Z}_q$-affine k-consistency relaxation with input $\mathbf{X}$. □

We note that the above proposition could be also proved by using the fact that $\mathrm{CSP}(\mathbb{Z}_p)$ is not expressible in fixed-point logic with modulo q rank operators (see, e.g., Holm [33]). This alternative approach has been communicated to us by Anuj Dawar [26] and precedes the proof given here.

It follows from the previous proposition that $\mathrm{CSP}(\mathbf{K}_3) \not\leq_{\mathsf{DL}} \mathrm{CSP}(\mathbb{Z}_2)$. More generally, Ciardo and Živný [23] proved that, for any $c \geq 3$ and $k \geq 2$, $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_c)$ is not solved by the k-th level of LP+AIP hierarchy, which implies that $\mathrm{CSP}(\mathbf{K}_3) \not\leq_{\mathsf{DL}} \mathrm{CSP}(\mathbb{Z})$. Since $\mathrm{CSP}(\mathbf{K}_3)$ allows a gadget reduction from any CSP with a finite template, we can view this lack of a reduction as witnessed by $\mathrm{CSP}(\mathbb{G})$ where $\mathbb{G}$ is not Abelian.

This suggests the following intriguing question: Could it be that group CSPs constitute a *complete* set of obstructions for consistency reductions between finite-template CSPs? More concretely, is it true that for every finite structures $\mathbf{A}$ and $\mathbf{B}$, $\mathrm{CSP}(\mathbf{A}) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbf{B})$ if and only if, for every finite group $\mathbb{G}$, $\mathrm{CSP}(\mathbb{G}) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbf{A})$ implies $\mathrm{CSP}(\mathbb{G}) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbf{B})$ or, alternatively, does every equivalence class of finite-template CSPs up to consistency reductions contain a group CSP? The positive verification in the particular case $\mathbf{B} = \bot$ is essentially the notorious bounded width conjecture of Larose and Zádori [41] confirmed by Barto and Kozik [9]. Although extending this proof to all $\mathbf{B}$ is currently out of reach, we believe that the answer will be affirmative. In particular, we conjecture the following.

Conjecture 4.10. *For every finite structure* $\mathbf{A}$, *either* $\mathrm{CSP}(\mathbb{G}) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbf{A})$ *for a non-Abelian finite group* $\mathbb{G}$, *or* $\mathrm{CSP}(\mathbf{A}) \leq_{\mathsf{DL}} \mathrm{CSP}(\mathbb{Z})$.

The conjecture has an important consequence. If true, it would show that all tractable CSPs are solved by the $\mathbb{Z}$-affine k-consistency relaxation for some $k \geq 1$. Combining results of Section 3 and known results it follows that this algorithm solves correctly all CSPs solvable by the k-consistency algorithm as well as systems of linear equations over a finite Abelian group, i.e., it solves the two prime examples of tractable CSPs.

Let us note that another polynomial-time CSP algorithm coming from the algebraic approach, *few subpowers*, does not compare well with consistency reductions. In particular, the class of CSPs having few subpowers (i.e., the class of CSPs that are solved by this algorithm) contains all CSPs of Abelian groups, and hence infinitely many problems that are incomparable by consistency reductions. We believe that confirming (or disproving) the conjecture in the case of CSPs with a Mal'cev polymorphism, which form a small subclass of few subpowers, would likely lead us close to a complete resolution.

We conclude by observing that several relaxation variants previously defined are at least as powerful as the $\mathbb{Z}$-affine k-consistency relaxation introduced here: the *LP+AIP hierarchy* [17], and *cohomological k-consistency* introduced by Ó Conghaile [44]. The power of these relaxations is not yet well understood even in the case of CSPs. For example, the comparison of the power of cohomological k-consistency with respect to the hierarchy of LP+AIP is not clear. Our conjecture implies that all variants solve exactly the same CSPs, namely, all tractable ones (assuming $\mathsf{P} \neq \mathsf{NP}$).

