CSP duality and trees of bounded pathwidth

Catarina Carvalho[a], Víctor Dalmau[b], Andrei Krokhin[*,c]

[a] *Centre for Algebra, University of Lisbon, 1649-003 Lisbon, Portugal*
[b] *Department of Technology, University Pompeu Fabra, Barcelona, 08018 Spain*
[c] *School of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE, UK. Tel: +44 191 3341743, fax: +44 191 3341701*

Abstract

We study non-uniform constraint satisfaction problems definable in monadic Datalog stratified by the use of non-linearity. We show how such problems can be described in terms of homomorphism dualities involving trees of bounded pathwidth and in algebraic terms. For this, we introduce a new parameter for trees that closely approximates pathwidth and can be characterised via a hypergraph searching game.

Key words: constraint satisfaction problem, homomorphism duality, Datalog, polymorphisms, bounded pathwidth

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. It is well known (see, e.g., [11, 18]) that the CSP can be cast as the following fundamental problem: given two finite relational structures **A** and **B**, is there a homomorphism from **A** to **B**? The non-uniform CSP, when the structure **B** is fixed, and only **A** is part of the input, is one the most studied forms of the CSP. The obtained problem is denoted by CSP(**B**). Examples of such problems include k-Sat, Graph H-colouring, and Systems of Equations (e.g., linear equations).

The classification of relational structures **B** with respect to computational (i.e., membership in a given complexity class) and descriptive (i.e., definability in a given logic) complexity of CSP(**B**) has been a very active research direction in the last decade (see, e.g., [11, 18, 27]). A variety of mathematical approaches to study problems CSP(**B**) has been recently suggested. The most advanced approaches use logic, combinatorics, universal algebra, and their combinations (see [8, 9, 11, 25]). The most famous open problem about the computational

*Corresponding author
Email addresses: `ccarvalho@cii.fc.ul.pt` (Catarina Carvalho), `victor.dalmau@upf.edu` (Víctor Dalmau), `andrei.krokhin@durham.ac.uk` (Andrei Krokhin)

complexity of CSP(**B**) is the Dichotomy (aka Feder-Vardi) Conjecture [18] which states that each problem CSP(**B**) is either tractable (i.e., in **PTIME**) or **NP**-complete. The precise boundary between the two cases was conjectured in [7] in algebraic terms, and this refined conjecture was proved in several important special cases (e.g., [5, 6]) via algebraic tools. The logic programming language Datalog and its fragments are arguably some of the most important tools for studying CSP(**B**). In fact, all problems CSP(**B**) that are known to be tractable can be solved via algorithms based on definability in Datalog, or on the "few subpowers property" (see [9]), or on a combination of the two. A characterisation of structures **B** with (the complement of) CSP(**B**) definable in Datalog was also conjectured in [18], and then, in more algebraic terms in [30] (see also [9]); it became known as the Bounded Width (aka Larose-Zádori) Conjecture and was the most important open problem about the descriptive complexity of non-uniform CSP until it was very recently solved (in positive) in [4], also via algebraic tools.

Classification of structures **B** with respect to definability of co-CSP(**B**) in fragments of Datalog, such as linear [13] or symmetric [16] Datalog, now becomes one of the most important problems in descriptive complexity of non-uniform CSP, and the present paper contributes to this direction. Definability in linear Datalog is also important for the classification of computational complexity because, for every problem CSP(**B**) that is currently known to belong to **NL**, the complement of CSP(**B**) can be defined in linear Datalog (see [8, 10, 13, 14]). Moreover, in [27], it is suggested that the converse might also hold, and the algebraic approach to the CSP is linked with definability in linear and symmetric Datalog.

The definability of CSP(**B**) in Datalog and its fragments is very closely related with homomorphism dualities (see survey [8]). Roughly, a structure **B** has duality (of some type) if the non-existence of a homomorphism from a given structure **A** to **B** can always be explained by the existence of a simple enough obstruction structure (i.e., one that homomorphically maps to **A** but not to **B**). The types of dualities correspond to interpretations of the phrase "simple enough". For non-uniform CSP, definability in Datalog is equivalent to bounded treewidth duality, while definability in linear Datalog is equivalent to bounded pathwidth duality (see [8, 13]). On the algebraic side, some necessary [27] and sufficient [10, 14] conditions for having bounded pathwidth duality are known (see also [8]), but the gap between them is still huge.

Tree duality is the most well understood duality (see [8, 18]), it is equivalent to definability in monadic Datalog, and also has a nice algebraic characterisation. In this paper, we consider the form of duality that corresponds to trees of bounded pathwidth, extending results from [10]. We introduce a new structural parameter for trees that closely approximates pathwidth and can be nicely characterised via a hypergraph searching game in the spirit of [1, 3, 19, 20]. We use the new parameter to show that structures that have obstruction sets consisting of trees of bounded pathwidth can be equivalently described in terms of Datalog and in terms of algebra (which is why we prefer it to pathwidth in our context).

2. Preliminaries

2.1. Structures

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 τ-structure $\mathbf{C}$ is called a *substructure* of $\mathbf{A}$ if $C \subseteq A$ and $R^{\mathbf{C}} \subseteq R^{\mathbf{A}}$ for all $R \in \tau$. If, in addition, $R^{\mathbf{C}} = R^{\mathbf{A}} \cap C^{\rho(R)}$ for every $R \in \tau$ then $\mathbf{C}$ is called a substructure *induced by* C, and is also denoted by $\mathbf{A}[C]$.

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 $\mathbf{B}$ (or $\mathbf{A}$ is homomorphic 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. Also, if $a_1, \ldots, a_r \in A$ and $b_1, \ldots, b_r \in B$ we will write $\mathbf{A}, a_1, \ldots, a_r \to \mathbf{B}, b_1, \ldots, b_r$ to indicate that there is a homomorphism h from $\mathbf{A}$ to $\mathbf{B}$ such that $h(a_i) = b_i$ for all $i = 1, \ldots, r$.

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 [8, 11].

Let $\mathbf{A}$ be a τ-structure. The *incidence multigraph* of $\mathbf{A}$ 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 example, if τ is the signature of digraphs then τ-trees are exactly the oriented trees, i.e., digraphs obtained from trees by orienting each edge in some way.

A hypergraph H is a pair $(V(H), E(H))$ consisting of a finite nonempty set $V(H)$ of vertices or nodes and a set $E(H) \subseteq \mathcal{P}(V(H))$ of hyperedges with $\cup E(H) = V(H)$. A path on a hypergraph H from u to v is a sequence $u = w_0, e_1, w_1, e_2, \ldots, e_k, w_k = v$ where $e_1, \ldots, e_k$ are distinct hyperedges, $w_0, \ldots, w_k$ are distinct vertices, and $\{v_{i-1}, v_i\} \subseteq e_i$ for every $1 \leq i \leq k$.

With this notion of path, the definitions of cycle, connectedness and connected component can be transferred from graphs to hypergraphs in a natural way. We say that a hypergraph H is a tree if it is connected and has no cycles. With every structure $\mathbf{A}$, one can associate a hypergraph $H(\mathbf{A})$ whose nodes are the elements of A and the hyperedges are sets of elements appearing in a tuple in some relation in $\mathbf{A}$. Note that $H(\mathbf{A})$ is a tree whenever $\mathbf{A}$ is a τ-tree, but the converse is not always true. In general, the shape of a structure can be much more complicated than that of its associated hypergraph. However, for trees, the hypergraph gives a rather faithful representation. Indeed, if $\mathbf{A}$ is a tree and $m \geq 2$, then there is a complete correspondence between the set of hyperedges of $H(\mathbf{A})$ of cardinality m and the set of m-ary tuples in relations in $\mathbf{A}$.

The *Gaifman graph* $G(\mathbf{A})$ of a structure $\mathbf{A}$ is defined to have the same universe (set of vertices) as $\mathbf{A}$ and the edges of $G(\mathbf{A})$ are the pairs (a, a') of distinct elements such that a and a' appear in the same tuple in some relation in $\mathbf{A}$. The *pathwidth* of a graph $G = (V, E)$, denoted pw(G), is the minimum number k such that there is a sequence $X_1, \ldots, X_t$ of sets of vertices of G with the following properties: (i) $\bigcup_{i=1}^{t} X_i = V$, (ii) if $e = (u, v) \in E$ then $u, v \in X_i$ for some i, (iii) if $i < j < l$ then $X_i \cap X_l \subseteq X_j$, and (iv) $\max\{|X_i| : 1 \leq i \leq t\} = k+1$. In this paper, we say that the pathwidth of a structure $\mathbf{A}$, denoted pw($\mathbf{A}$), is equal to pw(G($\mathbf{A}$)). A finer notion of pathwidth for structures can be found in [8], but the concept of bounded pathwidth is the same.

2.2. Datalog

We now briefly describe the basics of Datalog (for more details, see, e.g., [24]). 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 the 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 the body, and it is called *monadic* if each IDB is at most unary.

When using Datalog to study CSP($\mathbf{B}$), one usually speaks of the definability of co-CSP($\mathbf{B}$) in Datalog or its fragments (because any class definable in Datalog must be closed under extension). We now give two examples of Datalog programs defining classes of the form co-CSP($\mathbf{B}$), more examples can be found in [8, 13, 25].

Example 1. *Let $\mathbf{B}_{3H}$ be the structure with universe $\{0, 1\}$, one unary relation $U^{\mathbf{B}_{3H}} = \{1\}$ and two ternary relations $P^{\mathbf{B}_{3H}} = \{0, 1\}^3 \setminus \{(1, 1, 0)\}$ and*

$N^{\mathbf{B}_{3H}} = \{0,1\}^3 \setminus \{(1,1,1)\}$. *It is easy to see that every Horn 3-CNF formula* φ *with variables* $x_1, \ldots, x_n$ *can be represented as a structure* $\mathbf{A}_\varphi$ *with universe* $\{x_1, \ldots, x_n\}$ *and relations* $U^{\mathbf{A}_\varphi}$, $P^{\mathbf{A}_\varphi}$, $N^{\mathbf{A}_\varphi}$ *where* $U^{\mathbf{A}_\varphi}$ *is the set of all positive unit clauses (in* φ*),* $P^{\mathbf{A}_\varphi}$ *is the set of all clauses of the form* $(\neg x \vee \neg y \vee z)$*, and* $N^{\mathbf{A}_\varphi}$ *is the set of all clauses of the form* $(\neg x \vee \neg y \vee \neg z)$*. Clearly, we have* $\mathbf{A}_\varphi \to \mathbf{B}_{3H}$ *if and only if* φ *is satisfiable. Hence* Horn 3-Sat *is precisely* CSP($\mathbf{B}_{3H}$)*. It is well known that* Horn 3-Sat *can be solved by the unit propagation algorithm which can be represented as the following Datalog program (where* G *is the goal predicate).*

$$\begin{array}{lll} T(X) & :- & U(X) \\ T(Z) & :- & P(X,Y,Z), T(X), T(Y) \\ G & :- & N(X,Y,Z), T(X), T(Y), T(Z) \end{array}$$

Example 2. *Fix a number* $k \geq 3$*, and let* $\mathbf{B}_{ihs}$ *denote the Boolean structure with three relations, unary* $U^{\mathbf{B}_{ihs}} = \{0\}$*, binary* $O^{\mathbf{B}_{ihs}} = \{(0,0),(0,1),(1,1)\}$ *(the order relation), and* k*-ary* $W^{\mathbf{B}_{ihs}} = \{0,1\}^k \setminus \{(0,\ldots,0)\}$*. These relations are basic* implicative hitting-set bounded *relations, as introduced in [12]. It can be checked directly that the following program describes* co-CSP($\mathbf{B}_{ihs}$)*.*

$$\begin{array}{lll} Z(X) & :- & U(X) \\ Z(X) & :- & O(X,Y), Z(Y) \\ G & :- & W(X_1, X_2, \ldots, X_k), Z(X_1), Z(X_2), \ldots, Z(X_k) \end{array}$$

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., [11]).

The notion of a polymorphism plays the key role in the algebraic approach to the CSP. The polymorphisms of a structure are known to determine the complexity of CSP($\mathbf{B}$) as well as definability of co-CSP($\mathbf{B}$) in Datalog and in several fragments, including monadic Datalog and linear Datalog (see [8, 27]). Many algebraic sufficient conditions for definability of co-CSP($\mathbf{B}$) in various fragments of Datalog are known (see [8]).

Let us now define several types of operations that will be used in this paper. An n-ary operation f is called *idempotent* if $f(x, \ldots, x) = x$ for all x and *totally symmetric* if $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ whenever $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$. An n-ary ($n \geq 3$) operation is called an *NU (near-unanimity)* operation if it satisfies the identities $f(y, x, \ldots, x, x) = f(x, y, \ldots, x, x) = \ldots = f(x, x, \ldots, x, y) = x$. A ternary NU operation is called a *majority* operation.

2.4. Dualities

A comprehensive treatment of dualities for the CSP can be found in the survey [8].

Definition 3. *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 trees, or structures of bounded pathwidth or of bounded treewidth, then $\mathbf{B}$ is said to have tree (bounded pathwidth, bounded treewidth, respectively) duality. A structure with a finite obstruction set is said to have finite duality. It is known (see [8]) 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 [18]. To state the result, we need the following construction: for a τ-structure $\mathbf{B}$, define a τ-structure $\mathbf{U}(\mathbf{B})$ (sometimes referred to as the *power structure*) 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 there exists a relation $R' \subseteq R^{\mathbf{B}}$ such that, for each $j = 1, \ldots, r$, we have $\mathrm{pr}_j(R') = A_j$, where $\mathrm{pr}_j(R') = \{a_j \mid (a_1, \ldots, a_j, \ldots, a_r) \in R'\}$.

Theorem 4. *[18] 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.*

3. A characterisation theorem

The main result of this paper is a characterisation of structures that have an obstruction set consisting of trees of bounded pathwidth, in the spirit of Theorem 4. Before stating the theorem, we will define, as in the four conditions of Theorem 4, (i) some new structures that will be involved in obstruction sets, (ii) a new class of Datalog programs, (iii) a new "power structure", and (iv) new operations for the algebraic characterisation.

