23rd Annual IEEE Symposium on Logic in Computer Science

Caterpillar duality for constraint satisfaction problems

Catarina Carvalho
Durham University
Durham, DH1 3LE, UK
c.a.carvalho@durham.ac.uk

Víctor Dalmau
University Pompeu Fabra
Barcelona, 08003 Spain
victor.dalmau@upf.edu

Andrei Krokhin
Durham University
Durham, DH1 3LE, UK
andrei.krokhin@durham.ac.uk

Abstract

The study of constraint satisfaction problems definable in various fragments of Datalog has recently gained considerable importance. We consider constraint satisfaction problems that are definable in the smallest natural recursive fragment of Datalog - monadic linear Datalog with at most one EDB per rule. We give combinatorial and algebraic characterisations of such problems, in terms of caterpillar dualities and lattice operations, respectively. We then apply our results to study graph $\mathbf{H}$*-colouring problems.*

1 Introduction

The constraint satisfaction problem (CSP) provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in artificial intelligence and computer science. A constraint satisfaction problem is represented by a set of variables, a domain of values for each variable, and a set of constraints between variables. The aim in a constraint satisfaction problem is then to find an assignment of values to the variables that satisfies the constraints.

It is well known (see, e.g., [5, 11, 21]) that the constraint satisfaction problem can be recast as the following fundamental problem: given two finite relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism from $\mathbf{A}$ to $\mathbf{B}$? The CSP is **NP**-complete in general, and the identifying of its subproblems that have lower complexity has been a very active research direction in the last decade (see, e.g., [5, 11, 13, 21, 23]). One of the most studied restrictions on the CSP is when the structure $\mathbf{B}$ is fixed, and only $\mathbf{A}$ is part of the input. The obtained problem is denoted by $\mathrm{CSP}(\mathbf{B})$. Examples of such problems include k-SAT, GRAPH H-COLOURING, and SYSTEMS OF EQUATIONS (e.g., linear equations).

A variety of mathematical approaches to study problems $\mathrm{CSP}(\mathbf{B})$ has been recently suggested. The most advanced approaches use logic, combinatorics, universal algebra, and their combinations (see [4, 5, 20, 21]). The logic programming language Datalog and its fragments are arguably some of the most important tools for solving CSPs. In fact, all problems $\mathrm{CSP}(\mathbf{B})$ that are known to be tractable can be solved via Datalog, or via the "few subpowers property" [18], or via a combination of the two. Furthermore, for every problem $\mathrm{CSP}(\mathbf{B})$ that is currently known to belong to **NL** or to **LOGSPACE**, the complement of $\mathrm{CSP}(\mathbf{B})$ can be defined in linear Datalog and symmetric Datalog, respectively (see [4, 6, 9]). The algebraic approach to the CSP was recently linked with non-definability in the above fragments of Datalog [23].

The definability of $\mathrm{CSP}(\mathbf{B})$ in Datalog and its fragments is very closely related with homomorphism dualities (see, e.g., [4]) that were much studied in the context of graph homomorphisms (see [15]). Roughly, a structure $\mathbf{B}$ has duality (of some type) if the non-existence of a homomorphism from a given structure $\mathbf{A}$ to $\mathbf{B}$ can always be explained by the existence of a simple enough obstruction structure (i.e., one that homomorphically maps to $\mathbf{A}$ but not to $\mathbf{B}$). The types of dualities correspond to interpretations of the phrase "simple enough". The most important duality is probably bounded treewidth duality which is equivalent to definability in Datalog (see [4]). However, structures with this property are not (yet) characterised, and this property is not even known to be decidable. More success has been obtained for some particular cases of bounded treewidth duality such as finite duality [22, 25] and tree duality [7, 11]. Both of these properties are known to be decidable, and have nice logical, combinatorial, and algebraic characterisations (see the above papers or [4]). For example, they correspond to definability in first-order logic and in monadic Datalog, respectively. The simplest of trees are paths, and the concept of path duality was also much used to study graph homomorphism (see [16, 17]), and in this paper we argue that, in the setting of general relational structures, a slightly more general notion of *caterpillar* duality is more natural, and we give concise logical, combinatorial, and algebraic characterisations of structures having this form of duality (see Section 3).

The problems $\mathrm{CSP}(\mathbf{B})$ with $\mathbf{B}$ being a digraph $\mathbf{H}$ are

1043-6871/08 $25.00 © 2008 IEEE
DOI 10.1109/LICS.2008.19

actively studied in graph theory under the name of $\mathbf{H}$-colouring [15]. Recently, algebraic and logical approaches to the CSP were applied to solve well-known open problems (or to give short proofs of known results) about $\mathbf{H}$-colouring (see, e.g., [1, 2, 3]). We also apply our findings to obtain new results about $\mathbf{H}$-colouring in Section 4.

2 Preliminaries

2.1 Basic definitions

A *vocabulary* is a finite set of relation symbols or predicates. In what follows, τ always denotes a vocabulary. Every relation symbol R in τ has an *arity* $r = \rho(R) > 0$ associated to it. We also say that R is an r-ary relation symbol. A τ-structure $\mathbf{A}$ consists of a set A, called the *universe* of $\mathbf{A}$, and a relation $R^{\mathbf{A}} \subseteq A^r$ for every relation symbol $R \in \tau$ where r is the arity of R. All structures in this paper are assumed to be *finite*, i.e., structures with a finite universe. Throughout the paper we use the same boldface and slanted capital letters to denote a structure and its universe, respectively.

A *homomorphism* from a τ-structure $\mathbf{A}$ to a τ-structure $\mathbf{B}$ is a mapping $h : A \to B$ such that for every r-ary $R \in \tau$ and every $(a_1, \ldots, a_r) \in R^{\mathbf{A}}$, we have $(h(a_1), \ldots, h(a_r)) \in R^{\mathbf{B}}$. We denote this by $h : \mathbf{A} \to \mathbf{B}$. We also say that $\mathbf{A}$ homomorphically maps to $\mathbf{B}$, and write $\mathbf{A} \to \mathbf{B}$ if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ and $\mathbf{A} \not\to \mathbf{B}$ if there is no homomorphism. Now CSP($\mathbf{B}$) can be defined to be the class of all structures $\mathbf{A}$ such that $\mathbf{A} \to \mathbf{B}$. The class of all structures $\mathbf{A}$ such that $\mathbf{A} \not\to \mathbf{B}$ will be denoted by co-CSP($\mathbf{B}$). A number of examples of combinatorial problems representable as CSP($\mathbf{B}$) or co-CSP($\mathbf{B}$) for a suitable structure $\mathbf{B}$ can be found in [4, 5, 21].

A *retract* of a structure $\mathbf{B}$ is an induced substructure $\mathbf{B}'$ of $\mathbf{B}$ such that there is a homomorphism $h : \mathbf{B} \to \mathbf{B}'$ satisfying $h(b) = b$ for all $b \in \mathbf{B}'$. A structure is a *core* if it has no proper retracts, and a *core of a structure* is its retract that is a core. It is well known that all cores of a structure are isomorphic, so we will call any structure isomorphic to a core of $\mathbf{B}$ *the core* (of $\mathbf{B}$), denoted core($\mathbf{B}$). Obviously, the problems CSP($\mathbf{B}$) and CSP(core($\mathbf{B}$)) coincide.

We will now define structures that play an important role in this paper - trees and caterpillars, which are natural generalisations of the corresponding notions from graph theory. Let $\mathbf{A}$ be a τ-structure. As in [25], the *incidence multigraph* of $\mathbf{A}$, denoted $Inc(\mathbf{A})$, is defined as the bipartite multigraph with parts A and $Block(\mathbf{A})$, where $Block(\mathbf{A})$ consists of all pairs $(R, \overline{a})$ such that $R \in \tau$ and $\overline{a} \in R^{\mathbf{A}}$, and with edges $e_{a,i,Z}$ joining $a \in A$ to $Z = (R, (a_1, \ldots, a_r)) \in Block(\mathbf{A})$ when $a_i = a$. A structure $\mathbf{A}$ is said to be a *τ-tree* (or simply a *tree*) if its incidence multigraph is a tree (in particular, it has no multiple edges). For a τ-tree $\mathbf{A}$, we say that an element of A is a *leaf* if it is incident to exactly one block in $Inc(\mathbf{A})$. A block of $\mathbf{A}$ (i.e., a member of $Block(\mathbf{A})$) is said to be *pendant* if it is incident to at most one non-leaf element, and it is said to be *non-pendant* otherwise. For example, any block with a unary relation is always pendant.