5 CHARACTERISATION OF THE ARC-CONSISTENCY REDUCTION

In this section, we characterise the applicability of a special case of a Datalog$^\cup$ reduction. Namely, a reduction that is obtained from the k-consistency reduction by replacing enforcing k-consistency with enforcing *arc-consistency* instead. This characterisation can be used together with a result of Barto and Kozik [10] to obtain a new sufficient condition for Datalog$^\cup$ reductions [25, Appendix C.2].

We start with describing the reduction which is a combination of enforcing arc-consistency and the universal gadget reduction.

Definition 5.1 (the arc-consistency reduction). Let $\mathbf{A}$ be a Π-structure, and $\mathbf{B}$ be a Σ-structure. The corresponding *arc-consistency reduction* applied to a Π-structure $\mathbf{X}$ outputs a Σ-structure $\mathbf{Y}$ constructed as follows:

(1) For each variable $v \in X_t$ of type t, set $\mathcal{F}_v = A_t$.
(2) Ensure that for each constraint $(v_1, \dots, v_k) \in R^{\mathbf{X}}$, the sets $\mathcal{F}_{v_i}$ are consistent, i.e., for each i, update $\mathcal{F}_{v_i}$ to be equal to the i-th projection of $R^{\mathbf{A}} \cap (\mathcal{F}_{v_1} \times \cdots \times \mathcal{F}_{v_k})$.
(3) Repeat (2), while anything changes.
(4) For each $v \in X$, include in $\mathbf{Y}$ a copy of $\mathbf{B}^{\mathcal{F}_v}$ whose elements are denoted by $(v; b)$ where $b\colon \mathcal{F}_v \to B$.
(5) For constraint $(v_1, \dots, v_k) \in R^{\mathbf{X}}$, include in $\mathbf{Y}$ a copy of $\mathbf{B}^{C_{v,R}}$ where $C_{v,R} = R^{\mathbf{A}} \cap (\mathcal{F}_{v_1} \times \cdots \times \mathcal{F}_{v_k})$ whose elements are denoted by $((v, R); b)$ where $b\colon C \to B$.
(6) For each constraint as above, each $i \in [k]$, and $b\colon \mathcal{F}_{v_i} \to B$, identify the elements $((v, R); b \circ \pi_i)$ and $(v_i; b)$ where $\pi_i\colon C_{v,R} \to \mathcal{F}_{v_i}$ is the i-th projection.

We denote the resulting structure $\kappa_{\mathrm{arc}}^{\mathbf{A},\mathbf{B}}(\mathbf{X})$.

In order to formulate our main result in this section we recall the basic definitions of the algebraic theory of promise CSPs [7, Section 2]. In this paper we work with abstract minions, as opposed to function minions defined in [7, Definition 2.20].

Definition 5.2. An *(abstract) minion* is a functor $\mathscr{M}$ from finite sets to sets which preserves emptyness of sets. In detail, $\mathscr{M}$ is a mapping that assigns to each finite set X a set $\mathscr{M}^{(X)}$, and to each function $\pi\colon X \to Y$ between two finite sets X, Y, a function $\pi^{\mathscr{M}}\colon \mathscr{M}^{(X)} \to \mathscr{M}^{(Y)}$, such that $1_X^{\mathscr{M}} = 1_{\mathscr{M}^{(X)}}$ (recall that 1_X denotes the identity map on X), and $\pi^{\mathscr{M}} \circ \sigma^{\mathscr{M}} = (\pi \circ \sigma)^{\mathscr{M}}$ for all π and σ where the composition makes sense. Moreover, we require that $\mathscr{M}^{(X)} = \emptyset$ if and only if $X = \emptyset$. When the minion is clear from the context, we write f^π for $\pi^{\mathscr{M}}(f)$.