3.1. Cattrees

A subhypergraph of $H = (V(H), E(H))$ is any hypergraph $(V(H'), E(H'))$ with $E(H') \subseteq E(H)$. In graph theory, a caterpillar is a tree which becomes a path after all its leaves are removed. We say that a hypergraph is a *caterpillar* if it is a tree and its hyperedges can be ordered $e_1, \ldots, e_n$ in such a way that two consecutive edges share (exactly) one element. A hyperedge e of a caterpillar is an *extreme* if there exists such ordering with $e = e_1$.

Let H be a hypergraph which is a tree and let E and T be connected subhypergraphs of H (and hence trees) that have no hyperedge in common. We say that T is *cut off* by E in H if T contains exactly one hyperedge e_T with elements of $V(E)$ and is maximal with respect to that property. We shall call the hyperedge e_T the *hyperedge connecting T to E*.

Definition 5. *Let H be a hypergraph which is a tree, let e be a hyperedge of H and $k \geq 1$. Then the pair (H, e) is a k-cattree if:*

1. *H is a caterpillar, and e is an extreme of H, or*
2. *$k > 1$ and there is subhypergraph E of H which is a caterpillar with extreme e and, for every subtree T cut off by E, (T, e_T) is a $(k-1)$-cattree.*

A tree H is a k-cattree if (H, e) is a k-cattree for some $e \in E(H)$.

A structure $\mathbf{T}$ is a k-cattree if $\mathbf{T}$ is a tree and $H(\mathbf{T})$ is a k-cattree.

Note that a 1-cattree is simply a caterpillar. An example of a 2-cattree hypergraph is shown on Fig. 1.

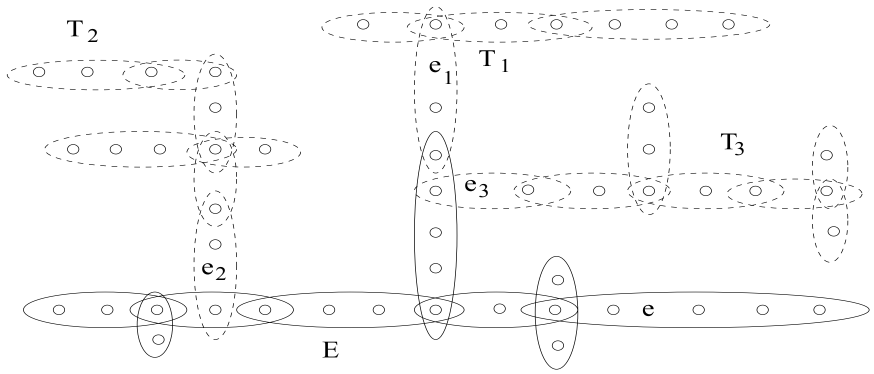

Figure 1: An example of a 2-cattree.

The above definition is new, but it is very close (in the case of graphs) to the definition of a k-caterpillar from [3] (the difference is that, in their definition, E is a path rather than a caterpillar). In Section 5, we will show that the new hypergraph parameter is natural by characterising it via a natural variation of a hypergraph searching game, which is a result of independent interest.

The following result connects the cattree parameter and the pathwidth of a tree.

Theorem 6. *If $\mathbf{A}$ is a τ-tree, r is the maximum arity in τ, and k is the minimum number such that $\mathbf{A}$ is a k-cattree, then $\frac{1}{2}(k-1) \leq \mathrm{pw}(\mathbf{A}) + 1 \leq rk$.*

Theorem 6 will immediately follow from Proposition 30, Remark 34, Theorem 40, and Corollary 43 that can be found in Section 5. More precisely, Corollary 43 gives an inequality for certain hypergraph parameters, while the other three statements connect these parameters with the numbers in Theorem 6. Note that Theorem 6 implies that any class of τ-trees has bounded pathwidth if and only if it consists of k-cattrees for some k. We say that a structure $\mathbf{B}$ has *k-cattree duality* if it has an obstruction set consisting of k-cattrees.

3.2. Layered monadic Datalog

For brevity, we will call monadic Datalog programs with at most one EDB per rule *tree programs*. For $k \geq 1$, a *k-layered tree program* is a tree program P in which the IDBs can be partitioned in k subsets, so that we speak of the level of an IDB, such that in every rule of P the following holds:

- If i is the level of the IDB in the head of the rule then the body of the rule can contain no IDB of level higher than i and at most one IDB of level i.

This means that, when applying a rule in the run of such a program, one can assume that all, but possibly one, IDBs in its body are already fully evaluated (i.e., will not change in this run). In a sense, this is a stratification of monadic Datalog (with at most one EDB per rule) by the use of non-linearity, somewhat akin to the standard stratification of Datalog($\neg$) by the use of negation [21].

Note that a k-layered tree program is in general not linear, but it is easy to see that, for any such program, there exists a linear, though not necessarily monadic, Datalog program that accepts precisely the same class of structures. Roughly, this program can be obtained by forming non-unary predicates whose co-ordinates correspond to the unary predicates of the original program, so that multiple unary IDBs in a rule could be replaced by a single non-unary IDB.

Recall the CSPs and Datalog programs from Examples 1 and 2. The tree program in Example 1 is not k-layered (for any k) because the IDB T appears twice in the body of the middle rule and also in the head of the same rule, so T cannot be assigned a level. In fact, co-CSP($\mathbf{B}_{3H}$) cannot be defined by a k-layered tree program because, as follows from [2], it cannot even be defined by a linear Datalog program. The tree program in Example 2 is 2-layered, the IDB Z has level 1, and the goal predicate G has level 2. Moreover, co-CSP($\mathbf{B}_{ihs}$) cannot be defined by a 1-layered tree program because $\mathbf{B}_{ihs}$ is easily seen not to have a majority polymorphism which is a necessary condition for such definability [10].

3.3. A new power structure

Let $\mathbf{B}$ be a τ-structure, let $R^{\mathbf{B}}$ be a relation in $\mathbf{B}$ of arity, say r, and let $(\mathcal{S}_1, \ldots, \mathcal{S}_r)$ be an r-tuple of families of subsets of B and let $(G_1, \ldots, G_r)$ be an r-tuple of subsets of B. We say that $(\mathcal{S}_1, \ldots, \mathcal{S}_r)$ is *coherent* with $R^{\mathbf{B}}$ in ground $(G_1, \ldots, G_r)$ if, for all $j, l = 1, \ldots, r$ we have

1. $\mathrm{pr}_l(R^{\mathbf{B}} \cap (G_1 \times \ldots \times G_r)) \in \mathcal{S}_l$, and
2. $\mathrm{pr}_l(R^{\mathbf{B}} \cap (G_1 \times \ldots \times G_{j-1} \times S \times G_{j+1} \times \ldots \times G_r)) \in \mathcal{S}_l$ for any $S \in \mathcal{S}_j$.

For $k \geq 1$, we construct a structure $\mathbf{C}_k(\mathbf{B})$ as follows.

- the elements of $\mathbf{C}_k(\mathbf{B})$ are the sequences $\overline{\mathcal{S}} = (\mathcal{S}[1], \ldots, \mathcal{S}[k])$ of families of *non-empty* subsets of B in which every *"level"* $\mathcal{S}[i]$ is closed under inverse inclusion, i.e., if $S \in \mathcal{S}[i]$ and $S \subseteq S'$ then $S' \in \mathcal{S}[i]$.
- for each r-ary $R \in \tau$, we have $(\overline{\mathcal{S}}_1, \ldots, \overline{\mathcal{S}}_r) \in R^{\mathbf{C}_k(\mathbf{B})}$ iff, for all $1 \leq i \leq k$, $(\overline{\mathcal{S}}_1[i], \ldots, \overline{\mathcal{S}}_r[i])$ is coherent with $R^{\mathbf{B}}$ in ground $(\bigcap \overline{\mathcal{S}}_1[i-1], \ldots, \bigcap \overline{\mathcal{S}}_r[i-1])$ (assuming that $\overline{\mathcal{S}}_j[0] = \{B\}$ for all $j = 1, \ldots, r$).

3.4. Layered polymorphisms

Let us now generalise the notion of an m-ABS operation from [10]. We call a operation f of arity $k \cdot m \cdot n$ on B *k-layered m-block symmetric* if it satisfies the following condition:

$$\begin{aligned} f(&\overbrace{x_{11}^{(1)}, \ldots, x_{1m}^{(1)}}^{S_1^{(1)}}, \ldots, \overbrace{x_{n1}^{(1)}, \ldots, x_{nm}^{(1)}}^{S_n^{(1)}}, \ldots, \overbrace{x_{11}^{(k)}, \ldots, x_{1m}^{(k)}}^{S_1^{(k)}}, \ldots, \overbrace{x_{n1}^{(k)}, \ldots, x_{nm}^{(k)}}^{S_n^{(k)}}) = \\ = f(&\underbrace{y_{11}^{(1)}, \ldots, y_{1m}^{(1)}}_{T_1^{(1)}}, \ldots, \underbrace{y_{n1}^{(1)}, \ldots, y_{nm}^{(1)}}_{T_n^{(1)}}, \ldots, \underbrace{y_{11}^{(k)}, \ldots, y_{1m}^{(k)}}_{T_1^{(k)}}, \ldots, \underbrace{y_{n1}^{(k)}, \ldots, y_{nm}^{(k)}}_{T_n^{(k)}}) \end{aligned}$$

whenever $\{S_1^{(l)}, \ldots, S_n^{(l)}\} = \{T_1^{(l)}, \ldots, T_n^{(l)}\}$ for each "level" l where, for all i, $S_i^{(l)} = \{x_{i1}^{(l)}, \ldots, x_{im}^{(l)}\}$ and $T_i^{(l)} = \{y_{i1}^{(l)}, \ldots, y_{im}^{(l)}\}$.

Equivalently, the above condition can be stated as follows: the value

$$f(x_{11}^{(1)}, \ldots, x_{1m}^{(1)}, \ldots, x_{n1}^{(1)}, \ldots, x_{nm}^{(1)}, \ldots, x_{11}^{(k)}, \ldots, x_{1m}^{(k)}, \ldots, x_{n1}^{(k)}, \ldots, x_{nm}^{(k)})$$

depends only on the sequence $\mathcal{S}_1, \ldots, \mathcal{S}_i, \ldots, \mathcal{S}_k$ where $\mathcal{S}_i = \{S_1^{(i)}, \ldots, S_n^{(i)}\}$, and $S_j^{(i)} = \{x_{j1}^{(i)}, \ldots, x_{jm}^{(i)}\}$, $j = 1, \ldots, n$. Therefore, we will often denote this value by $f(\mathcal{S}_1, \ldots, \mathcal{S}_i, \ldots, \mathcal{S}_k)$.

Let us call a sequence $\mathcal{S}_1, \ldots, \mathcal{S}_k$ *nested* if either $k = 1$ or, for each $1 \leq j < k$, every set in $\mathcal{S}_{j+1}$ is a subset of every set in $\mathcal{S}_j$. We say that a k-layered m-block symmetric operation f is a *k-layered m-ABS* operation if the following *absorption* property holds: for any $1 \leq i \leq k$ and for any nested sequence $\mathcal{S}_1, \ldots, \mathcal{S}_k$ we have

$$f(\mathcal{S}_1, \ldots, \mathcal{S}_i, \ldots, \mathcal{S}_k) = f(\mathcal{S}_1, \ldots, \mathcal{S}'_i, \ldots, \mathcal{S}_k)$$

where $\mathcal{S}'_i$ is any subset of $\mathcal{S}_i$ obtained by removing any element (i.e., a subset of B) in $\mathcal{S}_i$ that entirely contains some other element in $\mathcal{S}_i$.

It is easy to check that 1-layered m-ABS operations are exactly the m-ABS operations from [10].

Example 7. *Let $B = \{0,1\}^k$. In this example we will think of elements of B as k-columns of Boolean values. Consider the operation f on B such that*

$$f(x_{11}^{(1)},\dots,x_{1m}^{(1)},\dots,x_{n1}^{(1)},\dots,x_{nm}^{(1)},\dots,x_{11}^{(k)},\dots,x_{1m}^{(k)},\dots,x_{n1}^{(k)},\dots,x_{nm}^{(k)}) =$$

$$\left(\begin{array}{r} (\bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(1)}[1]) \wedge (\bigwedge_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(2)}[1]) \wedge \dots \wedge (\bigwedge_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(k)}[1]) \\ (\bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(2)}[2]) \wedge \dots \wedge (\bigwedge_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(k)}[2]) \\ \vdots \\ (\bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(k)}[k]) \end{array}\right)$$

where $x_{ij}^{(w)}[l]$ denotes the l-th component of variable $x_{ij}^{(w)}$.

It can be directly verified that f is a k-layered m-ABS operation. Indeed, since the operations $\bigvee$ and $\bigwedge$ are totally symmetric, it is easy to see that, for any fixed $1 \le l \le k$, the overall effect of the variables of the form $x_{ij}^{(l)}$ on the value of f (which corresponds to the l-th "column" in the above "matrix") depends only on $\mathcal{S}_l = \{S_1^{(l)},\dots,S_n^{(l)}\}$, where $S_i^{(l)} = \{x_{i1}^{(l)},\dots,x_{im}^{(l)}\}$, $i = 1,\dots,n$.