In graph theory, a caterpillar is a tree which becomes a path after all its leaves are removed. Following [24], we say that a τ-tree is a *τ-caterpillar* (or simply a caterpillar) if each of its blocks is incident to at most two non-leaf elements, and each element is incident to at most two non-pendant blocks. Informally, a τ-caterpillar has a body consisting of a chain of elements $a_1, \ldots, a_{n+1}$ with blocks $B_1, \ldots, B_n$ where B_i is incident to a_i and a_{i+1} ($i = 1, \ldots, n$), and legs of two types: (i) pendant blocks incident to exactly one of the elements $a_1, \ldots, a_{n+1}$, together with some leaf elements incident to such blocks, and (ii) leaf elements incident to exactly one the blocks $B_1, \ldots, B_n$.

Example 1. *(i) If τ is the signature of digraphs then the τ-caterpillars are the* oriented caterpillars*, i.e., digraphs obtained from caterpillar graphs by orienting each edge in some way.*

(ii) Let $\mathbf{B}$ be a structure with $B = \{1, 2, \ldots, 6\}$, one unary relation $R_1 = \{2, 3\}$, one binary relation $R_2 = \{(1, 2), (2, 3), (3, 6)\}$ and one ternary relation $R_3 = \{(3, 4, 5)\}$. The graph $Inc(\mathbf{B})$ is shown on Fig. 1. The elements 2 and 3 are the non-leaves, and $(R_2, (2, 3))$ is the only non-pendant block. In particular, $\mathbf{B}$ is a caterpillar.

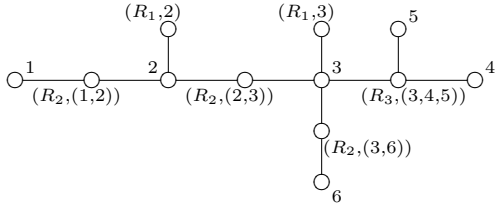

Figure 1. A caterpillar.

2.2 Datalog

We now briefly describe the basics of Datalog (for more details, see, e.g., [8, 19]). Fix a vocabulary τ. A *Datalog program* is a finite set of rules of the form $t_0 :- t_1, \ldots, t_n$ where each t_i is an atomic formula $R(x_{i_1}, \ldots, x_{i_k})$. Then t_0 is called the *head* of the rule, and the sequence $t_1, \ldots, t_n$ the *body* of the rule. The intended meaning of such a rule is that the conjunction of the predicates in the body implies the predicate in the head, with all variables not appearing in the head existentially quantified. The predicates occurring in the heads of the rules are not from τ and are called *IDBs* (from "intensional database predicates"), while all other predicates come from τ and are called *EDBs* (from

"extensional database predicates"). One of the IDBs, which is usually 0-ary in our case, is designated as the *goal predicate* of the program. Since the IDBs may occur in the bodies of rules, each Datalog program is a recursive specification of the IDBs, with semantics obtained via least fixed-points of monotone operators. The goal predicate is assumed to be initially set to `false`, and we say that a Datalog program *accepts* a τ-structure $\mathbf{A}$ if its goal predicate evaluates to `true` on $\mathbf{A}$. In this case we also say that the program *derives* the goal predicate on $\mathbf{A}$. It is easy to see that the class of structures accepted by any Datalog program is closed under homomorphism (i.e., if $\mathbf{A} \to \mathbf{B}$ and $\mathbf{A}$ is accepted then $\mathbf{B}$ is also accepted).

A Datalog program is called *linear* if each of its rules has at most one occurrence of an IDB in its body, and it is called *monadic* if each IDB in it is at most unary.

When using Datalog to study CSP($\mathbf{B}$), one speaks of the definability of co-CSP($\mathbf{B}$) in Datalog (or its fragments). Examples of Datalog programs defining classes of the form co-CSP($\mathbf{B}$) can be found in [4, 6, 20].

2.3 Polymorphisms

Let f be an n-ary operation on B, and R a relation on B. We say that f is a *polymorphism* of R if, for any tuples, $\bar{a}_1, \ldots, \bar{a}_n \in R$, the tuple obtained by applying f componentwise to $\bar{a}_1, \ldots, \bar{a}_n$ also belongs to R. In this case we also say that R is invariant under f.

We say that f is a *polymorphism of* $\mathbf{B}$ if it is a polymorphism of each relation in $\mathbf{B}$. It is easy to check that the n-ary polymorphisms of $\mathbf{B}$ are precisely the homomorphisms from the n-th direct power $\mathbf{B}^n$ to $\mathbf{B}$. It is well known and easy to verify that composition of polymorphisms of $\mathbf{B}$ is again a polymorphism of $\mathbf{B}$ (see, e.g., [5]).

The notion of a polymorphism plays the key role in the algebraic approach to the CSP. The polymorphisms of a (core) structure are known to determine the complexity of CSP($\mathbf{B}$) as well as definability of (the complement of) CSP($\mathbf{B}$) in Datalog and the following fragments: monadic, linear, symmetric (see [4, 23]). Many algebraic sufficient conditions for definability of co-CSP($\mathbf{B}$) in various fragments of Datalog are known (see [4]).

Let us now define several types of operations that will be used in this paper. An n-ary operation f on B is called *idempotent* if it satisfies the identity $f(x, \ldots, x) = x$, and it is called *conservative* if $f(x_1, \ldots, x_n) \in \{x_1, \ldots, x_n\}$ for all $x_1, \ldots, x_n \in B$.

An n-ary operation f is called *totally symmetric* if $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ whenever $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$.

A ternary operation f is called a *majority* operation if it satisfies $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$ for all x, y.

A binary associative commutative idempotent operation is called a *semilattice* operation. We will say that two binary operations f and g on B are *lattice* operations if each of them is a semilattice operation and, in addition, they satisfy the absorption identities: $f(x, g(x, y)) = g(x, f(x, y)) = x$. It is well known (see, e.g., [12]) that semilattice operations are in one-to-one correspondence with partial orders in which every two elements have a greatest lower bound (which is the result of applying the operation). Similarly, lattice operations are in one-to-one correspondence with partial orders in which every two elements have both a greatest lower bound and a least upper bound. The simplest example of lattice operations are the operations $\min$ and $\max$ with respect to any fixed linear order on B.

2.4 Dualities

A comprehensive treatment of dualities for the CSP can be found in the survey [4].

Definition 2. *A set $\mathcal{O}$ of τ-structures is called an* obstruction set *for $\mathbf{B}$ if, for any τ-structure $\mathbf{A}$, $\mathbf{A} \to \mathbf{B}$ if and only if $\mathbf{A}' \not\to \mathbf{A}$ for all $\mathbf{A}' \in \mathcal{O}$.*

If the set $\mathcal{O}$ can be chosen to consist of nicely behaved structures such as paths, caterpillars, trees, or structures of bounded pathwidth or of bounded treewidth, then $\mathbf{B}$ is said to have path (caterpillar, tree, bounded pathwidth, bounded treewidth, respectively) duality. A structure with a finite obstruction set is said to have finite duality.

It is known (see [4]) that a structure $\mathbf{B}$ has one of the following forms of duality: finite, tree, bounded pathwidth, bounded treewidth if and only if co-CSP($\mathbf{B}$) is definable in the following fragments of Datalog, respectively: recursion-free, monadic, linear, full.