A *minion homomorphism* from $\mathscr{M}$ to $\mathscr{N}$ is a natural transformation $\xi\colon \mathscr{M} \to \mathscr{N}$, i.e., a collection of maps $\xi_X\colon \mathscr{M}^{(X)} \to \mathscr{N}^{(X)}$, one for each finite set X, such that $\pi^{\mathscr{N}} \circ \xi_X = \xi_Y \circ \pi^{\mathscr{M}}$ for all $\pi\colon X \to Y$.

A different view on a minion homomorphism is to see it as a structure homomorphism where each minion is viewed as a typed structure, with an element f of type X for each $f \in \mathscr{M}^{(X)}$, and a binary relation $R_\pi = \{(f, f^\pi) \mid f \in \mathscr{M}^{(X)}\}$ for each $\pi\colon X \to Y$.

Definition 5.3. The *polymorphism minion* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ of a promise template $\mathbf{A}, \mathbf{B}$ is the minion $\mathscr{M}$ with $\mathscr{M}^{(X)} = \{f\colon \mathbf{A}^X \to \mathbf{B}\}$ where $\pi^{\mathscr{M}}$ is defined by $f_t^\pi(x) = f_t(x \circ \pi)$ for all types t, $x \in X_t^N$, and $\pi\colon N \to N'$. Elements of $\mathscr{M}^{(X)}$ are called *polymorphisms of arity* X.

The characterisation of the arc-consistency reduction uses minion homomorphisms together with the following transformation on minions.

Definition 5.4. Let $\mathscr{M}$ be an abstract minion; we define a minion $\omega(\mathscr{M})$ as follows:

$$\omega(\mathscr{M})^{(X)} = \{(Y, f) \mid Y \subseteq X, f \in \mathscr{M}^{(Y)}\}$$

The minor-taking operation for $\pi\colon X \to Y$, is defined by $(Z, f)^\pi = (\pi(Z), f^{\pi|_Z})$ where $\pi(Z) = \{\pi(x) \mid x \in Z\}$.

Intuitively, we interpret the construction above as 'remembering' for each element of $\mathscr{M}$ on which coordinates it depends. This influences the definition of the minor by taking into account the following argument: if g only depends on coordinates in Y then g^π only depends on coordinates in $\pi(Y)$. The algebraic intuition behind $\omega(\mathscr{M})$ is that this minion is constructed in such a way that it satisfies precisely *balanced minor conditions* (i.e., sets of identities of the form $f^\pi = g$ where π is surjective) satisfied in $\mathscr{M}$.

We note that $\omega(\mathrm{Pol}(\bot))$ is isomorphic to the polymorphism minion of Horn-3SAT; see [7, Example 4.2]. This is related to the fact that arc-consistency test is characterised via gadget reductions to Horn-3SAT [7, Theorem 7.4].

Finally, we can formulate the characterisation of arc-consistency reduction. The proof relies on Theorem 2.4 and its proof [7, Section 3]; the novel ingredient is a connection between ω and the arc-consistency enforcement (see [25, Lemma C.1]).

Theorem 5.5. *Let* $\mathbf{A}, \mathbf{A}'$ *and* $\mathbf{B}, \mathbf{B}'$ *be two promise templates. Then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}') \le_{\mathsf{AC}} \mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *if and only if there is a minion homomorphism* $\omega(\mathrm{Pol}(\mathbf{B}, \mathbf{B}')) \to \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$.

6 CONCLUSION

In the present paper, we introduced a general framework of well-structured polynomial-time computable reductions between (promise) CSPs. We see this theory as a foundational work for further study of the complexity of CSPs and promise CSPs. Apart from the obvious future goal of characterising Datalog$^\cup$ reductions by the means of some mathematical invariants of the problems considered, we enclose a few concrete interesting research directions for future work.

Hierarchies of polynomial time algorithms. The following two questions, which have been resolved for non-promise CSPs, remain open for promise CSPs: (1) a characterisation of promise CSPs of bounded width, and (2) a characterisation of promise CSPs solvable by some level of the Sherali–Adams hierarchy. Both of these classes of problems can be described as a k-consistency hierarchy associated to some problem, and hence it is plausible that our approach might be useful here.