Now let us turn our attention to the absorption property. Let us assume that an input of the function f constitutes a nested sequence. This implies that, for any two levels $1 \le l \le l'$, coordinate $1 \le p \le l$, and blocks $1 \le i, i' \le n$, we have that $\bigwedge_{j=1}^m x_{ij}^{(l)}[p] \le \bigwedge_{j=1}^m x_{i'j}^{(l')}[p]$ which implies that

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} x_{ij}^{(l)}[p] \le \bigwedge_{i'=1}^{n} \bigwedge_{j=1}^{m} x_{i'j}^{(l')}[p]$$

Consequently, the value of f for nested sequences is:

$$\left(\begin{array}{c} \bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(1)}[1] \\ \bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(2)}[2] \\ \vdots \\ \bigvee_{i=1}^n \bigwedge_{j=1}^m x_{ij}^{(k)}[k] \end{array}\right)$$

With this in hand, the absorption property of f follows from the absorption property $(x \wedge y) \vee x = x$ of the operations $\vee$ and $\wedge$.

3.5. Main Theorem

Theorem 8. *The following conditions are equivalent for any structure $\mathbf{B}$ and any $k \ge 1$:*

1. *$\mathbf{B}$ has k-cattree duality;*
2. *$\text{co-CSP}(\mathbf{B})$ is definable by a k-layered tree program;*
3. *$\mathbf{C}_k(\mathbf{B})$ admits a homomorphism to $\mathbf{B}$;*
4. *for every $m, n \ge 1$, $\mathbf{B}$ has an mkn-ary k-layered m-ABS polymorphism.*

Note that any structure $\mathbf{B}$ satisfying the conditions of the above theorem has bounded pathwidth duality, by Theorem 6. Therefore, by results of [13], co-CSP($\mathbf{B}$) is definable in linear Datalog and CSP($\mathbf{B}$) belongs to the complexity class **NL** for any such structure.

We prove Theorem 8 through a series of lemmas, main lemmas being Lemmas 13, 18, 22. Note that Theorem 8 for the case $k = 1$ was the main result of [10].

First, we relate k-cattree duality with k-layered tree programs. 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 [8, 18]. The canonical k-layered tree program for a structure $\mathbf{B}$ is a k-layered tree program that contains, for every subset S of B and every $l = 1, \dots, k$, the IDB I^l_S of level l, and it consists of all the rules satisfying (the defining condition of a k-layered tree program and also) the requirement that if every IDB I^l_S in the rule is interpreted as S and every EDB R is interpreted as $R^{\mathbf{B}}$, then every assignment of elements of B to the variables that satisfies all the atomic formulas in the body must also satisfy the atomic formula in the head. Finally, declare $I^k_\emptyset$ to be the goal predicate (or equivalently include the goal predicate G along with the rule $G :- I^k_\emptyset(x)$). Note that IDBs corresponding to some of the subsets S (those that cannot be defined by a primitive positive first-order formula in $\mathbf{B}$) are redundant in the sense that they are never used by the program in any derivation of the goal predicate on any input, so sometimes such IDBs are not included in the definition of a canonical program. Obviously, this does not affect the class of structures accepted by the program.

Example 9. *Take the poset $\mathbf{Q}$ whose Hasse diagram is shown in Fig. 2. Let structure $\mathbf{Q}_c$ be obtained from $\mathbf{Q}$ by adding all elements of the universe as singleton unary relations. $\mathbf{Q}_c$ has domain $\{a, a', b, b', c\}$ and the following relations: $U_a = \{a\}, U_{a'} = \{a'\}, U_b = \{b\}, U_{b'} = \{b\}, U_c = \{c\}$, $R = \{(a, a), (a', a'), (b, b), (b', b'), (c, c), (a, b), (a, b'), (a, c), (a', b), (a', b'), (a', c), (b, c), (b', c)\}$. The canonical 2-layered tree program, P, for $\mathbf{Q}_c$ has EDBs R and U_v, for each $v \in \{a, a', b, b', c\}$; and it has IDBs $I^{(1)}_S, I^{(2)}_S$ for each $S \subseteq \{a, a', b, b', c\}$. On level 1 all the rules will have only IDBs $I^{(1)}_S$, for all subsets S of the domain of $\mathbf{Q}_c$, and all rules that are valid implications when the IDBs $I^{(1)}_S$ are interpreted as the subsets S. The following are examples of rules in P:*

$$\begin{array}{rcl@{\qquad\qquad}rcl}
I^{(1)}_{\{b\}}(x) & :- & U_b(x) & I^{(1)}_{\{b,c\}}(y) & :- & R(x,y), I^{(1)}_{\{b\}}(x) \\
I^{(1)}_{\{a,a',b\}}(x) & :- & R(x,y), I^{(1)}_{\{b\}}(y) & I^{(1)}_{\{a,b',c\}}(x) & :- & I^{(1)}_{\{a,c\}}(x) \\
I^{(1)}_{\{a,a',b\}}(x) & :- & R(x,y), I^{(1)}_{\{a,a',b'\}}(y) & I^{(1)}_{\{c\}}(x) & :- & U_c(x), I^{(1)}_{\{b,c\}}(x)
\end{array}$$

On level 2, P contains the same type of rules it has on level 1, but for the IDBs $I^{(2)}_S$, and it contains extra rules that involve IDBs $I^{(2)}_S$ and $I^{(1)}_S$. These rules have at most one IDB of level 2, $I^{(2)}_S$, in the body, but can have several IDBs of level 1, $I^{(1)}_S$. As before, a rule appears in P if, when the IDBs in the rule *are interpreted as the respective subsets of* $\{a, a', b, b', c\}$ *and the EDB in the rule is interpreted as the respective relation in* $\mathbf{Q}_c$*, then every assignment of elements of* $\{a, a', b, b', c\}$ *to the variables in the rule that satisfies the body must also satisfy the head of the rule. Examples of rules of level* 2 *in* P *are:*

$$\begin{array}{lll} I^{(2)}_{\{a\}}(x) & :- & U_a(x) \\ I^{(2)}_{\{a\}}(y) & :- & R(x,y), I^{(2)}_{\{a\}}(x), I^{(1)}_{\{a,a',b\}}(y), I^{(1)}_{\{a,a',b'\}}(y) \\ I^{(2)}_{\emptyset}(x) & :- & U_{\{a'\}}(x), I^{(2)}_{\{a\}}(x) \end{array}$$

The program P *ends with the rule* $G : -I^{(2)}_{\emptyset}(x)$*, with* G *the goal predicate.*

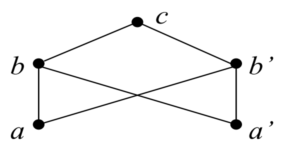

Figure 2: The poset $\mathbf{Q}$ from Example 9.

Note that $\mathbf{B}$ is not accepted by the canonical program for itself. Indeed, by construction, a derivation of the goal predicate 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, and the fact that any class definable in Datalog is closed under homomorphism, implies the following fact.

Lemma 10. *If the canonical* k*-layered tree program for* $\mathbf{B}$ *accepts a structure* $\mathbf{A}$ *then* $\mathbf{A} \not\to \mathbf{B}$.

Lemma 11. *If* $\mathbf{D}$ *is a tree,* e *is a hyperedge of* $H = H(\mathbf{D})$ *such that* (H, e) *is a* k*-cattree and* a *is any element in* e*, then the canonical* k*-layered tree program for* $\mathbf{B}$ *derives, on* $\mathbf{D}$*, the fact* $I^k_{S^{\mathbf{D}}_a}(a)$*, where* $S^{\mathbf{D}}_a$ *is the set* $\{h(a) \mid h : \mathbf{D} \to \mathbf{B}\}$.

Proof. The lemma is proved by induction on k. We prove only the inductive step and omit the proof of the base case $k = 1$ which is very similar (and can be found in Lemma 8 of [10]).

Since (H, e) is a k-cattree, there exists a caterpillar E such that E and e satisfy condition (2) of Definition 5. We shall prove the statement by induction on the number n of hyperedges of E. Again, we shall prove only the inductive step as the argument for the case base $n = 1$ is similar and, indeed, simpler.

Set $E' = E \backslash \{e\}$ and let H' be the connected component of $H \backslash \{e\}$ containing all hyperedges of E'. There exists an element v occurring both in e and $V(H')$. The element v must appear in an extreme e' of E' and hence by induction hypothesis (on the size of E) the k-layered tree program can derive, on the induced substructure $\mathbf{D}[\cup H']$ (and hence on $\mathbf{D}$), the fact $I^k_{S^{\mathbf{D}[\cup H']}_v}(v)$.

Now, let $T_1, \ldots, T_J$ be the set of subtrees of H cut off by $\{e\}$ and not containing v. For every $j = 1, \ldots, J$, let e_j be the edge of T_j connecting T_j with $\{e\}$ and let $\{v_j\}$ be $e_j \cap e$. By the definition of k-cattree, for every $j = 1, \ldots, J$ (T_j, e_j) is a $(k-1)$-cattree and by induction hypothesis (on k), the canonical $(k-1)$-layered program (and hence the k-layered program as well) can derive, on $\mathbf{D}[\cup T_j]$ (and hence on $\mathbf{D}$), the fact $I^{k-1}_{S^{\mathbf{D}[\cup T_j]}_{v_j}}(v_j)$.

Assume first that $|e| \geq 2$. In this case each node of H is in $e \cup H' \cup \bigcup_{j \in J} T_j$. In terms of homomorphisms this implies that if $R(u_1, \ldots, u_n)$ is the tuple of $\mathbf{D}$ corresponding to hyperedge e, and we let $v = u_m$ and for every $j \in 1, \ldots, J$ we let $v_j = u_{i_j}$ then we have that the relation $\{(h(u_1), \ldots, h(u_n)) \mid h : \mathbf{D} \to \mathbf{B}\}$ is equal to

$$\{(b_1, \ldots, b_n) \in R^{\mathbf{B}} : b_m \in S^{\mathbf{D}[\cup H']}_v \text{ and } b_{i_j} \in S^{\mathbf{D}[\cup T_j]}_{v_j} \text{ for all } j = 1, \ldots, J\}$$

Hence, if $a = u_l$ then the canonical k-layered program contains the rule

$$I^k_{S^{\mathbf{D}}_a}(x_l) : -R(x_1, \ldots, x_n), I^k_{S^{\mathbf{D}[\cup H']}_v}(x_m), I^{k-1}_{S^{\mathbf{D}[\cup T_1]}_{v_1}}(x_{i_1}), \ldots, I^{k-1}_{S^{\mathbf{D}[\cup T_J]}_{v_J}}(x_{i_J})$$

which would allow the canonical program to derive $I^k_{S^{\mathbf{D}}_a}(a)$.

If $|e| = 1$ then the reasoning is very similar. Clearly $a = v = v_1 = \cdots = v_J$, $e = \{a\}$, and we have that every node of H is in $H' \cup \bigcup_{j \in J} T_j$. Hence

$$\{h(a) \mid h : \mathbf{D} \to \mathbf{B}\} = S^{\mathbf{D}[\cup H']}_a \cap \bigcap_{j=1,\ldots,J} S^{\mathbf{D}[\cup T_j]}_a$$

Hence, the canonical k-layered program contains the rule

$$I^k_{S^{\mathbf{D}}_a}(x) : -I^k_{S^{\mathbf{D}[\cup H']}_a}(x), I^{k-1}_{S^{\mathbf{D}[\cup T_1]}_a}(x), \ldots, I^{k-1}_{S^{\mathbf{D}[\cup T_J]}_a}(x)$$

which would allow the canonical program to derive $I^k_{S^{\mathbf{D}}_a}(a)$. This concludes the proof.

Note that by definition of $S^{\mathbf{D}[\cup H']}_a$ we have that $S^{\mathbf{D}[\cup H']}_a \subseteq \{b \in B \mid (b, \ldots, b) \in R^{\mathbf{B}}\}$. Consequently, the canonical k-layered program contains also the rule

$$I^k_{S^{\mathbf{D}}_a}(x) : -R(x, \ldots, x), I^k_{S^{\mathbf{D}[\cup H']}_a}(x), I^{k-1}_{S^{\mathbf{D}[\cup T_1]}_a}(x), \ldots, I^{k-1}_{S^{\mathbf{D}[\cup T_J]}_a}(x)$$

which could also be used to derive $I^k_{S^{\mathbf{D}}_a}(a)$. This remark will later be used to justify Remark 15 which refers to this proof. ■

Before continuing we need to take care of some small technicalities which will allow us to restrict further the shape of the rules.

Lemma 12. *Let* $\mathbf{B}$ *be a relational structure. For every* k*-layered tree program* P *that defines* co-CSP($\mathbf{B}$)*, there exists an equivalent* k*-layered tree program* P' *such that every rule contains an EDB and, in addition, every variable appearing in a rule also appears in the EDB in the rule.*

Proof. Let $t_0 : -t_1, \ldots, t_n$ be a rule of P and assume that the set of variables can be partitioned in two proper subsets X, Y such that no atomic formula in the body contains variables from both sets simultaneously. Since the order of the atomic formulas does not alter a rule we can assume without loss of generality that there exists $1 \leq i < n$ such that every variable that occurs in some predicate t_j with $j \leq i$ belongs to X and every variable that occurs in some predicate t_j with $j > i$ belongs to Y. Also, we can assume that if the predicate in the head t_0 is unary then the variable x occurring in t_0 belongs to X.