Structures with tree duality were characterised in several equivalent ways in [11]. To state the result, we need the following construction: for a τ-structure $\mathbf{B}$, define a τ-structure $\mathbf{U}(\mathbf{B})$ whose elements are the non-empty subsets of B, and, for each r-ary $R \in \tau$, we have $(A_1, \ldots, A_r) \in R^{\mathbf{U}(\mathbf{B})}$ if and only if, for each $j = 1, \ldots, r$ and each $a \in A_j$, there exists $(a_1, \ldots, a_r) \in R^{\mathbf{B}}$ such that $a_j = a$.

Theorem 3. *[11, 7] Let $\mathbf{B}$ be a structure. The following conditions are equivalent:*

1. *$\mathbf{B}$ has tree duality;*
2. *co-CSP($\mathbf{B}$) is definable by a monadic Datalog program with at most one EDB per rule;*
3. *$\mathbf{U}(\mathbf{B})$ admits a homomorphism to $\mathbf{B}$;*
4. *for every $n \geq 1$, $\mathbf{B}$ has an n-ary totally symmetric polymorphism.*

If $\mathbf{B}$ *is a core then the above conditions are equivalent to the following:*

5. $\mathbf{B}$ *is the core of a structure with a semilattice polymorphism.*

It is known that any structure with finite duality has a finite obstruction set consisting of trees [25]; such structures are charactersied in many equivalent ways in [22]. The situation when these trees can be chosen to be caterpillars was considered in [24].

Theorem 4 ([24])**.** *Let* $\mathbf{B}$ *be a core structure with finite duality. Then* $\mathbf{B}$ *has an obstruction set consisting of caterpillars if and only if* $\mathbf{B}$ *has a majority polymorphism.*

3 Caterpillar duality

3.1 A characterisation

The main result of this paper is a characterisation of structures with caterpillar duality in the spirit of Theorem 3. First, we need to give some definitions.

Let k, n be positive integers. We call a (kn)-ary operation f on B *k-block symmetric* if it satisfies the following condition:

$$f(x_{11}, \dots, x_{1k}, \dots, x_{n1}, \dots, x_{nk}) = \\ = f(y_{11}, \dots, y_{1k}, \dots, y_{n1}, \dots, y_{nk})$$

whenever $\{S_1, \dots, S_n\} = \{T_1, \dots, T_n\}$ where, for all i, $S_i = \{x_{i1}, \dots, x_{ik}\}$ and $T_i = \{y_{i1}, \dots, y_{ik}\}$. Note that if $k = 1$ or $n = 1$ then f is totally symmetric.

We will often use the following notation for k-block symmetric operations. For (not necessarily distinct) subsets $S_1, \dots, S_n$ of B, with at most k elements each, let $f(S_1, S_2, \dots, S_n)$ denote $f(x_{11}, \dots, x_{1k}, \dots, x_{n1}, \dots, x_{nk})$ where $S_i = \{x_{i1}, \dots, x_{ik}\}$ for all i. Also, for $l \leq n$, let $f(S_1, \dots, S_l)$ denote $f(S_1, \dots, S_l, \dots, S_l)$. It is clear that $f(S_1, \dots, S_l)$ is well defined and depends neither on the order of the sets S_i nor on the number of repetitions of those sets. Therefore, we will also write $f(\mathcal{S})$ for a family of non-empty subsets $\mathcal{S} = \{S_1, \dots, S_l\}$ to denote $f(S_1, \dots, S_l)$.

If a k-block symmetric operation f satisfies $f(S_1, S_2, \dots, S_l) = f(S_2, S_2, S_3, \dots, S_l)$ whenever $S_2 \subseteq S_1$, we call it an *absorptive* k-block symmetric operation (or *k-ABS* operation, for short). The most typical example of such an operation is as follows.

Example 5. *(i) It is easy to check that, for any fixed linear order on B and any positive integers k, n, the operation*

$$f(x_{11}, \dots, x_{1k}, \dots, x_{n1}, \dots, x_{nk}) = \\ = \min(\max(x_{11}, \dots, x_{1k}), \dots, \max(x_{n1}, \dots, x_{nk})).$$

is a k-ABS operation.

(ii) More generally, if $(B, \sqcap, \sqcup)$ is a lattice (i.e., $\sqcap, \sqcup$ are lattice operations in infix notation) then it is easy to check that, for any k, n, the operation

$$f(x_{11}, \dots, x_{1k}, \dots, x_{n1}, \dots, x_{nk}) = \\ = (x_{11} \sqcup \dots \sqcup x_{1k}) \sqcap \dots \sqcap (x_{n1} \sqcup \dots \sqcup x_{nk}).$$

is a k-ABS operation.

For an r-ary relation R on B and a number $1 \leq m \leq r$, let $\mathrm{pr}_m(R) = \{a_m \mid (a_1, \dots, a_m, \dots, a_r) \in R\}$.

Let $\mathbf{B}$ be a τ-structure. We construct a structure $\mathbf{C}(\mathbf{B})$ as follows: the elements of $\mathbf{C}(\mathbf{B})$ are the families of non-empty subsets of B; for each r-ary relation $R^{\mathbf{B}}$, we have $(\mathcal{S}^1, \mathcal{S}^2, \dots, \mathcal{S}^r) \in R^{\mathbf{C}(\mathbf{B})}$ if, for all $j, m = 1, \dots, r$, we have

1. $\mathrm{pr}_m(R^{\mathbf{B}}) \in \mathcal{S}^m$, and
2. $\mathrm{pr}_m(R^{\mathbf{B}} \cap (B^{j-1} \times S \times B^{r-j})) \in \mathcal{S}^m$ for each $S \in \mathcal{S}^j$.

Note that the empty family belongs to the universe of $\mathbf{C}(\mathbf{B})$, but it never appears in any tuple in a relation in this structure.

Theorem 6. *Let* $\mathbf{B}$ *be a structure. The following conditions are equivalent:*

1. $\mathbf{B}$ *has caterpillar duality;*
2. co-CSP($\mathbf{B}$) *is definable by a linear monadic Datalog program with at most one EDB per rule;*
3. $\mathbf{C}(\mathbf{B})$ *admits a homomorphism to* $\mathbf{B}$*;*
4. *for every $k, n \geq 1$,* $\mathbf{B}$ *has a kn-ary k-ABS polymorphism.*

If $\mathbf{B}$ *is a core then the above conditions are equivalent to the following:*

5. $\mathbf{B}$ *is the core of a structure with lattice polymorphisms.*

The proof of Theorem 6 will follow from subsequent lemmas in this section.

For the sake of brevity, we will say that a linear monadic Datalog program with at most one EDB per rule is a *caterpillar program*. For a given structure $\mathbf{B}$ and a given fragment of Datalog, there is a standard way of constructing the *canonical program* for $\mathbf{B}$ in the given fragment of Datalog (see, e.g., [4, 11]). The canonical caterpillar program for a structure $\mathbf{B}$ is constructed as follows: let $S_0, S_1, \dots, S_p$ be an enumeration of unary relations on B (i.e., subsets of B) that can be expressed by a first-order $\exists\wedge$-formula over $\mathbf{B}$. Assume that S_0 is the empty relation. For each S_i, introduce a unary IDB I_i. Then the canonical caterpillar program for $\mathbf{B}$ involves the IDBs $I_0, \dots, I_p$ and EDBs

$R_1, \ldots, R_n$, and contains all the rules (whose body contains at most one EDB and at most one IDB) with the following property: if every I_i in the rule is replaced by S_i and every R_s by $R_s^{\mathbf{B}}$, then every assignment of elements of B to the variables that satisfies the conjunction of atomic formulas in the body must also satisfy the atomic formula in the head. Finally, include the goal predicate G along with the rule $G :- I_0(x)$.

Note that $\mathbf{B}$ is not accepted by the canonical program (for itself). Indeed, by construction, a derivation of G on $\mathbf{B}$ could be translated into a chain of valid implications which starts from an atomic formula and finishes with the empty (i.e. false) predicate, which is impossible. This implies the following fact.