For example, we have observed that $\mathrm{CSP}(\mathbb{G})$ is not solved by any level of the Sherali–Adams hierarchy for any non-trivial group $\mathbb{G}$, and hence no promise CSP that allows a Datalog$^\cup$ reduction from one of these problems can be solved by some level of the Sherali–Adams hierarchy. Is the converse true, i.e., is $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ solved by some level of the Sherali–Adams hierarchy if and only if $\mathrm{CSP}(\mathbb{G}) \not\le_{\mathsf{DL}} \mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ for all non-trivial groups $\mathbb{G}$?

Fragments of Datalog$^\cup$ reductions. We have aimed to define a class of reductions that is as expressive as possible while remaining polynomial-time computable. Nevertheless, there are fragments of our reductions that are worth studying for several reasons. Naturally, it is useful if this fragment is expressive enough to be able to emulate gadget reductions, and that it composes similarly to Datalog$^\cup$ reductions.

For example, we could use a log-space computable fragment of Datalog, called *symmetric Datalog*, which has been introduced by Egri, Larose, and Tesson [28], and is able to express reflexive symmetric transitive closure which is the essential part of proving that gadget reductions are expressible as Datalog$^\cup$ reductions. In view of the discussion in Kazda [36, Section 8], these *symmetric Datalog$^\cup$ reductions* could be a good approximation for all log-space reductions between (promise) CSPs.

Descriptive complexity of CSPs. We mentioned some results on the descriptive complexity of CSPs in passing: Atserias, Bulatov, and Dawar [2] proved that a finite-template CSP is expressible in the *fixed-point logic with counting operators* FPC if and only if it has bounded width, and Holm [33] showed that $\mathrm{CSP}(\mathbb{Z}_p)$ is not expressible in the *fixed-point logic with rank mod q operator* FPR_q for any two distinct primes p and q. We may ask which finite-template (promise) CSPs are expressible in these logics in general?

It can be verified that if a (promise) CSP is expressible in a logic that extends Datalog, then all the promise CSPs that reduce to it via a Datalog$^\cup$ reduction are as well. For example, all the CSPs solvable by some level of the $\mathbb{Z}_2$-affine k-consistency relaxation are expressible in FPR_2. Is the converse true? Could the converse be true in the scope of promise CSPs? Naturally, we can ask the same questions for any prime in place of 2.

REFERENCES

[1] Sanjeev Arora and Shmuel Safra. 1998. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM* 45, 1 (1998), 70–122. https://doi.org/10.1145/273865.273901

[2] Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. 2009. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.* 410, 18 (2009), 1666–1683. https://doi.org/10.1016/j.tcs.2008.12.049

[3] Per Austrin, Venkatesan Guruswami, and Johan Håstad. 2017. $(2+\epsilon)$-Sat Is NP-hard. *SIAM Jounal on Computing* 46, 5 (2017), 1554–1573. https://doi.org/10.1137/15M1006507

[4] Libor Barto, Diego Battistelli, and Kevin M. Berg. 2021. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021 (LIPIcs, Vol. 187)*. Schloss Dagstuhl – LZI, Saarbrücken, Germany, 10:1–10:16. https://doi.org/10.4230/LIPIcs.STACS.2021.10 arXiv:2010.04623

[5] Libor Barto, Bertalan Bodor, Marcin Kozik, Antoine Mottet, and Michael Pinsker. 2023. Symmetries of Graphs and Structures that Fail to Interpret a Finite Thing. In *LICS*. 1–13. https://doi.org/10.1109/LICS56636.2023.10175732

[6] Libor Barto, Zarathustra Brady, Andrei Bulatov, Marcin Kozik, and Dmitriy Zhuk. 2021. Minimal Taylor Algebras as a Common Framework for the Three Algebraic Approaches to the CSP. In *LICS*. IEEE, 1–13. https://doi.org/10.1109/LICS52264.2021.9470557