We shall prove that one can write an equivalent program in which no rule can be partitioned in this way. Observe that, since the IDBs are unary, this will imply that if a rule contains an EDB then necessarily every variable in the rule must appear in it. Assume, for contradiction, that, among all k-layered tree programs that define co-CSP($\mathbf{B}$), P has the smallest number of rules where the variables can be partitioned as above, and that $t_0 : -t_1, \ldots, t_n$ is such a rule in P, with X, Y and i defined as above. Consider the program P_Y obtained by replacing the rule $t_0 : -t_1, \ldots, t_n$ by the rule $G : -t_{i+1}, \ldots, t_n$ where G is the goal predicate. Note that P_Y will accept everything that P accepts. If the P_Y is equivalent to P then we get a contradiction with the choice of P. Otherwise, there exists some structure $\mathbf{A}_Y$ which is homomorphic to $\mathbf{B}$ but accepted by P_Y. Now consider the program P_X, obtained by replacing, in P, the rule $t_0 : -t_1, \ldots, t_n$ by $t_0 : -t_1, \ldots, t_i$. We claim that P_X is equivalent to P. Again, it is easy to see that P_X accepts everything that P accepts (in fact, by using the same sequence of rules as P). To prove that P_X is equivalent to P, assume, for contradiction, that P_X accepts some structure $\mathbf{A}_X$ which is homomorphic to $\mathbf{B}$. Then we claim that P accepts the disjoint union $\mathbf{A}_X \oplus \mathbf{A}_Y$ of the structures $\mathbf{A}_X$ and $\mathbf{A}_Y$. In fact, it does so by replicating the run of P_X on input $\mathbf{A}_X$. To see this, notice first that if $\mathbf{A}_Y$ is accepted by P_Y then this is necessarily by application of the rule $G : -t_{i+1}, \ldots, t_n$. Let $s_Y : Y \to A_Y$ be the instantiation of the variables of the rule in this application. Hence, every time when there is some instantiation of the variables in X, $s_X : X \to A_X$, that allows one to use the rule $t_0 : -t_1, \ldots, t_i$ with input $\mathbf{A}_X$, s_X can be combined with s_Y to obtain an instantiation for the rule $t_0 : -t_1, \ldots, t_n$ with input $\mathbf{A}_X \oplus \mathbf{A}_Y$. Hence, P accepts $\mathbf{A}_X \oplus \mathbf{A}_Y$. Since P defines co-CSP($\mathbf{B}$), we have that $\mathbf{A}_X \oplus \mathbf{A}_Y$ is not homomorphic to $\mathbf{B}$, which contradicts the assumption that both $\mathbf{A}_X$ and $\mathbf{A}_Y$ are homomorphic to $\mathbf{B}$.

It only remains to see that we can get rid of all the rules that do not contain an EDB. By the previous considerations and the fact that we are dealing with monadic predicates we can assume that every rule not containing an EDB is of the form $t_0 : -I_1(x), \ldots, I_n(x)$ where I_i, $1 \leq i \leq n$ are IDBs. Consider now the Datalog program, P', obtained by replacing the rule $t_0 : -I_1(x), \ldots, I_n(x)$ by the family of rules of the form $t_0 : -I_1(x), \ldots, I_n(x), R(x_1, \ldots, x_r)$ where R is a r-ary EDB, $\{x_1, \ldots, x_r\}$ are different variables, and $x_i = x$ for some $1 \leq i \leq r$. It is easy to see that P accepts every structure accepted by P' because, whenever P' uses a newly introduced rule in a run, P can simulate it with an application of the old rule from which the new rule was obtained. We claim that, in fact, P' is equivalent to P. Assume towards a contradiction that P' fails to accept a structure $\mathbf{A}$ that is not homomorphic to $\mathbf{B}$. We can without loss of generality assume that $\mathbf{A}$ does not contain isolated nodes (such nodes never affect acceptance by a Datalog program or membership in CSP($\mathbf{B}$)), i.e., assume that every node participates in at least one tuple in some relation. Since $\mathbf{A}$ is not homomorphic to $\mathbf{B}$ we know that P would derive the goal predicate on input $\mathbf{A}$. We shall see that this derivation could be replicated by program P', reaching a contradiction. Indeed, every time that P needs to apply rule $t_0 : -I_1(x), \ldots, I_n(x)$ with x instantiated to element a in $\mathbf{A}$, we know that a is not isolated and henceforth there exist some $R \in \tau$ and $(a_1 \ldots, a_r) \in R^{\mathbf{A}}$ with $a = a_i$ for some $1 \leq i \leq r$. Hence P' could replicate this movement by applying rule $t_0 : -I_1(x), \ldots, I_n(x), R(x_1, \ldots, x_k)$ with $x_i = x$. ■

Lemma 13. *Let* $\mathbf{B}$ *be a structure and* $k \geq 1$*. The following are equivalent:*

1. co-CSP($\mathbf{B}$) *is definable by a* k*-layered tree program.*
2. $\mathbf{B}$ *has an obstruction set consisting of* k*-cattrees.*

Proof. ($\mathbf{1}$) $\Rightarrow$ ($\mathbf{2}$) Suppose that co-CSP($\mathbf{B}$) is defined by a k-layered tree program P. 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. We shall also assume that P satisfies the extra requirements given by Lemma 12.

If $\mathbf{A} \not\to \mathbf{B}$ then by the Sparse Incomparability Lemma [34] there is a structure $\mathbf{C}$ that is homomorphic to $\mathbf{A}$ but not to $\mathbf{B}$ that does not contain loops, i.e., such that, for every tuple $(a_1, \ldots, a_r)$ in any relation in $\mathbf{C}$, $a_i \neq a_j$ for all $1 \leq i \neq j \leq r$.

We show by induction on the number of levels k, that if I is an IDB of the k-th level and $I(a)$ is derived by the Datalog program on $\mathbf{C}$ then there exist some tree $\mathbf{T}$, some hyperedge e of $H(\mathbf{T})$, and some $t \in e$ such that $(H(\mathbf{T}), e)$ is a k-cattree, we have $\mathbf{T}, t \to \mathbf{C}, a$, and P derives $I(t)$ on $\mathbf{T}$.

It is convenient to do first the inductive step. Let us read the section of the derivation of $I(a)$ involving only rules whose IDB in the head has level k reversing the order of derivation. By our assumptions on the program P we obtain a sequence of the form:

$$\begin{array}{rcl}
I(a) & :- & R_1(c_1^1, \ldots, c_{r_1}^1),\ I_1(a_1),\ J_1^1(b_1^1), \ldots, J_1^{n_1}(b_1^{n_1}) \\
I_1(a_1) & :- & R_2(c_1^2, \ldots, c_{r_2}^2),\ I_2(a_2),\ J_2^1(b_2^1), \ldots, J_2^{n_2}(b_2^{n_2}) \\
 & \vdots & \\
I_{l-1}(a_{l-1}) & :- & R_l(c_1^l, \ldots, c_{r_l}^l),\ I_l(a_l),\ J_l^1(b_l^1), \ldots, J_l^{n_l}(b_l^{n_l}) \\
I_l(a_l) & :- & R_{l+1}(c_1^{l+1}, \ldots, c_{r_{l+1}}^{l+1}),\ J_{l+1}^1(b_{l+1}^1), \ldots, J_{l+1}^{n_{l+1}}(b_{l+1}^{n_{l+1}})
\end{array}$$

where the I_i's are (not necessarily different) IDBs of level k, the J_i^j's are (not necessarily different) IDBs of level smaller than k, R_i is an EBD of arity r_i, and every element of the form a_u or b_u^* is also one of c_*^u.

Consider the τ-structure $\mathbf{E}$ defined in the following way: The universe of $\mathbf{E}$ is the subset of $C \times \{1, \ldots, l{+}1\}$ containing all pairs (x, i) such that x appears in the body of the i-th step of the above derivation. Secondly, for every $i = 1, \dots, l+1$ add the tuple $((c^i_1, i), \dots, (c^i_{r_i}, i))$ to $R^{\mathbf{E}}_i$. Finally, for every $i = 1, \dots, l$ glue together the elements (a_i, i) and $(a_i, i+1)$. Clearly $H(\mathbf{E}, \{(c^1_1, 1), \dots, (c^1_{r_1}, 1)\})$ is a 1-cattree and the first projection defines a homomorphism $\mathbf{E}, (a, 1) \to \mathbf{C}, a$.

For every $1 \leq i \leq l+1$, $1 \leq j \leq n_i$, the predicates J^j_i are of smaller level and hence, by the induction hypothesis, there exists a structure $\mathbf{T}^j_i$, an hyperedge $e^j_i \in H(\mathbf{T}^j_i)$, and a distinguished element $t^j_i \in e^j_i$ such that $(H(\mathbf{T}^j_i), e^j_i)$ is a $(k-1)$-cattree, $\mathbf{T}, t^j_i \to \mathbf{C}, b^j_i$ and P derives $J^j_i(t^j_i)$ on $\mathbf{T}^j_i$.

Now construct a structure $\mathbf{T}$ as the disjoint union of $\mathbf{E}$ and all $\mathbf{T}^j_i$'s and glue every t^j_i to (b^j_i, i). Observe that since all the structures involved in the construction are homomorphic to $\mathbf{C}$ by homomorphisms such that every pair of glued elements have the same image, if we set $t = (a, 1)$ we have $\mathbf{T}, t \to \mathbf{C}, a$. Furthermore program P can derive $I(t)$ on $\mathbf{T}$ by replicating the derivation on the preimages of the homomorphism. Finally, according to rule (2) of Definition 5, $(H(\mathbf{T}), \{(c^1_1, 1), \dots, (c^1_{r_1}, 1)\})$ is a k-cattree.

The proof of the base case ($k = 1$) is very similar (and indeed simpler as we do not have to deal with IDBs of lower levels).

Finally, to complete the proof we assume (by rewriting our Datalog program if necessary) that any rule with the goal predicate in the head is of the form $G : -I(a)$ where I is an IDB. The claim implies that there exists a tree $\mathbf{T}$ homomorphic to $\mathbf{C}$ and hence to $\mathbf{A}$ such that P derives the goal predicate on $\mathbf{T}$ and hence $\mathbf{T} \not\to \mathbf{B}$.

(2) $\Rightarrow$ **(1)** Conversely, assume that $\mathbf{B}$ has an obstruction set of the form specified by condition (2) of this lemma. We claim that the canonical k-layered tree program defines co-CSP($\mathbf{B}$). By Lemma 10 the program never accepts a structure that homomorphically maps to $\mathbf{B}$. Now, let $\mathbf{A}$ be a structure not homomorphic to $\mathbf{B}$. By assumption there is a k-cattree $\mathbf{T}$ such that $\mathbf{T} \to \mathbf{A}$ and $\mathbf{T} \not\to \mathbf{B}$. As a direct consequence of Lemma 11, the canonical k-layered tree program accepts $\mathbf{T}$ and, since the set of accepted structures of a Datalog program is closed under homomorphism, it accepts $\mathbf{A}$ as well. ■

Corollary 14. *If* co-CSP($\mathbf{B}$) *is definable by a k-layered tree program then it is definable by the canonical one.*

Furthermore, by inspecting the proofs of Lemmas 11 and 13 we see that the rules used in the derivation of the goal predicate are of the restricted form specified in Lemma 12. Hence we have:

Remark 15. *If the canonical k-layered tree program for* $\mathbf{B}$ *can derive the goal predicate on a given structure* $\mathbf{A}$ *then it can also do so by using only rules that contain an EDB and, in addition, such that every variable appearing in a rule also appears in the EDB in the rule.*

Lemma 16. $\mathbf{C}_k(\mathbf{B})$ *is not accepted by the canonical k-layered tree program for* $\mathbf{B}$.

Proof. We will show by induction on the length of derivation that whenever the fact $I^l_S(\overline{\mathbb{S}})$ is derived by the program, we have $S \in \overline{\mathbb{S}}[l]$.

Assume first that $I^l_S(\overline{\mathbb{S}})$ 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 $I^l_S(\overline{\mathbb{S}}) : -R(\ldots, \overline{\mathbb{S}}, \ldots)$ where $\overline{\mathbb{S}}$ appears in the m-th component in the tuple on the right. Note that this tuple belongs to $R^{\mathbf{C}_k(\mathbf{B})}$. Then, by the definition of the canonical program, we have $S \supseteq \mathrm{pr}_m(R^{\mathbf{B}})$, which must be contained in $\overline{\mathbb{S}}_m[l]$ by the definition of $\mathbf{C}_k(\mathbf{B})$.

Assume now that $I^l_S(\overline{\mathbb{S}})$ is derived by a non-introductory rule of the form

$$I^l_S(X_i) : -R(X_1, \ldots, X_r), I^l_{S'}(X_j), I_{11}(X_1), \ldots, I_{1s_1}(X_1), \ldots, I_{r1}(X_r), \ldots, I_{rs_r}(X_r)$$

where each IDB I_{uv} is of level at most $l-1$. Observe that not all parts of the body (such as the IDB $I^l_{S'}(X_j)$) must be present. However, we will assume that the EDB $R(X_1, \ldots, X_r)$ is always present. This will be sufficient due to Remark 15.

Let $(\overline{\mathbb{S}}_1, \ldots, \overline{\mathbb{S}}_r)$ be the tuple in $R^{\mathbf{C}_k(\mathbf{B})}$ used with the rule to derive $I^l_S(\overline{\mathbb{S}})$. By the induction hypothesis, we have that (i) for each $u = 1, \ldots, r$ and $v = 1, \ldots, s_u$, the subset corresponding to IDB I_{uv} is included in $\overline{S}_u[i]$ where i is the level of I_{uv}, and hence by the construction of $\mathbf{C}_k(\mathbf{B})$, is in $\overline{S}_u[l-1]$. Also, by induction hypothesis, we have $S' \in \overline{\mathbb{S}}_j[l]$ (if $I^l_{S'}(X_j)$ is present). Now it follows directly from the fact $(\overline{\mathbb{S}}_1, \ldots, \overline{\mathbb{S}}_r) \in R^{\mathbf{C}_k(\mathbf{B})}$ (see the definition of $\mathbf{C}_k(\mathbf{B})$) and the definition of the canonical program that $S \in \overline{\mathbb{S}}[l]$.

Assume now that $\mathbf{C}_k(\mathbf{B})$ is accepted by the canonical program. Then the program can derive $I^l_\emptyset(\overline{\mathbb{S}})$ for some l and some $\overline{\mathbb{S}}$. By Remark 15, it can do so using only rules such as above. Then, as we just proved, the empty set belongs to $\overline{\mathbb{S}}[l]$ which is impossible by the definition of $\mathbf{C}_k(\mathbf{B})$. ■

Lemma 17. *A structure* $\mathbf{A}$ *is not accepted by the canonical* k*-layered tree program for* $\mathbf{B}$ *if and only if* $\mathbf{A} \to \mathbf{C}_k(\mathbf{B})$.