Fact 7. *If the canonical caterpillar program for* $\mathbf{B}$ *accepts a structure* $\mathbf{A}$ *then* $\mathbf{A} \not\to \mathbf{B}$.

We now relate caterpillar duality with caterpillar Datalog programs.

Lemma 8. *For any structures* $\mathbf{A}$ *and* $\mathbf{B}$*, if there exists a caterpillar* $\mathbf{C}$ *such that* $\mathbf{C} \to \mathbf{A}$ *and* $\mathbf{C} \not\to \mathbf{B}$ *then the canonical program for* $\mathbf{B}$ *accepts* $\mathbf{A}$.

Proof. Since the class of structures accepted by any Datalog program is closed under homomorphism, it suffices to show that $\mathbf{C}$ is accepted by the program. Clearly, this is the case when $Inc(\mathbf{C})$ has no non-leaves (i.e., the total number of tuples in relations in $\mathbf{C}$ is 1) because we must have $R^{\mathbf{B}} = \emptyset$, where $R^{\mathbf{C}}$ is the only non-empty relation in $\mathbf{C}$, and then the program has the rule $I_0(x_1) :- R(x_1, \ldots, x_r)$.

Now let $\mathbf{D}$ be an arbitrary caterpillar with a total of at least two tuples in its relations, and let a be a non-leaf in $Inc(\mathbf{D})$ which has at most one other non-leaf at distance two from it. Informally, this means that a is at one of the two extremes of the body of the caterpillar $\mathbf{D}$. We claim that the canonical program can derive (on $\mathbf{D}$) the fact $I_j(a)$ where I_j is the IDB corresponding to a set $S_j = \{h(a) \mid h : \mathbf{D} \to \mathbf{B}\}$. This claim implies the lemma, since I_0 corresponds to the empty set.

We prove the claim by induction on the total number of tuples in relations in $\mathbf{D}$. If this number is two then $Inc(\mathbf{D})$ has two blocks $Z_1 = (R_i, \bar{a}_1)$ and $Z_2 = (R_t, \bar{a}_1)$ where i and t may coincide and tuples $\bar{a}_1$ and $\bar{a}_2$ share exactly one element – the non-leaf a. Assume without loss of generality that a appears in the first coordinate in $\bar{a}_1$ and in the second coordinate in $\bar{a}_2$, and let $S_1 = \mathrm{pr}_1(R_i^{\mathbf{B}})$, $S_2 = \mathrm{pr}_2(R_t^{\mathbf{B}})$, and $S_3 = S_1 \cap S_2$. Then the canonical program has the rules $I_1(x_1) :- R_i(x_1, x_2, \ldots, x_{r_i})$ and $I_3(x_2) :- R_t(x_1, x_2, \ldots, x_{r_t}), I_1(x_2)$. It is clear that $S_3 = \{h(a) \mid h : \mathbf{D} \to \mathbf{B}\}$, and that $I_3(a)$ is derived on $\mathbf{D}$ if and only if $\mathbf{D} \to \mathbf{B}$. This proves the base of induction.

Assume now that the claim holds for all caterpillars in which the total number of tuples is less than that in $\mathbf{D}$. Fix a non-leaf a satisfying the assumptions of the claim. Then we can choose a pendant block $Z_1 = (R_i, \bar{a}_1)$ such that a is the only non-leaf in $\bar{a}$. Without loss of generality assume that a appears in the first component in $\bar{a}_1$. Let $\mathbf{D}'$ be the caterpillar obtained from $\mathbf{D}$ by removing $\bar{a}_1$ from $R_i^{\mathbf{D}}$ (and all leaves in $\bar{a}_1$ from D). We consider two cases depending on whether or not a becomes a leaf in $\mathbf{D}'$.

If a is still a non-leaf in $\mathbf{D}'$ (that is, Z_1 was not the only pendant block incident to a in $Inc(\mathbf{D})$) then, by the induction hypothesis, the canonical program can derive (on $\mathbf{D}'$) the fact $I_p(a)$ where I_p is the IDB corresponding to a set $S_p = \{h(a) \mid h : \mathbf{D}' \to \mathbf{B}\}$. Let $S_j = S_p \cap \mathrm{pr}_1(R_i^{\mathbf{B}})$. It is clear that $S_j = \{h(a) \mid h : \mathbf{D} \to \mathbf{B}\}$, while the canonical program derives (on $\mathbf{D}$) $I_j(a)$ via the rule $I_j(x_1) :- R_i(x_1, x_2, \ldots, x_{r_i}), I_p(x_1)$.

Consider the case when a becomes a leaf in $\mathbf{D}'$. This means that Z_1 was the only pendant block incident to a in $Inc(\mathbf{D})$. Let $Z_2 = (R_t, \bar{a}_2)$ be the (only) other block incident to a (both in $Inc(\mathbf{D})$ and $Inc(\mathbf{D}')$). Note that this block was non-pendant in $\mathbf{D}$, but becomes pendant in $\mathbf{D}'$. Assume without loss of generality that a appears in the first coordinate in $\bar{a}_2$, while the single non-leaf in $\bar{a}_2$ (in $\mathbf{D}'$), say a', appears in the second component in $\bar{a}_2$. By the induction hypothesis, the canonical program can derive (on $\mathbf{D}'$) the fact $I_p(a')$ where I_p is the IDB corresponding to the set $S_p = \{h(a') \mid h : \mathbf{D}' \to \mathbf{B}\}$. Let $S_q = \mathrm{pr}_1(R_t^{\mathbf{B}} \cap (B \times S_p \times B^{r_t-2}))$ and $S_j = S_q \cap \mathrm{pr}_1(R_i^{\mathbf{B}})$. It is easy to check that $S_j = \{h(a) \mid h : \mathbf{D} \to \mathbf{B}\}$, while the program derives (on $\mathbf{D}$) $I_j(a)$ via the rules $I_q(x_1) :- R_t(x_1, x_2, \ldots, x_{r_t}), I_p(x_2)$, and $I_j(x_1) :- R_i(x_1, x_2, \ldots, x_{r_i}), I_q(x_1)$. ∎

Lemma 9. *A structure* $\mathbf{B}$ *has caterpillar duality if and only if* co-CSP($\mathbf{B}$) *can be defined by a caterpillar program.*

Proof. Suppose that co-CSP($\mathbf{B}$) is defined by a caterpillar program. This means that a structure $\mathbf{A}$ satisfies $\mathbf{A} \not\to \mathbf{B}$ if and only if $\mathbf{A}$ is accepted by the program.

If $\mathbf{A} \not\to \mathbf{B}$ then the program derives the goal predicate on $\mathbf{A}$. Reading the derivation from the end to the beginning we obtain

$$\begin{aligned}
G :- \quad & I_0(a_0) \\
I_0(a_0) :- \quad & R_{i_1}(\ldots, a_0, \ldots, a_i, \ldots), I_{j_1}(a_i) \\
I_{j_1}(a_i) :- \quad & R_{i_2}(\ldots, a_i, \ldots, a_j, \ldots), I_{j_2}(a_j) \\
& \vdots \\
I_{j_{(l-1)}}(a_p) :- \quad & R_{i_l}(\ldots, a_p, \ldots, a_q, \ldots), I_{j_l}(a_q) \\
I_{j_l}(a_q) :- \quad & R_{i_{l+1}}(\ldots, a_q, \ldots).
\end{aligned}$$

Consider a substructure $\mathbf{A}'$ of $\mathbf{A}$ such that, for any $R \in \tau$ and any $\bar{a}$, we have $\bar{a} \in R^{\mathbf{A}'}$ if and only if $R(\bar{a})$ appears in the above derivation. Now modify the structure $\mathbf{A}'$ by giving new names to the occurrences of each element in such a way that in the obtained structure we have the following:

- there is no repetition of elements in any tuple in any relation, and
- if two tuples (possibly in different relations) share an element then this element appears in the heads of all rules between the rules corresponding to the two tuples.