[7] Libor Barto, Jakub Bulín, Andrei Krokhin, and Jakub Opršal. 2021. Algebraic Approach to Promise Constraint Satisfaction. *J. ACM* 68, 4 (2021), 28:1–66. https://doi.org/10.1145/3457606

[8] Libor Barto and Silvia Butti. 2022. Weisfeiler-Leman Invariant Promise Valued CSPs. In *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel (LIPIcs, Vol. 235)*. Schloss Dagstuhl – LZI, 4:1–4:17. https://doi.org/10.4230/LIPIcs.CP.2022.4

[9] Libor Barto and Marcin Kozik. 2014. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J. ACM* 61, 1 (2014). https://doi.org/10.1145/2556646

[10] Libor Barto and Marcin Kozik. 2022. Combinatorial Gap Theorem and Reductions between Promise CSPs. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Virtual Conference, 1204–1220. https://doi.org/10.1137/1.9781611977073.50

[11] Libor Barto, Andrei Krokhin, and Ross Willard. 2017. Polymorphisms, and How to Use Them. In *The Constraint Satisfaction Problem: Complexity and Approximability*, Andrei Krokhin and Stanislav Živný (Eds.). Dagstuhl Follow-Ups, Vol. 7. Schloss Dagstuhl – LZI, Dagstuhl, Germany, 1–44. https://doi.org/10.4230/DFU.Vol7.15301.1

[12] Christoph Berkholz and Martin Grohe. 2015. Limitations of Algebraic Approaches to Graph Isomorphism Testing. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9134)*, Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (Eds.). Springer, 155–166. https://doi.org/10.1007/978-3-662-47672-7_13

[13] Christoph Berkholz and Martin Grohe. 2017. Linear Diophantine Equations, Group CSPs, and Graph Isomorphism. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017* (Barcelona, Spain, Hotel Porta Fira, January 16–19), Philip N. Klein (Ed.). SIAM, 327–339. https://doi.org/10.1137/1.9781611974782.21

[14] Manuel Bodirsky and Albert Vucaj. 2020. Two-element structures modulo primitive positive constructability. *Algebra universalis* 81, 2 (2020), 1–17.

[15] Mikolaj Bojanczyk and Bartek Klin. 2024. Polyregular Functions on Unordered Trees of Bounded Height. *Proc. ACM Program. Lang.* 8, POPL (2024), 1326–1351. https://doi.org/10.1145/3632887

[16] Joshua Brakensiek and Venkatesan Guruswami. 2021. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM Jounal on Computing* 50, 6 (2021), 1663–1700. https://doi.org/10.1137/19M128212X

[17] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. 2020. The Power of the Combined Basic Linear Programming and Affine Relaxation for Promise Constraint Satisfaction Problems. *SIAM J. Comput.* 49, 6 (2020), 1232–1248. https://doi.org/10.1137/20M1312745

[18] Alex Brandts, Marcin Wrochna, and Stanislav Živný. 2021. The Complexity of Promise SAT on Non-Boolean Domains. *ACM Trans. Comput. Theory* 13, 4 (2021). https://doi.org/10.1145/3470867

[19] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. 2005. Classifying the Complexity of Constraints using Finite Algebras. *SIAM J. Comput.* 34, 3 (2005), 720–742. https://doi.org/10.1137/S0097539700376676

[20] Andrei A. Bulatov. 2017. A Dichotomy Theorem for Nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, Berkeley, CA, USA, 319–330. https://doi.org/10.1109/FOCS.2017.37

[21] Andrei A. Bulatov, Andrei A. Krokhin, and Benoît Larose. 2008. Dualities for Constraint Satisfaction Problems. In *Complexity of Constraints – An Overview of Current Research Themes [Result of a Dagstuhl Seminar] (Lecture Notes in Computer Science, Vol. 5250)*, Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer (Eds.). Springer, 93–124. https://doi.org/10.1007/978-3-540-92800-3_5