Proof. Assume first that $\mathbf{A} \to \mathbf{C}_k(\mathbf{B})$. Since the class of structures accepted by any Datalog program is closed under homomorphism, the required condition follows from Lemma 16.

Conversely, assume that $\mathbf{A}$ is not accepted by the program. Hence the canonical program stabilizes without deriving the goal predicate. For each element a of $\mathbf{A}$, define $\overline{\mathbb{S}}_a$ as follows: for each level $1 \le l \le k$, set $\overline{\mathbb{S}}_a[l] = \{S \mid I^l_S(a) \text{ is derived}\}$. It is easy to see that the family $\overline{\mathbb{S}}_a[l]$ is non-empty for any level l and any a that appears in a tuple in a relation in $\mathbf{A}$. Moreover, since the goal predicate is not derived, $I^l_\emptyset(a)$ is not derived either, and so each subset in a non-empty $\overline{\mathbb{S}}_a[l]$ is non-empty. It is straightforward to check that the mapping $h : A \to C_k(B)$ given by $h(a) = \overline{\mathbb{S}}_a$ (set $h(a)$ arbitrarily if a does not participate in any tuple) is a homomorphism from $\mathbf{A}$ to $\mathbf{C}_k(\mathbf{B})$. ■

Lemma 18. *For any structure* $\mathbf{B}$*,* co-CSP$(\mathbf{B})$ *is definable by a* k*-layered tree program if and only if* $\mathbf{C}_k(\mathbf{B})$ *admits a homomorphism to* $\mathbf{B}$*.*

Proof. Suppose that co-CSP($\mathbf{B}$) is definable by a k-layered tree program. By Lemma 16, $\mathbf{C}_k(\mathbf{B})$ is not accepted by the canonical program, and so $\mathbf{C}_k(\mathbf{B}) \to \mathbf{B}$.

Conversely, suppose that $\mathbf{C}_k(\mathbf{B}) \to \mathbf{B}$ and let $\mathbf{A}$ be an arbitrary structure. If $\mathbf{A}$ is not accepted by the canonical k-layered Datalog Program, then, by Lemma 17, we have that $\mathbf{A} \to \mathbf{C}_k(\mathbf{B})$ and it follows that $\mathbf{A} \to \mathbf{B}$ from transitivity of homomorphism. If $\mathbf{A}$ is accepted by the canonical k-layered tree program then $\mathbf{A} \not\to \mathbf{B}$ from Lemma 10. ■

Remark 19. *If $(\mathcal{S}_1, \dots, \mathcal{S}_r)$ and $(\mathcal{T}_1, \dots, \mathcal{T}_r)$ are coherent with $R^{\mathbf{B}}$ in ground $(G_1, \dots, G_r)$ then so are $(\mathcal{S}_1 \cup \mathcal{T}_1, \dots, \mathcal{S}_r \cup \mathcal{T}_r)$ and $(\mathcal{S}_1 \cap \mathcal{T}_1, \dots, \mathcal{S}_r \cap \mathcal{T}_r)$.*

Remark 20. *Let $(\mathcal{S}_1, \dots, \mathcal{S}_r)$ be a tuple such that for every $i = 1, \dots, r$, $\mathcal{S}_i$ is closed under inverse inclusion. If $(\mathcal{S}_1, \dots, \mathcal{S}_r)$ is coherent with $R^{\mathbf{B}}$ in ground $(G_1, \dots, G_r)$ then it is also such in ground $(G_1, \dots, G_{j-1}, G'_j, G_{j+1}, \dots, G_r)$ for every $j \in \{1, \dots, r\}$ and $G_j \subseteq G'_j$.*

Lemma 21. *For any structure $\mathbf{B}$ and any $k, m, n \geq 1$, the structure $\mathbf{C}_k(\mathbf{B})$ has a $(m \cdot k \cdot n)$-ary k-layered m-ABS polymorphism.*

Proof. Let f be the operation as in Example 7, but with $\vee$ and $\wedge$ replaced by $\cup$ and $\cap$, respectively. It can be straightforwardly verified that this operation is still a k-layered m-ABS operation. It remains to check that it is a polymorphism of $\mathbf{C}_k(\mathbf{B})$. First it is easy to observe, by using Remarks 19 and 20 that the component-wise intersection (i.e., the binary operation such that $(\overline{\mathcal{S}}_1 \cap \overline{\mathcal{S}}_2)[l] = \overline{\mathcal{S}}_1[l] \cap \overline{\mathcal{S}}_2[l]$ for all l) is a polymorphism of $\mathbf{C}_k(\mathbf{B})$. With a little bit of extra work we can obtain the following generalization of this result.

Let g_j be the binary operation on $C_j(B)$ such that $g_j(\overline{\mathcal{S}}_1, \overline{\mathcal{S}}_2)[l] = \overline{\mathcal{S}}_1[l] \cap \overline{\mathcal{S}}_2[l]$ for $1 \leq l \leq j-1$ and $g(\overline{\mathcal{S}}_1, \overline{\mathcal{S}}_2)[j] = \overline{\mathcal{S}}_1[j] \cup \overline{\mathcal{S}}_2[j]$. Furthermore, for $1 \leq i \leq j$, let $h_{i,j} : C_i(B) \times C_j(B) \to C_j(B)$ be defined by the rule $h_{i,j}(\overline{\mathcal{S}}_1, \overline{\mathcal{S}}_2)[l] = \overline{\mathcal{S}}_1[l] \cap \overline{\mathcal{S}}_2[l]$ for $1 \leq l \leq i$ and $h_{i,j}(\overline{\mathcal{S}}_1, \overline{\mathcal{S}}_2)[l] = \overline{\mathcal{S}}_2[l]$ for $i < l \leq j$.

Claim 1. *We have that (1) f is a polymorphism of $\mathbf{C}_j(\mathbf{B})$ and (2) $h_{i,j}$ is a homomorphism from $\mathbf{C}_i(\mathbf{B}) \times \mathbf{C}_j(\mathbf{B})$ to $\mathbf{C}_j(\mathbf{B})$.*

Proof. We prove part (1), part (2) is very similar. Let $(\overline{\mathcal{S}}_1, \dots, \overline{\mathcal{S}}_r)$ and $(\overline{\mathcal{T}}_1, \dots, \overline{\mathcal{T}}_r)$ be tuples in $R^{\mathbf{C}_j(\mathbf{B})}$. We need to show that $(g_j(\overline{\mathcal{S}}_1, \overline{\mathcal{T}}_1), \dots, g_j(\overline{\mathcal{S}}_r, \overline{\mathcal{T}}_r)) \in R^{\mathbf{C}_j(\mathbf{B})}$. For each level, we need to check the coherence condition from the definition of $\mathbf{C}_j(\mathbf{B})$. The coherence condition for level 1 follows directly from Remark 19. Assume now that $l > 1$. By definition, the l-th level $(\overline{\mathcal{S}}_1[l], \dots, \overline{\mathcal{S}}_r[l])$ of the first tuple is coherent with $R^{\mathbf{B}}$ in ground $(\bigcap \overline{\mathcal{S}}_1[l-1], \dots, \bigcap \overline{\mathcal{S}}_r[l-1])$ and hence it is also coherent in ground $(G_1, \dots, G_r)$ where $G_i = \bigcap(\overline{\mathcal{S}}_i[l-1] \cap \overline{\mathcal{T}}_i[l-1])$, $i = 1, \dots, r$, by Remark 20. The same reasoning shows that $(\overline{\mathcal{T}}_1[l], \dots, \overline{\mathcal{T}}_r[l])$ is also coherent in ground $(G_1, \dots, G_r)$. According to the definition of g_j, we need to distinguish cases $1 < l < j$ and $l = j$, but in both cases part (1) of the claim now follows from Remark 19. ■

Finally, by iterative application of the previous claim one can easily prove by (reverse) induction on $r = 1, \dots, k$ that the operation

$$f(\overline{\mathbb{S}}^{(r)}_{11}, \dots, \overline{\mathbb{S}}^{(r)}_{1m}, \dots, \overline{\mathbb{S}}^{(r)}_{n1}, \dots, \overline{\mathbb{S}}^{(r)}_{nm}, \dots, \overline{\mathbb{S}}^{(k)}_{11}, \dots, \overline{\mathbb{S}}^{(k)}_{1m}, \dots, \overline{\mathbb{S}}^{(k)}_{n1}, \dots, \overline{\mathbb{S}}^{(k)}_{nm})$$

defined as

$$\left(\begin{array}{r}
(\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r)}_{ij}[1]) \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r+1)}_{ij}[1]) \cap \dots \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(k)}_{ij}[1]) \\
(\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r)}_{ij}[2]) \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r+1)}_{ij}[2]) \cap \dots \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(k)}_{ij}[2]) \\
\vdots \\
(\bigcup_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r)}_{ij}[r]) \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r+1)}_{ij}[r]) \cap \dots \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(k)}_{ij}[r]) \\
(\bigcup_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(r+1)}_{ij}[r+1]) \cap \dots \cap (\bigcap_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(k)}_{ij}[r+1]) \\
\vdots \\
(\bigcup_{i=1}^n \bigcap_{j=1}^m \overline{\mathbb{S}}^{(k)}_{ij}[k])
\end{array}\right)$$

is a polymorphism of $\mathbf{C}_k(\mathbf{B})$. ■

Lemma 22. *For any fixed $k \geq 1$, a structure $\mathbf{B}$ has an mkn-ary k-layered m-ABS polymorphism for all m, n if and only if $\mathbf{C}_k(\mathbf{B}) \to \mathbf{B}$.*

Proof. Let $h : \mathbf{C}_k(\mathbf{B}) \to \mathbf{B}$ be a homomorphism. By Lemma 21, the structure $\mathbf{C}_k(\mathbf{B})$ has an mkn-ary k-layered m-ABS polymorphism $f_{n,m}$ for all n, m. By Lemma 17 and Lemma 10, there exists a homomorphism $g : \mathbf{B} \to \mathbf{C}_k(\mathbf{B})$. It is easy to check that the operations $h(f_{n,m}(g(x_1), \dots, g(x_{nmk})))$ are the required polymorphisms of $\mathbf{B}$.

For the other direction, let f be an mkn-ary k-layered m-ABS polymorphism of $\mathbf{B}$ with $m = \rho \cdot |B|$ and $n = \rho \cdot (2^{|B|} - 1)$, where ρ is the maximum of the arities of the relations in $\mathbf{B}$. We can assume without loss of generality that every element of $\mathbf{B}$ participates in some tuple. Define a map $h : C_k(B) \to B$ by the rule $h(\overline{\mathbb{S}}) = f(\min(\overline{\mathbb{S}}))$ where the sequence $\min(\overline{\mathbb{S}})$ is obtained from $\overline{\mathbb{S}}$ so that, for each level l, $\min(\overline{\mathbb{S}})[l]$ contains only those $S \in \overline{\mathbb{S}}[l]$ that are minimal under inclusion in $\overline{\mathbb{S}}[l]$.

By the properties of f, we see that h is well-defined. It remains to show that h defines an homomorphism.

Take an arbitrary (say, r-ary) relation $R \in \tau$ and fix $(\overline{\mathbb{S}}^1, \dots, \overline{\mathbb{S}}^r) \in R^{\mathbf{C_k(B)}}$. We need to show that $(h(\overline{\mathbb{S}}^1), \dots, h(\overline{\mathbb{S}}^r)) \in R^{\mathbf{B}}$. For this, we build a matrix M, as follows.

For every $i = 1, \dots, r$, and $l = 1, \dots, k$, define G^i_l to be B if $l = 1$ and $\bigcap \overline{\mathbb{S}}^i[l-1]$ if $l > 1$. Also, for every $i = 1, \dots, r$ and $l = 1, \dots, k$ and for each set $S \in \overline{\mathbb{S}}^i[l]$, construct a $(m \times r)$-matrix $M^i_S[l]$ whose entries are elements from B and such that

1. each row of $M^i_S[l]$ is an element of $R^{\mathbf{B}}$, and
2. for any $1 \le s \le r$, the set of entries in the s-th column is exactly $\mathrm{pr}_s(R^{\mathbf{B}} \cap (G^1_l \times \cdots \times G^{i-1}_l \times S \times G^{i+1}_l \times \cdots \times G^r_l))$.

That is, the matrix can be seen as a sequence of m tuples $t_1, \ldots, t_m$ (the rows) of $R^{\mathbf{B}}$ such that $\{t_1, \ldots, t_m\} = R^{\mathbf{B}} \cap (G^1_l \times \cdots \times G^{i-1}_l \times S \times G^{i+1}_l \times \cdots \times G^r_l)$. This is easily achieved by placing in the matrix all tuples $t_1, \ldots, t_m$ and repeating some of them if necessary.

Observe that by the definition of $\mathbf{C}_k(\mathbf{B})$, $(\overline{\mathbb{S}}^1[l], \ldots, \overline{\mathbb{S}}^r[l])$ is coherent with $R^{\mathbf{B}}$ in ground $(G^1_l, \ldots, G^r_l)$ and hence, for every s, the set of all entries in the s-th column of $M^i_S[l]$ (a) belongs to $\overline{\mathbb{S}}^s[l]$ and (b) is a subset of G^s_l (here we use both parts of the definition of coherence). Furthermore, if S is minimal in $\overline{\mathbb{S}}^i[l]$ then the set of entries in the i-th column is precisely S.

Now construct for every $l = 1, \ldots, k$ a $(mn \times r)$-matrix $M[l]$ as follows. It is divided into n layers of consecutive m rows, each layer is a matrix $M^i_S[l]$ for some $1 \le i \le r$ and some $S \in \overline{\mathbb{S}}^i[l]$, and each matrix of this form appears as a layer. By the choice of n, this is possible.

Finally form the $(mkn \times r)$-matrix M whose first mn rows are occupied by matrix $M[1]$, next mn rows by matrix $M[2]$ and so on.