Call the obtained structure $\mathbf{C}$. It is clear that $\mathbf{C}$ is a caterpillar. We also have that $\mathbf{C}$ homomorphically maps to $\mathbf{A}'$ (via reverse renaming) and hence to $\mathbf{A}$, but not to $\mathbf{B}$ because the program still derives the goal predicate on $\mathbf{C}$. Hence, $\mathbf{B}$ has caterpillar duality.

Conversely assume that $\mathbf{B}$ has caterpillar duality, i.e. for any structure $\mathbf{A}$ we have $\mathbf{A} \to \mathbf{B}$ if and only if all caterpillars that homomorphically map to $\mathbf{A}$ also map to $\mathbf{B}$. We claim that the canonical caterpillar program for $\mathbf{B}$ defines co-CSP($\mathbf{B}$). By Fact 7, the canonical program never accepts a structure that homomorphically maps to $\mathbf{B}$. Now let $\mathbf{A} \in$ co-CSP($\mathbf{B}$) be arbitrary. By assumption, there is a caterpillar $\mathbf{C}$ such that $\mathbf{C} \to \mathbf{A}$ and $\mathbf{C} \not\to \mathbf{B}$. It follows from Lemma 8 that the program accepts $\mathbf{A}$. ∎

Corollary 10. *If* co-CSP($\mathbf{B}$) *is definable by some caterpillar program then it is definable by the canonical one.*

Lemma 11. *A structure* $\mathbf{A}$ *is not accepted by the canonical caterpillar program for* $\mathbf{B}$ *if and only if* $\mathbf{A} \to \mathbf{C}(\mathbf{B})$.

Proof. Assume first that $\mathbf{A} \to \mathbf{C}(\mathbf{B})$ and show that $\mathbf{A}$ is not accepted by the canonical program. Since the class of structures accepted by any Datalog program is closed under homomorphism, it suffices to show that $\mathbf{C}(\mathbf{B})$ is not accepted by the canonical program. We will show by induction on the length of derivation that whenever the fact $I_j(\mathcal{S})$ is derived by the program, we have $S_j \in \mathcal{S}$ where S_j is the subset of B corresponding to I_j.

Assume first that $I_j(\mathcal{S})$ is derived by an introductory rule (i.e., one whose body contains no IDB and $R \in \tau$ is the EDB in the rule), that is, we have $I_j(\mathcal{S}) : -R(\ldots, \mathcal{S}, \ldots)$ where $\mathcal{S}$ appears in the m-th component in the tuple on the right. Note that this tuple belongs to $R^{\mathbf{C}(\mathbf{B})}$. Then, by the definition of the canonical program I_j corresponds to the subset $\mathrm{pr}_m(R^{\mathbf{B}})$ of B, which must be contained in $\mathcal{S}$ by the definition of $\mathbf{C}(\mathbf{B})$.

Assume now that $I_j(\mathcal{S})$ is derived by a rule $I_j(\mathcal{S}) : -R(\ldots, \mathcal{S}, \ldots, \mathcal{T}, \ldots), I_l(\mathcal{T})$. Assume without loss of generality that S_j is the smallest set such that I_j can be in the head of this rule. By the induction hypothesis, we have $S_l \in \mathcal{T}$. Assume that $\mathcal{S}$ appears in the m-th component and $\mathcal{T}$ in the k-th component in the EDB of the rule. Again, by the definition of the program, we have $S_j = \mathrm{pr}_m(R^{\mathbf{B}} \cap (B^{k-1} \times S_l \times B^{r-k}))$ where r is the arity of R. Then we must have $S_j \in \mathcal{S}$ by the definition of $\mathbf{C}(\mathbf{B})$.

Assume now that $\mathbf{C}(\mathbf{B})$ is accepted by the canonical program. Then the program can derive $I_0(\mathcal{S})$ for some $\mathcal{S}$. Then, as we just proved, the empty set S_0 belongs to $\mathcal{S}$ which is impossible by the definition of $\mathbf{C}(\mathbf{B})$.

Conversely, assume $\mathbf{A}$ is not accepted by the program. Hence the program stabilizes without deriving the goal predicate. Recall that every IDB I_j in the canonical program corresponds to a subset S_j of B. For every element $a \in A$, consider the family $\mathcal{S}_a = \{S_j \mid I_j(a) \text{ is derived }\}$ of subsets of B. It is easy to see that the family is non-empty for any a that appears in a tuple in a relation in $\mathbf{A}$. Moreover, since the goal predicate is not derived, $I_0(a)$ is not derived either, and so each subset in a non-empty $\mathcal{S}_a$ is non-empty. It is straightforward to check that the mapping $h : A \to C(B)$ given by $h(a) = \mathcal{S}_a$ is a homomorphism from $\mathbf{A}$ to $\mathbf{C}(\mathbf{B})$. ∎

Lemma 12. *For any structure* $\mathbf{B}$, co-CSP($\mathbf{B}$) *is definable by a caterpillar program if and only if* $\mathbf{C}(\mathbf{B})$ *admits a homomorphism to* $\mathbf{B}$.

Proof. Suppose that co-CSP($\mathbf{B}$) is definable by a caterpillar program. Then it is definable by the canonical one, by Corollary 10. By Lemma 11, $\mathbf{C}(\mathbf{B})$ is not accepted by the canonical program, and hence $\mathbf{C}(\mathbf{B}) \to \mathbf{B}$.

Conversely, suppose that $\mathbf{C}(\mathbf{B}) \to \mathbf{B}$. By Fact 7, each structure from CSP($\mathbf{B}$) is not accepted by the canonical caterpillar program for $\mathbf{B}$. On the other hand, if a structure $\mathbf{A}$ is not accepted by the program then we have $\mathbf{A} \to \mathbf{C}(\mathbf{B})$ by Lemma 11, and so $\mathbf{A} \to \mathbf{B}$. Hence, co-CSP($\mathbf{B}$) is definable by the canonical caterpillar program for $\mathbf{B}$. ∎

Lemma 13. *The relations in* $\mathbf{C}(\mathbf{B})$ *are invariant under lattice operations.*

Proof. We show that every relation in $\mathbf{C}(\mathbf{B})$ is invariant under set intersection and set union operations, which are easily seen to be lattice operations. Let $R^{\mathbf{C}(\mathbf{B})}$ be an r-ary relation in $\mathbf{C}(\mathbf{B})$ and take arbitrary tuples $(\mathcal{S}^1, \ldots, \mathcal{S}^r), (\mathcal{T}^1, \ldots, \mathcal{T}^r) \in R^{\mathbf{C}(\mathbf{B})}$.

It follows directly from the definition of $\mathbf{C}(\mathbf{B})$ that for all $j, m = 1, \ldots, r$ we have

1. $\mathrm{pr}_m(R^{\mathbf{B}}) \in \mathcal{S}^m \cap \mathcal{T}^m$, and
2. $\mathrm{pr}_m(R^{\mathbf{B}} \cap (B^{j-1} \times S \times B^{r-j})) \in \mathcal{S}^m \cap \mathcal{T}^m$ for each $S \in \mathcal{S}^j \cap \mathcal{T}^j$.

It follows that $(\mathcal{S}^1 \cap \mathcal{T}^1, \ldots, \mathcal{S}^r \cap \mathcal{T}^r) \in R^{\mathbf{C}(\mathbf{B})}$. The fact that $(\mathcal{S}^1 \cup \mathcal{T}^1, \ldots, \mathcal{S}^r \cup \mathcal{T}^r) \in R^{\mathbf{C}(\mathbf{B})}$ can be verified equally easily. ∎

Lemma 14. *A structure* $\mathbf{B}$ *has a* kn*-ary* k*-ABS polymorphism for all* k, n *if and only if* $\mathbf{C}(\mathbf{B}) \to \mathbf{B}$.