[22] Silvia Butti and Victor Dalmau. 2021. The Complexity of the Distributed Constraint Satisfaction Problem. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16–19, 2021, Saarbrücken, Germany (LIPIcs, Vol. 187)*. Schloss Dagstuhl – LZI, 20:1–20:18. https://doi.org/10.4230/LIPIcs.STACS.2021.20

[23] Lorenzo Ciardo and Stanislav Živný. 2023. Approximate Graph Colouring and the Hollow Shadow. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20–23, 2023*. ACM, 623–631. https://doi.org/10.1145/3564246.3585112 arXiv:2211.03168

[24] Lorenzo Ciardo and Stanislav Živný. 2023. Hierarchies of Minion Tests for PCSPs through Tensors. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. SIAM, 568–580. https://doi.org/10.1137/1.9781611977554.ch25 arXiv:2207.02277

[25] Víctor Dalmau and Jakub Opršal. 2023. Local consistency as a reduction between constraint satisfaction problems. (2023). https://doi.org/10.48550/arXiv.2301.05084 arXiv:2301.05084v3

[26] Anuj Dawar. 2022. Personal communication. (March 2022). Cambridge, UK.

[27] Irit Dinur, Oded Regev, and Clifford Smyth. 2005. The Hardness of 3-Uniform Hypergraph Coloring. *Combinatorica* 25, 5 (2005), 519–535. https://doi.org/10.1007/s00493-005-0032-4

[28] László Egri, Benoît Larose, and Pascal Tesson. 2007. Symmetric Datalog and Constraint Satisfaction Problems in Logspace. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*. IEEE Computer Society, 193–202. https://doi.org/10.1109/LICS.2007.47

[29] Tomás Feder and Moshe Y. Vardi. 1998. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory. *SIAM J. Comput.* 28, 1 (1998), 57–104. https://doi.org/10.1137/S0097539794266766

[30] Miron Ficak, Marcin Kozik, Miroslav Olšák, and Szymon Stankiewicz. 2019. Dichotomy for symmetric Boolean PCSPs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP'19) (LIPIcs, Vol. 132)*. Schloss Dagstuhl – LZI, Wadern, 57:1–12. https://doi.org/10.4230/LIPIcs.ICALP.2019.57

[31] Mikael Goldmann and Alexander Russell. 2002. The Complexity of Solving Equations over Finite Groups. *Information and Computation* 178, 1 (2002), 253–262. https://doi.org/10.1006/inco.2002.3173

[32] Venkatesan Guruswami and Sai Sandeep. 2020. d-To-1 Hardness of Coloring 3-Colorable Graphs with $O(1)$ Colors. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (LIPIcs)*. Schloss Dagstuhl–LZI, Dagstuhl, Germany, 62:1–62:12. https://doi.org/10.4230/LIPIcs.ICALP.2020.62

[33] Bjarki Holm. 2011. *Descriptive complexity of linear algebra*. Ph. D. Dissertation. University of Cambridge, UK. http://bjarkiholm.com/publications/Holm_2010_phd-thesis.pdf

[34] Sangxia Huang. 2013. Improved Hardness of Approximating Chromatic Number. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. APPROX/RANDOM 2013* (Berkeley, CA, USA) *(Lecture Notes in Computer Science, Vol. 8096)*. Springer, 233–243. https://doi.org/10.1007/978-3-642-40328-6_17

[35] Peter Jeavons, David A. Cohen, and Marc Gyssens. 1997. Closure properties of constraints. *J. ACM* 44, 4 (1997), 527–548. https://doi.org/10.1145/263867.263489

[36] Alexandr Kazda. 2018. n-permutability and linear Datalog implies symmetric Datalog. *Log. Methods Comput. Sci.* 14, 2 (2018). https://doi.org/10.23638/LMCS-14(2:3)2018 arXiv:1508.05766