For every $s = 1, \ldots, r$, consider $\hat{\mathbb{S}}^s$, which is the sequence $(\hat{\mathbb{S}}^s[1], \ldots, \hat{\mathbb{S}}^s[k])$ such that for any $T \subseteq B$ and $l = 1, \ldots, k$, we have $T \in \hat{\mathbb{S}}^i[l]$ if and only if T is the set of entries of the s-th column of some $M^i_S[l]$. Hence, if we apply f to matrix M column-wise, we obtain the tuple $(f(\hat{\mathbb{S}}^1), \ldots, f(\hat{\mathbb{S}}^r))$ which belongs to $R^{\mathbf{B}}$ because every row of M is in this relation and f is a polymorphism of $\mathbf{B}$.

By the remarks made after the construction of $M^i_S[l]$ we have that, for every $s = 1, \ldots, r$, $\min(\overline{\mathbb{S}}^s) = \min(\hat{\mathbb{S}}^s)$. By the construction of the matrices, for $l > 1$, every $T \in \hat{\mathbb{S}}^s[l]$ is a subset of G^s_l (which coincides with $\bigcap \hat{\mathbb{S}}^s[l-1]$), and it follows that $\hat{\mathbb{S}}^s$ is a nested sequence. It then follows from the absorption property of f that, for every $s = 1, \ldots, r$, we have $f(\hat{\mathbb{S}}^s) = f(\min(\hat{\mathbb{S}}^s))$, and since $\min(\overline{\mathbb{S}}^s) = \min(\hat{\mathbb{S}}^s)$, we have $(h(\overline{\mathbb{S}}^1), \ldots, h(\overline{\mathbb{S}}^r)) = (f(\min(\overline{\mathbb{S}}^1), \ldots, f(\min(\overline{\mathbb{S}}^r)) = (f(\hat{\mathbb{S}}^1), \ldots, f(\hat{\mathbb{S}}^r)) \in R^{\mathbf{B}}$. We conclude that $h : \mathbf{C}_k(\mathbf{B}) \to \mathbf{B}$. ■

Remark 23. *If a structure* $\mathbf{B}$ *has* mkn*-ary* k*-layered* m*-ABS polymorphism for* $m = \rho \cdot |B|$ *and* $n = \rho \cdot (2^{|B|} - 1)$*, where* ρ *is the maximum of the arities of the relations in* $\mathbf{B}$*, then, for any* m*,* $\mathbf{B}$ *has* k*-layered* m*-ABS polymorphisms of all arities divisible by* mk*.*

Proof. (of Theorem 8).

(1) ⇔ (2) follows from Lemma 13.

(2) ⇔ (3) follows from Lemma 18.

(3) ⇔ (4) follows from Lemma 22. ■

4. Case study: posets with constants

In this section we investigate k-cattree duality for special structures obtained from posets. Fix a (finite) poset $\mathbf{Q}$ and let $\mathbf{Q}_c$ denote the structure obtained from $\mathbf{Q}$ by adding all elements of Q as singleton unary relations. Let us denote the signature of $\mathbf{Q}_c$ by τ, the binary relation in τ by R, and the unary relations in τ by $U_q, q \in Q$. Problems of the form $\mathrm{CSP}(\mathbf{Q}_c)$ are closely related to the so-called poset retraction problems (see, e.g., [15]).

For a partial order $\sqsubseteq$ on a set A, and for elements $s, s' \in A$, we say that (s, s') is a *covering pair* if we have $s \sqsubset s'$, and $s \sqsubseteq t \sqsubseteq s'$ implies that $t = s$ or $t = s'$. Note that if we remove such a pair (s, s') from the relation $\sqsubseteq$ then we still have a partial order on A. If $\mathbf{A}$ is a τ-structure such that $R^{\mathbf{A}}$ is a partial order, let $\overline{\mathbf{A}}$ denote the *covering structure* of $\mathbf{A}$, obtained from $\mathbf{A}$ by replacing $R^{\mathbf{A}}$ with the set of all covering pairs in $R^{\mathbf{A}}$. If a τ-structure $\mathbf{A}'$ is obtained from τ-structure $\mathbf{A}$ by adding a new element w to the universe, replacing some pair (u, v) in $R^{\mathbf{A}}$ by two pairs (u, w), (w, v), and leaving the rest of $\mathbf{A}$ unchanged then we say that $\mathbf{A}'$ is obtained from $\mathbf{A}$ by *subdividing an arc*. Moreover, we say that a τ-structure $\mathbf{A}''$ is a *subdivision* of $\mathbf{A}$ if $\mathbf{A}''$ can be obtained from $\mathbf{A}$ by a successive subdividing of arcs. The following claim is easy to verify.

Remark 24. *If* $\mathbf{A} \in \text{co-CSP}(\mathbf{Q}_c)$ *and* $R^{\mathbf{A}}$ *is a partial order then the covering structure* $\overline{\mathbf{A}}$ *and all its subdivisions are also in* $\text{co-CSP}(\mathbf{Q}_c)$.

We will use the notion of a zigzag for a poset [28, 36, 37, 38], not the original definition, but an equivalent characterisation from Proposition 3.1 of [38] (see also Claim 1.1 in [37]). Consider a τ-structure $\mathbf{Z}$ and assume that no element of Z belongs to two different unary relations in $\mathbf{Z}$. The structure $\mathbf{Z}$ is called a $\mathbf{Q}$-*zigzag* if $\mathbf{Z} \in \text{co-CSP}(\mathbf{Q}_c)$, $R^{\mathbf{Z}}$ is a connected partial order and every structure obtained from $\mathbf{Z}$ by removing a covering pair from $R^{\mathbf{Z}}$ belongs to $\mathrm{CSP}(\mathbf{Q}_c)$. It is easy to see that every τ-structure $\mathbf{A}$ such that $R^{\mathbf{A}}$ is a partial order belongs to $\text{co-CSP}(\mathbf{Q}_c)$ if and only if the unary relations in $\mathbf{A}$ are not pairwise disjoint or $\mathbf{A}$ contains a $\mathbf{Q}$-zigzag as a substructure.

Example 25. *(i) Every poset Q has so-called* non-monotone *zigzags which are two-element structures* $\mathbf{N}$ *with universe* $N = \{s, t\}$, $R^{\mathbf{N}} = \{(s, s), (s, t), (t, t)\}$ *being an order on* N, *and such that* $s \in U_u$, $t \in U_v$ *for some* $u \not\sqsubseteq v$ *(in* $\mathbf{Q}$*), and all other unary relations are empty. (All other zigzags are called monotone).*

(ii) Consider the poset $\mathbf{Q}$ *from Fig. 3 (left), same as in Example 9. The monotone zigzags for this poset were described in [37] (see Fig. 6 in [37]). Intuitively, they witness the fact that one cannot homomorphically map an oriented path to* $\mathbf{Q}$ *in such a way that one end of the path goes to* a*, the other to* a'*, and every element of the path is mapped below both* b *and* b'*. More formally, each monotone* $\mathbf{Q}$*-zigzag is a structure of the form* $\mathbf{Z}_{j,t}$ *(where either* $j = 0$ *and* $t \geq 2$ *or* $j = 1$ *and* $t \geq 1$*), which can be described as follows.*

- *Let* $C_0 = \{c_0, c_1, \ldots, c_t\}$, *and* $C_1 = C_0 \cup \{c_{-1}\}$. *The universe of* $\mathbf{Z}_{j,t}$ *is* $Z_{j,t} = C_j \cup \{d_i, e_i \mid 1 \leq i < t \text{ is odd }\}$.

- *For* $j = 0$, *the binary relation of the covering structure* $\overline{\mathbf{Z}}_{j,t}$ *consists of*
  - *all pairs* (c_i, c_{i+1}), *where* $i \leq t-1$ *is even,*
  - *all pairs* (c_{i+2}, c_{i+1}) *where* $i \leq t-2$ *is even, and*
  - *all pairs* (c_i, d_i) *and* (c_i, e_i) *where* $1 \leq i < t$ *is odd.*

  For $j = 1$, *it also contains the pair* (c_0, c_{-1}).
- *The unary relations are as follows:* $U_a = \{c_0\}$ *if* $j = 0$ *and* $U_a = \{c_{-1}\}$ *if* $j = 1$, $U_{a'} = \{c_t\}$, $U_b = \{d_i \mid i < t \text{ is odd }\}$, $U_{b'} = \{e_i \mid i < t \text{ is odd }\}$, *and* $U_c = \emptyset$.

An example of such a structure (with $j = 0$ *and odd* t*) is shown on Fig. 3 (right), where the elements in unary relations are depicted in black and labelled by the corresponding elements of* $\mathbf{Q}$.

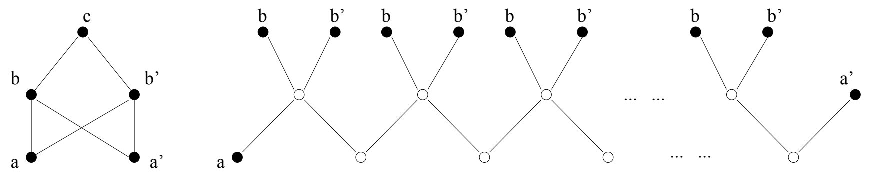

Figure 3: The poset $\mathbf{Q}$ from Example 25(ii) and a $\mathbf{Q}$-zigzag.

For a poset $\mathbf{Q}$, let $\mathcal{O}_{\mathbf{Q}}$ denote the class consisting of (1) all subdivisions of covering structures $\overline{\mathbf{Z}}$ where $\mathbf{Z}$ runs through all $\mathbf{Q}$-zigzags and (2) all one-element τ-structures in which the single element is contained in two different unary relations and all other relations are empty.

Lemma 26. *For a poset* $\mathbf{Q}$, *the class of* τ*-structures* $\mathcal{O}_{\mathbf{Q}}$ *is an obstruction set for* $\mathbf{Q}_c$.

Proof. By Remark 24, we have that $\mathcal{O}_{\mathbf{Q}}$ is a subclass of co-CSP($\mathbf{Q}_c$). Now fix a τ-structure $\mathbf{A} \in$ co-CSP($\mathbf{Q}_c$). If the unary relations in $\mathbf{A}$ are not pairwise disjoint then obviously $\mathbf{A}$ contains a substructure of type (2) above. So assume that this is not the case and show that $\mathbf{A}$ admits a homomorphism from a subdivision of $\overline{\mathbf{Z}}$ for some $\mathbf{Q}$-zigzag $\mathbf{Z}$.

Let θ denote the reflexive transitive closure of $R^{\mathbf{A}}$. The relation θ is a quasi-order, so it is well known that the relation $\epsilon = \theta \cap \theta^{-1}$ is an equivalence relation on A, and θ induces a partial order $\preceq$ on the set A/ϵ of ϵ-classes. Let $\mathbf{A'}$ denote the τ-structure whose universe is A/ϵ and the relations are obtained from those of $\mathbf{A}$ by replacing each element a in each tuple in each relation by its equivalence class a/ϵ. Note that $R^{\mathbf{A'}}$ is exactly $\preceq$, and we have $a/\epsilon \preceq b/\epsilon$ in $\mathbf{A'}$ if and only if $a = b$ or there is a directed path from a to b in $R^{\mathbf{A}}$.

Suppose first that some element x/ϵ is contained in $U_u^{\mathbf{A'}} \cap U_v^{\mathbf{A'}}$ for some distinct $u, v \in Q$. This means that there exist $a, b \in x/\epsilon$ such that $a \in U_u^{\mathbf{A}}$

and $b \in U_v^{\mathbf{A}}$. Since a and b are in the same ϵ-class, they are connected in $R^{\mathbf{A}}$ by directed paths in both directions. Assume without loss of generality that $u \not\leq v$ in $\mathbf{Q}$. Then the substructure of $\mathbf{A}$ consisting of the directed path from a to b (as its binary relation) and a, b in the unary relations U_u, U_v, respectively, is a subdivision of the covering structure of a non-monotone $\mathbf{Q}$-zigzag (see Example 25(i)).

So we can assume from now on that the unary relations in $\mathbf{A}'$ are pairwise disjoint. If h' is a homomorphism from $\mathbf{A}'$ to $\mathbf{Q}_c$ then it is easy to verify that the mapping $h : A \to Q$ defined by $h(a) = h'(a/\epsilon)$ is homomorphism from $\mathbf{A}$ to $\mathbf{Q}_c$. Hence, by our assumption on $\mathbf{A}$, we have $\mathbf{A}' \in$ co-CSP$(\mathbf{Q}_c)$. It follows that $\mathbf{A}'$ contains a $\mathbf{Q}$-zigzag $\mathbf{Z}$. For each element $K \in Z$, fix an element $a_K \in K$ in such a way that $a_K \in U_q^{\mathbf{A}}$ whenever $K \in U_q^{\mathbf{Z}}$. Consider the following subdivision of $\overline{\mathbf{Z}}$: for every arc (K, K') in $R^{\overline{\mathbf{Z}}}$ fix a directed path from a_K to $a_{K'}$ in $R^{\mathbf{A}}$, and subdivide the arc (K, K') as many times as needed to match the length of the path from a_K to $a_{K'}$. Call the obtained structure $\mathbf{T}$. By Remark 24, we have $\mathbf{T} \in$ co-CSP$(\mathbf{Q}_c)$. It remains to show that $\mathbf{T} \to \mathbf{A}$. Define $h : T \to A$ as follows: if $K \in Z$ then let $h(K) = a_K$, and if $x \in T \setminus Z$ is an i-th element on the (unique) path from some $K \in Z$ to some $K' \in Z$ in $R^{\mathbf{T}}$ then let $h(x)$ be the i-th element on the (fixed) path from a_K to $a_{K'}$ in $R^{\mathbf{A}}$. It is easy to see that $h : \mathbf{T} \to \mathbf{A}$. ■

Example 27. *It is easy to see from the description of zigzags for the poset $\mathbf{Q}$ on Fig. 3 (see Example 25) that the class $\mathcal{O}_{\mathbf{Q}}$ obtained as in Lemma 26 consists of 2-cattrees (note that the covering structures of $\mathbf{Q}$-zigzags are all 1-cattrees, but not necessarily such after subdividing arcs). Hence the structure $\mathbf{Q}_c$ has 2-cattree duality. Other similar examples of structures with 2-cattree duality can be obtained by using results from [37]. Note that $\mathbf{Q}_c$ does not have 1-cattree duality because any such structure would have a majority polymorphism (see [10]), while $\mathbf{Q}_c$ has no NU polymorphisms, as shown in [37].*

However, we will now show that, for any poset $\mathbf{Q}$ with an NU polymorphism (of some arity), the structure $\mathbf{Q}_c$ has k-cattree duality for some k. Posets with NU polymorphisms have been characterised in many equivalent ways in [28]. Note that it is a well-known open question in the study of dualities for CSP whether every structure $\mathbf{B}$ with an NU polymorphism has bounded pathwidth duality (see [8, 14]).