Proof. Let $h : \mathbf{C}(\mathbf{B}) \to \mathbf{B}$ be a homomorphism. By Lemma 13, the structure $\mathbf{C}(\mathbf{B})$ has polymorphisms which are set-theoretic union and intersection operations. Since composition of polymorphisms is again a polymorphism, it follows that $\mathbf{C}(\mathbf{B})$ also has the k-ABS polymorphisms

$$f_{k,n}(X_1, \ldots, X_n) = (\bigcap X_1) \cup (\bigcap X_2) \cup \ldots \cup (\bigcap X_n),$$

where $X_i = \{x_{i1}, \ldots, x_{ik}\}$. By Lemma 11, there exists a homomorphism $g : \mathbf{B} \to \mathbf{C}(\mathbf{B})$. It is straightforward to check that the operations $h(f_{k,n}(g(x_1), \ldots, g(x_{kn})))$ are k-ABS polymorphisms of $\mathbf{B}$.

Conversely, let f be a kn-ary k-ABS polymorphism of $\mathbf{B}$ with $k = \rho \cdot |B|$ and $n = \rho(2^{|B|} - 1)$, where ρ is the maximum of the arities of the relations in $\mathbf{B}$. Define a map $h : C(B) \to B$ by the rule $h(\mathbb{S}) = f(\mathbb{S})$ for non-empty $\mathbb{S}$ and set $h(\emptyset)$ arbitrarily. Let us now show that h is a homomorphism. By the properties of f, h is clearly a well-defined function. Take an arbitrary (say, r-ary) relation $R \in \tau$ and fix $(\mathbb{S}^1, \ldots, \mathbb{S}^r) \in R^{\mathbf{C}(\mathbf{B})}$. We need to show that $(h(\mathbb{S}^1), \ldots, h(\mathbb{S}^r)) \in R^{\mathbf{B}}$.

Let $\hat{\mathbb{S}}^i = \{X \cap \mathrm{pr}_i\, R^{\mathbf{B}} \mid X \in \mathbb{S}^i\}$. It immediately follows from the definition of the structure $\mathbf{C}(\mathbf{B})$ that we have $\hat{\mathbb{S}}^i \subseteq \mathbb{S}^i$ for all $1 \leq i \leq r$, and also that $(\hat{\mathbb{S}}^1, \ldots, \hat{\mathbb{S}}^r) \in R^{\mathbf{C}(\mathbf{B})}$. Since f is absorptive, we have $f(\hat{\mathbb{S}}^i) = f(\mathbb{S}^i)$. Therefore, we can without loss of generality assume that each $\mathbb{S}^i$ contains only subsets of $\mathrm{pr}_i\, R^{\mathbf{B}}$.

For a set $S \in \mathbb{S}^i$, construct a $(k \times r)$-matrix M^i_S whose entries are elements from B and such that

1. each row of M^i_S is an element of $R^{\mathbf{B}}$, and
2. for any $1 \leq m \leq r$, the set of entries in the m-th column is $\mathrm{pr}_m(R^{\mathbf{B}} \cap (B^{i-1} \times S \times B^{r-i}))$.

Let us show that this is possible. Recall that $S \subseteq \mathrm{pr}_i(R^{\mathbf{B}})$. Divide the matrix into r submatrices of $|B|$ consecutive rows. For $1 \leq m \leq r$, the rows of the m-th submatrix are tuples (from $R^{\mathbf{B}}$) whose i-th coordinate belongs to S and whose m-th coordinates cover all of $\mathrm{pr}_m(R^{\mathbf{B}} \cap (B^{i-1} \times S \times B^{r-i}))$.

Now construct a matrix M with kn rows and r columns, as follows. It is divided into n layers of k consecutive rows, each layer is a matrix M^i_S for some $1 \leq i \leq r$ and some $S \in \mathbb{S}^i$, and each matrix of this form appears as a layer. By the choice of n, this is possible.

It remains to notice that the operation f applied to the i-th column of M gives the value $f(\mathbb{S}^i)$, and, since f is a polymorphism of $\mathbf{B}$ and every row of M is in $R^{\mathbf{B}}$, we have $(f(\mathbb{S}^1), \ldots, f(\mathbb{S}^r)) \in R^{\mathbf{B}}$, as required. Thus $(h(\mathbb{S}^1), h(\mathbb{S}^2), \ldots, h(\mathbb{S}^r)) \in R^{\mathbf{B}}$. We conclude that $h : \mathbf{C}(\mathbf{B}) \to \mathbf{B}$. ∎

Remark 15. If a structure $\mathbf{B}$ has kn-ary k-ABS polymorhism for $k = \rho \cdot |B|$ and $n = \rho(2^{|B|} - 1)$, where ρ is the maximum of the arities of the relations in $\mathbf{B}$, then, for any k, $\mathbf{B}$ has k-ABS polymorphisms of all arities divisible by k.

Proof. (of Theorem 6).
(1) ⇔ (2) follows from Lemma 9.
(2) ⇔ (3) follows from Lemma 12.
(3) ⇔ (4) follows from Lemma 14.

Assume now that that $\mathbf{B}$ is a core. If condition (3) holds then we have homomorphisms $\mathbf{B} \to \mathbf{C}(\mathbf{B})$ (by Lemma 11) and $\mathbf{C}(\mathbf{B}) \to \mathbf{B}$ implying, since $\mathbf{B}$ is a core, that $\mathbf{B}$ is the core of $\mathbf{C}(\mathbf{B})$. The structure $\mathbf{C}(\mathbf{B})$ has lattice polymorphisms by Lemma 13, so (3) implies (5). On the other hand, (5) implies (4) because any structure with lattice polymorphisms has the required k-ABS polymorphisms (see Example 5) which can be transferred to $\mathbf{B}$ as in the proof of Lemma 14. ∎

Theorem 16. *The problem of checking whether a given structure has caterpillar duality is decidable, but* **NP***-hard.*

Proof. Decidability of the problem immediately follows from condition (3) of Theorem 6. We now prove, by reduction from 3-SAT, that the problem is **NP**-hard even when restricted to digraphs. Let $\mathbf{T}_n$ be the transitive tournament on n vertices. It is shown in the proof of Theorem 6.1 of [22] that, given an instance $\mathcal{I}$ of 3-SAT, one can construct in polynomial time a digraph $\mathbf{H}$ such that (i) $\mathbf{T}_n$ is the core of $\mathbf{H}$ if and only if $\mathcal{I}$ is satisfiable, and (ii) either $\mathbf{T}_n$ is the core of $\mathbf{H}$ or else $\mathbf{H}$ does not have tree duality. It remains to say that the directed path on $n+1$ vertices forms an obstruction set for $\mathbf{T}_n$ [15], so $\mathbf{T}_n$ has caterpillar duality. ∎

3.2 Caterpillar duality vs. path duality

One can define *τ-paths* as τ-caterpillars with at most two pendant blocks. Say, if τ is the signature of digraphs then τ-paths are oriented paths (i.e., digraphs obtained from paths by orienting each edge in some way). One can also define path dualities in a natural way, and obtain a characterisation similar to conditions (1)-(3) of Theorem 6. However, the fragment of Datalog arising from this connection is not very natural and it does not seem to have a natural equivalent algebraic condition such as conditions (4) and (5) of Theorem 6. Since paths and caterpillars are very close structurally, it is natural to ask whether caterpillar duality and path duality are equivalent properties.