[37] Phokion G. Kolaitis and Moshe Y. Vardi. 2000. Conjunctive-Query Containment and Constraint Satisfaction. *J. Comput. System Sci.* 61, 2 (2000), 302–332. https://doi.org/10.1006/jcss.2000.1713

[38] Andrei Krokhin and Jakub Opršal. 2019. The complexity of 3-colouring H-colourable graphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS'19)*. IEEE, Baltimore, MD, USA, 1227–1239. https://doi.org/10.1109/FOCS.2019.00076

[39] Andrei Krokhin and Jakub Opršal. 2022. An invitation to the promise constraint satisfaction problem. *ACM SIGLOG News* 9, 3 (2022), 30–59. https://doi.org/10.1145/3559736.3559740 arXiv:2208.13538

[40] Andrei A. Krokhin, Jakub Opršal, Marcin Wrochna, and Stanislav Živný. 2023. Topology and adjunction in promise constraint satisfaction. *SIAM J. Comput.* 52, 1 (2023), 38–79. https://doi.org/10.1137/20M1378223 arXiv:2003.11351

[41] Benoit Larose and László Zádori. 2007. Bounded width problems and algebras. *Algebra universalis* 56, 3 (2007), 439–466. https://doi.org/10.1007/s00012-007-2012-6

[42] Antoine Mottet. 2024. Promise and Infinite-Domain Constraint Satisfaction. In *32nd EACSL Annual Conference on Computer Science Logic, CSL 2024* (Naples, Italy, February 19–23) *(LIPIcs, Vol. 288)*, Aniello Murano and Alexandra Silva (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:19. https://doi.org/10.4230/LIPICS.CSL.2024.41

[43] Tamio-Vesa Nakajima and Stanislav Živný. 2023. Boolean symmetric vs. functional PCSP dichotomy. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'23)* (Boston, MA, USA). IEEE, 1–12. https://doi.org/10.1109/

LICS56636.2023.10175746 arXiv:2210.03343

[44] Adam Ó Conghaile. 2022. Cohomology in Constraint Satisfaction and Structure Isomorphism. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS'22) (LIPIcs, Vol. 241)*. Schloss Dagstuhl – LZI, 75:1–75:16. https://doi.org/10.4230/LIPIcs.MFCS.2022.75

[45] Aleš Pultr. 1970. The right adjoints into the categories of relational systems. In *Reports of the Midwest Category Seminar IV (Lecture Notes in Mathematics, Vol. 137)*. Springer, 100–113. https://doi.org/10.1007/BFb0060437

[46] Francesca Rossi, Peter van Beek, and Toby Walsh (Eds.). 2006. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence, Vol. 2. Elsevier. https://www.sciencedirect.com/science/bookseries/15746526/2

[47] Thomas J. Schaefer. 1978. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho (Eds.). ACM, 216–226. https://doi.org/10.1145/800133.804350

[48] Hanif D. Sherali and Warren P. Adams. 1990. A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM J. Discret. Math.* 3, 3 (1990), 411–430. https://doi.org/10.1137/0403036

[49] Johan Thapper and Stanislav Živný. 2017. The Power of Sherali-Adams Relaxations for General-Valued CSPs. *SIAM J. Comput.* 46, 4 (2017), 1241–1279. https://doi.org/10.1137/16M1079245

[50] Marcin Wrochna. 2022. A note on hardness of promise hypergraph colouring. (2022). https://doi.org/10.48550/arXiv.2205.14719

[51] Marcin Wrochna. 2022. Notes on consistency (in minor conditions and minions). (March 2022). unpublished manuscript.

[52] Marcin Wrochna and Stanislav Živný. 2020. Improved hardness for H-colourings of G-colourable graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*. SIAM, Salt Lake City, UT, USA, 1426–1435. https://doi.org/10.1137/1.9781611975994.86

[53] Dmitriy Zhuk. 2020. A Proof of the CSP Dichotomy Conjecture. *J. ACM* 67, 5 (2020), 30:1–30:78. https://doi.org/10.1145/3402029