Proposition 28. *If $\mathbf{Q}$ is a poset with an NU polymorphism then $\mathbf{Q}_c$ has k-cattree duality for some k.*

Proof. Assume first that $\mathbf{Q}$ is connected. It is shown in [28] that a connected poset has an NU polymorphism if and only if it has finitely many zigzags (note that this does not imply that $\mathbf{Q}_c$ has finite duality, as defined in Subsection 2.4). Moreover, by a result of [29], such a poset $\mathbf{Q}$ has a totally symmetric idempotent polymorphism of arity $|Q|$. Then, by Proposition 1 of [36], $\mathbf{Q}$ has totally symmetric idempotent polymorphisms of all arities, and Corollary 5 of [36] says that in this case every $\mathbf{Q}$-zigzag $\mathbf{Z}'$ admits a surjective homomorphism from a $\mathbf{Q}$-zigzag $\mathbf{Z}$ such that $\overline{\mathbf{Z}}$ is a tree (i.e., the digraph $R^{\overline{\mathbf{Z}}}$ is an oriented tree).

Let $\mathcal{O}'_{\mathbf{Q}}$ denote the class consisting of (i) all subdivisions of $\overline{\mathbf{Z}}$ such that $\mathbf{Z}$ is a $\mathbf{Q}$-zigzag and $\overline{\mathbf{Z}}$ is a tree (ii) all one-element τ-structures in which the single element is contained in two different unary relations (and all other relations are empty). Note that if a τ-structure is a $(k-1)$-cattree then all of its subdivisions are k-cattrees. So, if k is the minimum number such that each covering structure of a $\mathbf{Q}$-zigzag that is a tree is in fact a $(k-1)$-cattree then every structure in $\mathcal{O}'_{\mathbf{Q}}$ is a k-cattree. (This number k exists because $\mathbf{Q}$ has finitely many zigzags). In view of Lemma 26, it remains to show that if $\mathbf{T}'$ is a subdivision of $\overline{\mathbf{Z}'}$ where $\mathbf{Z}'$ is a $\mathbf{Q}$-zigzag, and there is a surjective homomorphism f from a $\mathbf{Q}$-zigzag $\mathbf{Z}$ to $\mathbf{Z}'$, then there is subdivision $\mathbf{T}$ of $\overline{\mathbf{Z}}$ which admits a homomorphism to $\mathbf{T}'$. To construct $\mathbf{T}$, we subdivide $\overline{\mathbf{Z}}$ as follows: each arc (s, s') is subdivided to obtain a path (from s to s') of the same length as the (unique) path from $f(s)$ to $f(s')$ in $\mathbf{T}'$. It is now clear how to build a homomorphism from $\mathbf{T}$ to $\mathbf{T}'$ - it coincides with f on Z and simply matches the paths on $T \setminus Z$, as in the proof of Lemma 26. We conclude that $\mathcal{O}'_{\mathbf{Q}}$ is an obstruction set for $\mathbf{Q}_c$.

Assume now that $\mathbf{Q}$ is not connected. Call a $\mathbf{Q}$-zigzag $\mathbf{Z}$ a connectedness zigzag if it is a path with ends coloured by elements from different connected components of $\mathbf{Q}$; more formally, it is obtained from the structure $\mathbf{Z}_{j,t}$ (see Example 25(ii)) as follows: (i) remove all elements d_i and e_i from the universe and all tuples containing them from the relations; (ii) keep the relations U_a and $U_{a'}$ the same, but this time a and a' is an arbitrary pair of elements from different connected components of $\mathbf{Q}$; (iii) make all other unary relations empty. It is easy to verify that if a $\mathbf{Q}$-zigzag $\mathbf{Z}$ has non-empty unary relations corresponding to elements from different connected components of $\mathbf{Q}$ then it must be a connectedness zigzag. On the other hand, if all non-empty unary relations in $\mathbf{Z}$ correspond to elements from the same connected component of $\mathbf{Q}$ then $\mathbf{Z}$ must be a zigzag for that connected component (technically, expanded with empty unary relations corresponding to the elements from other connected components).

Note that, for each connectedness $\mathbf{Q}$-zigzag $\mathbf{Z}$, the covering structure $\overline{\mathbf{Z}}$ and all its subdivisions are 1-cattrees, and hence k-cattrees for all k. It is easy to see that the NU polymorphism of $\mathbf{Q}$ preserves each connected component of $\mathbf{Q}$, so each connected component of $\mathbf{Q}$ has an NU polymorphism. It is clear now how to find the required obstruction set for $\mathbf{Q}_c$: take all subdivisions of the covering structures of connectedness zigzags and also all structures from $\mathcal{O}'_{\mathbf{Q}'}$ (suitably expanded with empty unary relations) where $\mathbf{Q}'$ runs through all connected components of $\mathbf{Q}$. ∎

5. Cattrees and hypergraph searching games

In this section, we investigate searching games on trees. In particular, we characterise k-cattrees in terms of such games and also prove Theorem 6. Most of our results in this section can be of independent interest in graph theory.

Graph and hypergraph searching games have recently attracted considerable attention (see [19, 20]). Such a game usually involves a fugitive and a set of searchers. The main types of searchers are cops, who can only occupy vertices, and marshals, who can only occupy (hyper)edges. A cop controls only the vertex it occupies, while a marshal controls all vertices in the (hyper)edge it occupies. In each round of the game, searchers can change their positions. The fugitive occupies a vertex, he sees where the searchers are landing and can move to a different vertex at infinite speed through paths in the (hyper)graph as long as he does not cross nodes controlled by a searcher. The goal of the cops/marshals, who act in a coordinated manner, is to land on a vertex/(hyper)edge occupied by the fugitive whereas the goal of the fugitive is to elude capture. A large number of variants of searching games were studied, with additional features of fugitive or searchers, and the minimum number of cops/marshals necessary to capture the fugitive on a (hyper)graph in a given variant of the game is often closely related with important structural parameters of (hyper)graphs.

Some of the most important optional features of search games are the invisibility of the fugitive (i.e., searchers have no information on his current position), the connectedness of the "cleared" space (of vertices where the fugitive cannot be at a given stage), and the monotonicity (i.e., the fugitive must be prevented from ever entering any cleared vertex) [3, 19]. For example, the minimum number of marshals in the monotone game characterises hypertree-width of a hypergraph [20], and the requirement of monotonicity is necessary for this [1].

In this paper we consider the searching game on hypergraphs in which marshals can arbitrarily change their positions in each round and the fugitive is invisible. This variant of the game has been intensively studied for graphs, but, to the best of our knowledge, it has not been introduced for hypergraphs.

In order to formalize a winning strategy in a game, we need to introduce some definitions. The Gaifman graph of a hypergraph $H = (V(H), E(H))$, denoted by $G(H)$, is the graph G with $V(G) = V(H)$ and that has an edge (u, v) if $u \neq v$ and there exists some hyperedge e with $\{u, v\} \subseteq e$. Let H be a hypergraph, $X \subseteq V(H)$, and $u, v \in V(H) \setminus X$. We say that u and v are *connected under blockage* X if there is a path connecting u and v in $G(H)$ that does not cross any element of X.

In this section, we will consider only hypergraphs and hence if we refer to, say H, as a tree, we mean that H is a hypergraph that is also a tree. Also, we call a subhypergraph of H which is a tree a subtree of H.

Let $H = (V(H), E(H))$ be a hypergraph and $k \geq 1$ an integer. We denote the invisible fugitive and k marshals game on H by $\mathrm{InvMar}(H, k)$. A position in this game is a pair (W, X) where W is a subset of $V(H)$ which indicates the set of *contaminated* nodes, i.e., the set of nodes that can possibly harbor the fugitive, and X is a subset of at most k hyperedges of $E(H)$ which indicates the positions of the marshals. The initial configuration of the game is $(V(H), \emptyset)$.

Let (W_i, X_i) be the position in the i-th round of the game. A move in the game consists in changing the location of the marshals to a new set X_{i+1} of hyperedges. The set of contaminated nodes after the move is $W_{i+1} = W'_{i+1} \setminus$

$\bigcup X_{i+1}$ where W'_{i+1} is the set of nodes accessible from W_i under blockage $\bigcup X_i \cap \bigcup X_{i+1}$. That is, when changing the location of the marshals from X_i to X_{i+1} the nodes in $\bigcup X_i \cap \bigcup X_{i+1}$ are blocked, i.e, the fugitive cannot run through them. At a given position, a node is *cleared* if it is not contaminated. The goal of the game is to clear simultaneously all nodes, i.e, to reach a configuration of the form $(\emptyset, X)$. Formally, we define a *(winning) strategy* for the game on H as a sequence $(W_i, X_i), i = 1, \ldots, m$ of positions where $W_1 = V(H)$, $X_1 = \emptyset$, $W_m = \emptyset$ and for every $i = 2, \ldots, m$, $|X_i| \leq k$ and $W_i = W'_i \setminus \bigcup X_i$ where W'_i is the set of nodes accessible from W_{i-1} under blockage $\bigcup X_{i-1} \cap \bigcup X_i$.

In the *monotone* version of the game, it is required that the set of cleared nodes increases monotonically, that is, if (W, X) and (W', X') are two successive positions of the game then $W' \subseteq W$. We say that a hyperedge e of H is cleared at position (W, X) if $W \cap e = \emptyset$. In the *connected* version of the game it is required that, at every round, the subhypergraph of H constituted by the cleared hyperedges is connected. We shall call strategies for these versions monotone and connected, respectively (and monotone connected when both restrictions are applied simultaneously).

For a hypergraph H, let $im(H)$ denote the minimum k such that the marshals have a winning strategy in $\mathrm{InvMar}(H, k)$. The corresponding numbers for the monotone, connected, and connected monotone games are denoted by $mim(H)$, $cim(H)$, and $cmim(H)$, respectively.

Let H be a tree, let $S = (W_i, X_i), i = 1, \ldots, m$ be a connected strategy for H, and let e be a hyperedge of H. We say that S is a connected strategy for (H, e) if, additionally, the hyperedge e is cleared in all rounds of the game (except the first). Define $cim(H, e)$ to be the minimum number of marshals necessary to obtain such a strategy. The number $cmim(H, e)$ is naturally defined in a similar way.

Example 29. *Let H be a caterpillar and let $e_1, e_2, \ldots, e_n$ be an ordering of its hyperedges such that two consecutive hyperedges share exactly one node. A unique marshal can clear all nodes of H by just following the previous sequence. Furthermore, this strategy is monotone and connected. Hence we have $mim(H) = cim(H) = cim(H, e_1) = cmim(H) = cmim(H, e_1) = 1$.*

The invisible fugitive and cops game and its variants are defined analogously with the only difference that cops occupy vertices instead of hyperedges. Formally, a position of the $\mathrm{InvCop}(H, k)$ game is a pair (W, Y) where $W, Y \subseteq V(H)$, and $|Y| \leq k$. A winning strategy for this game is a sequence $(W_i, Y_i), i = 1, \ldots, m$ of positions where $W_1 = V(H)$, $X_1 = \emptyset$, $W_m = \emptyset$ and for every $i = 2, \ldots, m$, $|Y_i| \leq k$ and $W_i = W'_i \setminus Y_i$ where W'_i is the set of nodes accessible from W_{i-1} under blockage $Y_{i-1} \cap Y_i$. The numbers $ic(H)$ and $mic(H)$ are defined similarly to $im(H)$ and $mim(H)$. Observe that there is no difference between playing the cop game on H or its Gaifman graph. Hence by the results of [22, 23, 33] we have:

Proposition 30. *Let H be a hypergraph and let G be its Gaifman graph. Then $mic(H) = mic(G) = \mathrm{pw(G)} + 1$.*

5.1. Games on trees

In this subsection, we investigate the parameters im, mim, cim and $cmim$ for trees. In particular, we will show that the last two parameters essentially correspond to the cattree parameter of a tree.

Kirousis and Papadimitriou [23], using [26], showed that if G is a graph then $mic(G) = ic(G)$, which implies that $mic(H) = ic(H)$ for any arbitrary hypergraph H. Also, recontamination does not help (i.e., dropping the requirement of monotonicity never reduces the necessary number of cops) in the variant of the fugitive and cops game in which the fugitive is *visible* [35]. However, it is not clear whether the same holds when cops are replaced by marshals; for example, recontamination can help in the *visible* variant of the game with marshals [1]. In this section we show, among other things, that $im(H) = mim(H)$ and $cim(H) = cmim(H)$ if H is a (hypergraph) tree. The main ideas of the proofs are inspired by the work on graphs [17, 32].

Let H be a tree an let $\mathcal{T}_H$ be the set of all trees T such that there is a subtree E of H with T being cut off by E (for the definition of a tree being cut off, see Subsection 3.1). For every $T \in \mathcal{T}_H$, let E be any subtree of H such that T is cut off by E and define e_T to be the hyperedge connecting E to T and v_T to be the only element in $V(T) \cap V(E)$. Notice that e_T and v_T do not depend on the choice of E. Let $e_T^* = e_T \setminus \{v_T\}$. Define T^* to be the tree obtained from T by removing v_T from $V(T)$ and replacing e_T by e_T^*.