Proposition 17. *There exist digraphs that have caterpillar duality, but not path duality.*

Proof. Let $\mathbf{C}$ be an arbitrary core digraph that is an oriented caterpillar, but not a path. By results of [25], there exists a structure $\mathbf{B}$ such that $\mathbf{C}$ is the obstruction set for $\mathbf{B}$, that is, for any digraph $\mathbf{G}$, we have either $\mathbf{C} \to \mathbf{G}$ or else $\mathbf{G} \to \mathbf{B}$. Clearly, $\mathbf{B}$ has caterpillar duality. We claim that it does not have path duality. Suppose, for a contradiction, that it does. Then, since $\mathbf{C} \not\to \mathbf{B}$, there is an oriented path $\mathbf{P}$ such that $\mathbf{P} \to \mathbf{C}$ and $\mathbf{P} \not\to \mathbf{B}$. The latter property implies that $\mathbf{C} \to \mathbf{P}$, in which case $\mathbf{C}$ must be the core of $\mathbf{P}$, which is impossible. ∎

4 Applications to list H-colouring

In the *list* $\mathbf{H}$*-colouring* problem for a fixed (di)graph $\mathbf{H}$, one is given a (di)graph $\mathbf{G}$, and, for each vertex v of $\mathbf{G}$, a list L_v of possible target vertices in $\mathbf{H}$. The question is whether there is a homomorphism $h : \mathbf{G} \to \mathbf{H}$ such that $h(v) \in L_v$ for each vertex v of $\mathbf{G}$. It is well known (and easy to see) that this problem is exactly CSP($\mathbf{H}_u$) where $\mathbf{H}_u$ is the structure obtained by expanding the (di)graph $\mathbf{H}$ with unary relations U where U runs through all non-empty subsets of H. It is easy to see that the polymorphisms of $\mathbf{H}_u$ are exactly the conservative polymorphisms of $\mathbf{H}$.

Recall that a (di)graph is called *reflexive* it contains all loops, and *irreflexive* if it contains no loop.

4.1 List H-colouring for undirected graphs

All graphs in this subsection are undirected. It was shown in [10] that, for a reflexive graph $\mathbf{H}$, the list $\mathbf{H}$-colouring problem is solvable in polynomial time if $\mathbf{H}$ is an interval graph and **NP**-complete otherwise. Recall that a (reflexive) graph is called an *interval graph* if its vertices can be represented by intervals (on the real line) in such a way that two vertices are adjacent if and only if the corresponding intervals intersect.

Assume now that $\mathbf{H} = (H, E)$ is a reflexive interval graph. By modifying the proof in [10], it is possible to show directly that the structure $\mathbf{H}_u$ (as above) has caterpillar duality. We give a short proof of this fact using Theorem 6.

Theorem 18. *For every k and n, the graph $\mathbf{H}$ has a conservative k-ABS polymorphism of arity kn.*

Proof. Fix an interval representation of $\mathbf{H}$. We can without loss of generality assume that the endpoints of the intervals representing vertices of $\mathbf{H}$ are pairwise distinct [10]. Given an interval $u \in V$, we denote by $l(u)$ and $r(u)$ the left and right endpoints of u, respectively.

Let $k, n \geq 1$ be arbitrary. Define two functions on H, Min_l and Max_r, as follows:

$$\mathrm{Min}_l(u_1, \ldots, u_n) = u_i \text{ such that } l(u_i) = \min_j l(u_j),$$

$$\mathrm{Max}_r(u_1, \ldots, u_k) = u_i \text{ such that } r(u_i) = \max_j r(u_j).$$

Note that the functions are well defined because the intervals in H cannot have the same endpoints.

Let $S_1, S_2, \ldots, S_n$ be sets of vertices of $\mathbf{H}$ (i.e., sets of intervals) with at most k elements each. We obtain from them a new sequence of sets, as follows: for each $j = 1, \ldots, n$ such that S_j properly contains some set S_i, choose S_i so that S_i is minimal with this property and replace S_j by S_i. Break ties arbitrarily. Call the obtained sets $S'_1, S'_2, \ldots, S'_n$.

Define an operation $h : H^{nk} \to H$ as follows:

$$h(x_{11}, \ldots, x_{1k}, \ldots, x_{n1}, \ldots, x_{nk}) = \\ = \mathrm{Min}_l(\mathrm{Max}_r(S'_1), \ldots, \mathrm{Max}_r(S'_n))$$

where, for $1 \leq i \leq n$, $S_i = \{x_{i1}, \ldots, x_{ik}\}$, and S'_i is obtained as described above. Note that the set $\{S'_1, \ldots, S'_n\}$ depends only on $\{S_1, \ldots, S_n\}$. This and the (obvious) fact that the operations Max_r and Min_l are totally symmetric implies that the operation h is well defined and also that it is a k-block symmetric operation.

Let us show that h is absorptive. We now can use notation $h(S_1, S_2, \ldots, S_n)$ since h is k-block symmetric. Assume that $S_2 \subset S_1$. Then $S'_1 = S_i$ for some $i > 1$. Note that, by construction, we have $S'_i = S_i$. Assume without loss of generality that $i = 3$. Then we have

$$h(S_1, S_2, \ldots, S_n) = \\ = \mathrm{Min}_l(\mathrm{Max}_r(S'_3), \mathrm{Max}_r(S'_2), \ldots, \mathrm{Max}_r(S'_n)),$$

and

$$h(S_2, S_2, S_3, \ldots, S_n) = \\ = \mathrm{Min}_l(\mathrm{Max}_r(S'_2), \mathrm{Max}_r(S'_2), \ldots, \mathrm{Max}_r(S'_n)).$$

The right-hand sides of the above equations are the same (since Min_l is totally symmetric), so the left-hand sides are equal as well, as required.

It is obvious that h is conservative. It remains to show that it is a polymorphism of $\mathbf{H}$. For all $1 \leq i \leq n$ and $1 \leq l \leq k$, let s_{il} and t_{il} be intervals in V that intersect (i.e., adjacent in $\mathbf{H}$). Also let $S_i = \{s_{i1}, \ldots, s_{ik}\}$ and $T_i = \{t_{i1}, \ldots, t_{ik}\}$ for $1 \leq i \leq n$. We need to show that the intervals $s = h(S_1, \ldots, S_n)$ and $t = h(T_1, \ldots, T_n)$ also intersect.

We have $s = \mathrm{Min}_l(\mathrm{Max}_r(S'_1), \ldots, \mathrm{Max}_r(S'_n))$ and $t = \mathrm{Min}_l(\mathrm{Max}_r(T'_1), \ldots, \mathrm{Max}_r(T'_n))$. Hence, $s = \mathrm{Max}_r(S'_i)$ for some i and $t = \mathrm{Max}_r(T'_j)$ for some j. It is easy to see that i and j can be chosen so that $S_i = S'_i$ and $T_j = T'_j$. Since $S_i = S'_i$, and $T'_i \subseteq T_i$ we know that every interval in T'_i intersects some interval in S'_i. Similarly, every interval in S'_j intersects some interval in T'_j.

Suppose, for a contradiction, that $s \cap t = \emptyset$. Assume first that t precedes s, i.e. $r(t) < l(s)$. Since

$s = \text{Min}_l(\text{Max}_r(S'_1), \ldots, \text{Max}_r(S'_n))$, we have $l(s) \leq l(\text{Max}_r(S'_j))$. Since $\text{Max}_l(S'_j) \in S'_j$, it intersects some interval $t_j \in T'_j$. In particular, we have $l(\text{Max}_r(S_j)) < r(t_j)$. By combining the three above inequalities, we obtain $r(t) < l(s) \leq l(\text{Max}_r(S_j)) < r(t_j)$, which contradicts the fact $t = \text{Max}_r(T'_j)$. If $r(s) < l(t)$ then the argument is symmetric.

Thus h is a polymorphism and the theorem is proved. ■

Corollary 19. *For a reflexive graph* $\mathbf{H}$*, either* $\mathbf{H}_u$ *has caterpillar duality or* CSP($\mathbf{H}_u$) *is* **NP***-complete.*

Remark 20. If $\mathbf{H}$ is the reflexive claw (i.e., the complete bipartite graph $\mathbf{K}_{1,3}$ with loops) then it is easy to check that $\mathbf{H}_u$ does not have lattice polymorphisms, even though it is the core of a structure with such polymorphisms (by Theorem 6).

Remark 21. By using results from [22], one can show that, for a reflexive graph $\mathbf{H}$, $\mathbf{H}_u$ does not have finite duality unless the graph $\mathbf{H}$ is complete.

4.2 List H-colouring for directed graphs

All graphs in this subsection are directed and, unless specified otherwise, irreflexive (i.e., loopless). It was shown in [17] that every oriented path has path duality (that is it has an obstruction set consisting of oriented paths). Since every oriented path is a caterpillar, every oriented path has caterpillar duality. We will show how to generalise this result to a much wider class of digraphs which, in particular, includes all oriented caterpillars. A directed acyclic graph (DAG) $\mathbf{G}$ is called *layered* (or *balanced*) if each vertex u of $\mathbf{G}$ can be assigned a positive integer $l(u)$, the *level* of u, so that every arc (u, v) of $\mathbf{G}$ satisfies $l(u) + 1 = l(v)$. Every layered DAG can be embedded into the plane in such a way that each vertex u lies on the horizontal line $y = l(u)$, and the arcs are straight lines. If, in addition, the embedding can be arranged in such a way that the arcs never cross each other then the graph is called a *planar layered* DAG. It is easy to see that every oriented caterpillar is a planar layered DAG.

Theorem 22. *If* $\mathbf{H}$ *is a planar layered DAG then* $\mathbf{H}_u$ *has caterpillar duality.*

Proof. Fix a planar layered embedding of $\mathbf{H}$ into the plane such that the vertices lie on horizontal lines (as described above) and consider the following total order on H: $u < v$ if and only if either (i) $l(u) < l(v)$ or else (ii) $l(u) = l(v)$ and u is to the left of v.

Let $\min$ and $\max$ be the lattice operations with respect to the above order. We now show that they are polymorphisms of $\mathbf{H}$. Let (a_1, b_1) and (a_2, b_2) be arcs in $\mathbf{H}$. We need to show that $(\min(a_1, a_2), \min(b_1, b_2))$ and $(\max(a_1, a_2), \max(b_1, b_2))$ are also arcs in $\mathbf{H}$. We consider the former case, the latter is similar. Assume without loss of generality that $\min(a_1, a_2) = a_1$. If $l(a_1) < l(a_2)$ then $l(b_1) < l(b_2)$ so $\min(b_1, b_2) = b_1$ and we are done. If $l(a_1) = l(a_2)$ then $l(b_1) = l(b_2)$ and we again have $\min(b_1, b_2) = b_1$ because otherwise the arcs (a_1, b_1) and (a_2, b_2) would cross. By Example 5 and Theorem 6, we are done. ■

Reflexive digraphs that admit polymorphisms $\min$ and $\max$ with respect to some linear ordering of the vertices were characterised in [14]. Obviously, if $\mathbf{H}$ is such a digraph then $\mathbf{H}_u$ has caterpillar duality.

Now let $\mathbf{H}$ be a reflexive digraph, and let $\mathbf{H}_c$ denote the structure obtained from $\mathbf{H}$ by adding all unary relations of the form $\{a\}$, $a \in H$. The problem CSP($\mathbf{H}_c$) is known in graph theory as *one-or-all* list $\mathbf{H}$-homomorphism problem, and it is equivalent to the so-called $\mathbf{H}$-retraction problem [10, 15]. It is easy to see that the polymorphisms of $\mathbf{H}_c$ are the idempotent polymorphisms of $\mathbf{H}$. Note that if τ is the signature of $\mathbf{H}_c$ then a τ-path is an oriented path each of whose ends may belong to a unary relation.

Theorem 23. *For any reflexive digraph* H*, the following are equivalent:*

1. $\mathbf{H}_c$ *has caterpillar duality;*
2. $\mathbf{H}_c$ *has tree duality and a majority polymorphism;*
3. $\mathbf{H}_c$ *has path duality.*

Proof. Clearly, (3) implies (1). Let us show that (1) implies (2). Caterpillar duality trivially implies tree duality. By Theorem 6, $\mathbf{H}_c$ has a 6-ary 2-ABS polymorphism f. It is easy to check that $f(x, y, z, x, y, z)$ is a majority polymorphism of $\mathbf{H}_c$. Finally, let us show that (2) implies (3). Let τ be the signature of $\mathbf{H}_c$ (i.e. one binary and $|H|$ unary relations). Let $\mathbf{A}$ a τ-structure such that $\mathbf{A} \not\to \mathbf{H}_c$. By the tree duality of $\mathbf{H}_c$, there exists a τ-tree $\mathbf{T}$ that is homomorphic to $\mathbf{A}$, but not to $\mathbf{H}_c$. Take $\mathbf{T}$ to be minimal, that is, any proper substructure of $\mathbf{T}$ is homomorphic to $\mathbf{H}_c$. Then, since $\mathbf{H}_c$ has a majority polymorphism, Theorem 1.17 of [26] implies that at most two elements of $\mathbf{T}$ are in unary relations in $\mathbf{T}$. This, the fact that $\mathbf{H}$ is reflexive, and the minimality of $\mathbf{T}$ imply that $\mathbf{T}$ is in fact a path. ■

References

[1] A. Atserias. On digraph coloring problems and treewidth duality. *European Journal of Combinatorics*, 2008. to appear.

[2] L. Barto, M. Kozik, and T. Niven. Graphs, polymorphisms and the complexity of homomorphism problems. In *STOC'08*, 2008. to appear.

[3] A. Bulatov. H-coloring dichotomy revisited. *Theoretical Computer Science*, 349(1):31–39, 2005.

[4] A. Bulatov, A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *Surveys on Complexity of Constraints*. 2008. (to appear).

[5] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8. Elsevier, 2006.

[6] V. Dalmau. Linear Datalog and bounded path duality for relational structures. *Logical Methods in Computer Science*, 1(1), 2005. (electronic).

[7] V. Dalmau and J. Pearson. Set functions and width 1 problems. In *CP'99*, volume 1713 of *LNCS*, pages 159–173, 1999.

[8] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1999.

[9] L. Egri, B. Larose, and P. Tesson. Symmetric Datalog and constraint satisfaction problems in Logspace. In *LICS'07*, pages 193–202, 2007.

[10] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Ser.B*, 72:236–250, 1998.

[11] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[12] G. Grätzer. *General Lattice Theory*. Birkhäuser, 2nd edition, 2002.

[13] M. Grohe. The structure of tractable constraint satisfaction problems. In *MFCS'06*, volume 4162 of *LNCS*, pages 58–72, 2006.

[14] A. Gupta, P. Hell, M. Karimi, and A. Rafiey. Minimum cost homomorphisms to reflexive digraphs. In *LATIN'08*, volume 4957 of *LNCS*, 2008. to appear.

[15] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[16] P. Hell, J. Nešetřil, and X. Zhu. Duality of graph homomorphisms. In *Combinatorics, Paul Erdös is Eighty (Vol.2)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 271–282. János Bolyai Math. Soc., 1996.

[17] P. Hell and X. Zhu. Homomorphisms to oriented paths. *Discrete Mathematics*, 132:107–114, 1994.

[18] P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. In *LICS'07*, pages 213–222, 2007.

[19] P. Kolaitis. On the expressive power of logics on finite models. In *Finite Model Theory and its Applications*, EATCS Series: Texts in Theoretical Computer Science. Springer, 2007.

[20] P. Kolaitis and M. Vardi. A logical approach to constraint satisfaction. In *Finite Model Theory and its Applications*, EATCS Series: Texts in Theoretical Computer Science. Springer, 2007.

[21] A. Krokhin, A. Bulatov, and P. Jeavons. The complexity of constraint satisfaction: an algebraic approach. In *Structural Theory of Automata, Semigroups, and Universal Algebra*, volume 207 of *NATO Science Series II: Math., Phys., Chem.*, pages 181–213. Springer Verlag, 2005.

[22] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007. (electronic).

[23] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. In *ICALP'07*, volume 4596 of *LNCS*, pages 267–278, 2007.

[24] C. Loten and C. Tardif. Majority functions on structures with finite duality. *European Journal of Combinatorics*, 2008. to appear.

[25] J. Nešetřil and C. Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *Journal of Combinatorial Theory, Ser.B*, 80:80–97, 2000.

[26] L. Zádori. Relational sets and categorical equivalence for algebras. *International Journal of Algebra and Computation*, 7(5):561–576, 1997.