Call a node of H a *leaf* if it is contained in a single hyperedge, and a *non-leaf* otherwise. For a subtree E of H and a set $\mathcal{X} \subseteq \mathcal{T}_H$, we say that a node $v \in V(E)$ is connecting E with $\mathcal{X}$ if there is a tree $T \in \mathcal{X}$ cut off by E such that $v = v_T$.

Before proving the main results of this subsection, we need several technical lemmas.

Lemma 31. *Let H be a tree and let $\mathcal{X} \subseteq \mathcal{T}_H$ be such that every subtree E of H has at most two leaves connecting it to $\mathcal{X}$. Then there is a subtree of H which is a caterpillar and has no nodes connecting it to $\mathcal{X}$.*

Proof. Assume first that there exists a caterpillar E that has two leaves connecting it to $\mathcal{X}$. Then there exists one such that the connecting leaves appear in opposite extremes, that is, such that its hyperedges can be ordered $e_1, \dots, e_n$ ($n \geq 1$) in such a way that two consecutive hyperedges share one element, and one of the connecting leaves belongs to e_1 and the other to e_n. One such caterpillar is obtained by taking the sequence of hyperedges in the path in E that connects the two leaves. Now let E' be an inclusion-wise maximal caterpillar with connecting leaves appearing in opposite extremes, and let E'' be the caterpillar obtained by adding to E' all hyperedges (of H) that contain one or the other connecting leaf of E'. Let us show that E'' is a required caterpillar. It follows that every hyperedge e of H that contains a non-leaf of E'' belongs to E''. Indeed, if the non-leaf is a connecting leaf of E', this follows from the construction of E'', whereas in any other case it follows by the maximality of E'. Hence, if E'' has a connecting node v then it certainly needs to be a leaf of E''. It follows that $v \notin V(E')$ since otherwise E' would have three different leaves connecting it to $\mathcal{X}$. But if v belongs to an extreme e of E'', the caterpillar obtained by adding e to E' would contradict the maximality of E'. Thus, E'' is the required caterpillar.

Now assume that every caterpillar in H contains at most one leaf connecting it to $\mathcal{X}$. Again, if there exists one that contains such a leaf then we can assume that this leaf appears in one of the extremes. Pick again a maximal E' with this property and construct E'' by adding to E' all hyperedges containing the leaf. It follows in a similar way to the previous case that E'' has no leaves connecting it to $\mathcal{X}$. ■

Lemma 32. *Let H be a tree, let $S = (W_i, X_i), i = 1, \dots, m$ be a strategy for* $\mathrm{InvMar}(H, k)$*, and let $T \in \mathcal{T}_H$.*

1. *Assume that $im(T^*) = k$ and l is a round $(2 \le l \le m)$ such that $V(T^*) \subseteq W_{l-1}$. Then there exists some round $l \le i \le m$, such that X_i contains k different hyperedges from T.*
2. *Assume that $cim(T^*, e_T^*) \ge k$, S is connected and that there is some round $2 \le l \le m$ such that $V(T^*) \subseteq W_{l-1}$ and $e_T \cap W_i = \emptyset$ for all $l \le i \le m$. Then there exists some round $l \le i \le m$, such that X_i contains k different hyperedges from T.*

Proof. Define S^* to be the sequence $(W_i^*, X_i^*), i = l, \dots, m$ where $X_i^* = \{e \cap V(T^*) \mid e \in X_i\}$ and W_i^* is the set of nodes of T^* contaminated in round $i, i = l, \dots, m$, according to the rules of the game.

(1) It is enough to show that S^* is a strategy for T^*. If B is the blockage from round $i-1$ to round i according to S then the blockage from round $i-1$ to i according to S^* is $B \cap V(T^*)$. It follows that $W_i^* \subseteq W_i$ for all $i = l, \dots, m$, and so $W_m^* = \emptyset$.

(2) It is enough to show that S^* is a connected strategy for (T^*, e_T^*). It follows from the assumptions and from the fact that $W_i^* \subseteq W_i$ that e_T^* is cleared in every round. It remains to show that S^* is connected.

Towards a contradiction, assume that, at some round i, the set of cleared hyperedges by S^* is not connected. Since e_T^* is cleared and we can choose a path $v_0 e_1 v_1 e_2 \dots e_n v_n$ such that $v_0 \in e_T^*$ and v_n is cleared at round i while v_{k-1} is not. We shall show that this implies that strategy S is not connected at round i, contradicting the assumptions. In particular we shall show that at round i, according to S, v_0 and v_n are cleared whereas v_{n-1} is not. The fact that v_0 is cleared follows from the fact that e_T is cleared at round i whereas the fact that v_{n-1} is not cleared follows from $W_i^* \subseteq W_i$. In order to show that v_n is cleared, according to S, at round i we notice that v_n shares an hyperedge, namely e_n, with a node which is contaminated according to S^*, namely v_{n-1}. Hence the only reason why v_n can possibly be cleared (according to S^*) at round i is because a marshal occupies some edge e containing v_n. By the definition of S^* this implies that, in S, there is a marshall seating in some hyperedge e' with $e' \cap V(T^*) = e$ (in fact, since v_n cannot be in e_T^* we have that $e = e'$, but this does not play any role). Since $v_n \in e'$ we conclude that v_n is cleared according to S at round i. ■

Lemma 33. *Let H be a tree, let E be a subtree of H, and let T_1, T_2, T_3 be cut off by E, such that $v_{T_1}, v_{T_2}, v_{T_3}$ are leaves of E. Then:*

1. *If $im(H) \leq k$ and $im(T_j^*) = k$ for $j = 1, 2, 3$, then $|\{v_{T_1}, v_{T_2}, v_{T_3}\}| \leq 2$.*
2. *If $cim(H) \leq k$ and $cim(T_j^*, e_{T_j}^*) \geq k$ for $j = 1, 2, 3$, then $|\{v_{T_1}, v_{T_2}, v_{T_3}\}| \leq 2$.*
3. *If $cim(H, e) \leq k$ for some hyperedge e of E and $cim(T_j^*, e_{T_j}^*) \geq k$ for $j = 1, 2$, then $v_{T_1} = v_{T_2}$.*

Proof. All proofs are by contradiction:

(1) Assume that $v_{T_1}, v_{T_2}, v_{T_3}$ are all different. Let $(W_i, X_i), i = 1, \dots, m$ be a strategy for InvMar(H, k). There exists a round in which all the vertices of (at least) two of the three trees are completely cleared. Choose $n \in \{1, \dots, m\}$ to be minimal with such property and assume without loss of generality that the trees cleared are T_1 and T_2. For $j = 1, 2$ let l_j be minimal with the property that $V(T_j)$ contains cleared nodes in every round $l_j \leq i \leq n$.

Assume without loss of generality that $l_1 \leq l_2$. It follows from the fact that $im(T_2^*) = k$ and Lemma 32(1) that there exists some round $l_2 \leq i \leq n$ in which all the marshals sit on hyperedges of T_2. By the minimality of n there is, at round $i-1$, some contaminated node $v \in V(T_1) \cup V(T_3)$. Since E is connected and v_{T_2} is a leaf of E which is different from both v_{T_1} and v_{T_3}, it follows that, during the transition from round $i-1$ to round i, there is a non-blocked path between v_{T_1} and v_{T_3}, and hence a non-blocked path connecting v with any node in $V(T_1)$. Hence every node in $V(T_1)$ is contaminated at round i contradicting the definition of l_1.

(2) Assume that $v_{T_1}, v_{T_2}, v_{T_3}$ are all different and the strategy $(W_i, X_i), i = 1, \dots, m$ is connected. Hence there exists a round in which all the vertices of (at least) two of the three trees are completely clear. Choose n to be minimal with such property and assume without loss of generality that T_1 and T_2 are cleared. For $j = 1, 2$, let l_j be minimal with the property that e_{T_j} is cleared in every round $l_j \leq i \leq n$.

Assume first that $l_1 < l_2$. Since e_{T_1} is cleared in round $l_2 - 1$ and e_{T_2} is not, it follows by connectivity of the strategy that none of the hyperedges of T_2 is cleared at round $l_2 - 1$. From the fact that $cim(T_2^*, e_{T_2}^*) \geq k$ and Lemma 32(2) we get that there exist some round $l_2 \leq i \leq n$ in which all marshals sit on hyperedges of T_2. Since n is minimal, there is, in round $i-1$, some contaminated node $v \in V(T_1) \cup V(T_3)$. Since E is connected and v_{T_2} is a leaf of E which is different from both v_{T_1} and v_{T_3}, it follows that, during the transition from round $i-1$ to round i, there is a non-blocked path connecting v with any node in e_{T_1}. Hence e_{T_1} is not cleared in round i contradicting the definition of l_1. The case $l_2 < l_1$ is symmetric.

The case $l_1 = l_2$ follows very similarly. Neither of e_{T_1}, e_{T_2} is cleared in round $l_1 - 1$. Therefore, again by connectivity, there is at most one $j \in \{1, 2\}$ such that T_j has a hyperedge cleared in round $l_1 - 1$. Assume without loss of generality that $j = 1$. It follows from Lemma 32(2) that there exists some round $l_2 \leq i \leq n$ in which all the marshals sit on hyperedges of T_2. From here the proof proceeds as in the previous case.

(3) Let $(W_i, X_i), i = 1, \ldots, m$ be a connected strategy for (H, e), let n be the first round in which at least one of the trees, say T_1, is entirely cleared. Define l_1 to be minimal with the property that e_{T_1} is cleared in every round $l_1 \leq i \leq n$. Since e is cleared at round $l_1 - 1$ it follows by connectivity that none of the hyperedges of T_1 is cleared at round $l_1 - 1$ and, henceforth, there is some round $l_1 \leq i \leq n$ in which all marshals sit on hyperedges of T_1. If $v_{T_1} \neq v_{T_2}$ then, by the minimality of n, T_2 contains, in round i, some contaminated node v which would allow to recontaminate, via v_{T_2} and E, any node in e, a contradiction. ■

Let $H = (V, E)$ and $H' = (V, E')$ be trees with the same set of nodes. We say that H is *obtained by decorating* H' if $E = E'$ or E is obtained by adding singleton hyperedges to E'. A hypergraph obtained by decorating a caterpillar or a k-cattree will be called a *decorated caterpillar* or *decorated k-cattree*, respectively. If e is a hyperedge of H' such that (H', e) is a k-cattree then we also say that (H, e) is a decorated k-cattree. The following fact can be easily derived from Definition 5 by induction on k.

Remark 34. *For any k, every decorated k-cattree is a $(k+1)$-cattree.*

Notice that, strictly speaking, one cannot capture k-cattrees by means of games. Indeed, it is not difficult to see that if H is obtained by decorating a caterpillar H' then H is not necessarily a caterpillar but $im(H) = 1$ as the same clearing sequence for H' will also work in H. In the rest of this section we show that, besides this technical point, the connected game with invisible fugitive and k marshals captures k-cattrees.

Lemma 35. *Let H be a tree, and e its hyperedge.*

1. *If $im(H) = 1$ then H is a decorated caterpillar.*
2. *If $cim(H, e) = 1$ then (H, e) is a decorated caterpillar.*

Proof. (1) Define $\mathcal{X}$ to be the set consisting of all $T \in \mathcal{T}_H$ such that $im(T^*) = 1$ (clearly, it is not possible that $im(T^*) > 1$). By Lemma 33(1), set $\mathcal{X}$ satisfies the assumptions of Lemma 31. It now follows from Lemma 31 that there exists some caterpillar E such that for every T cut off by E, $im(T^*) = 0$. Consequently every tree T in $\mathcal{T}_E$ has only one (singleton) hyperedge and, henceforth, H is obtained by decorating E.

(2) We need to show that H is obtained by decorating a caterpillar E that has e as an extreme. Pick any sequence $e_1, \ldots, e_n$ of hyperedges of H with $e_1 = e$, where $e_2, \ldots, e_n$ are not singletons and such that every pair of consecutive

hyperedges share exactly one node. Assume that n is maximal and let us denote by E the subhypergraph of H constituted by the hyperedges in the ordering. The claim follows if we can prove that H is obtained by decorating E. Assume, for a contradiction, that this is not the case. Then there exists a tree T cut off by E which has a non-singleton hyperedge. It follows that v_T is a leaf of E, since otherwise we could add e_T to E obtaining a caterpillar with one extra hyperedge. Let $i \in \{1, \ldots, n\}$ be such that e_i contains v_T and let $j \geq i$ be maximal with the property that $e_j \cap e_{i-1} \neq \emptyset$. If $j = n$ then the sequence $e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_n, e_i, e_T$ contradicts the maximality of n. Otherwise let E' be the subhyperegraph constituted by $e_1, \ldots, e_j$. Clearly T is cut off by E' as well. Also there is a tree T' cut off by E' containing e_{j+1}. By the maximality of j, $v_{T'}$ is a leaf of E'. Furthermore we have that $v_T \neq v_{T'}$ because v_T is a leaf of E whereas $v_{T'}$ is not. This contradicts Lemma 33(3) and we are done. ■

Together with Example 29, Lemma 35 implies the following.

Corollary 36. *For any tree H, H is a decorated caterpillar if and only if any (equivalently, each) of parameters $im(H)$, $mim(H)$, $cim(H)$, $cmim(H)$ is equal to 1.*

We are now in a situation to show that $im(H) = mim(H)$ for every tree H. This is done via a structural characterization of the trees with $im(H) \leq k$. This result is not used in our study of the CSP in this paper, but it will complete our treatment of the invisible fugitive and k marshals game on trees.

Recall that, if T is a tree cut off by some subtree in H, then T^* is defined to be the tree obtained from T by removing v_T from $V(T)$ and replacing e_T by $e^*_T = e_T \setminus \{v_T\}$.

Definition 37. *Let H be a tree and let $k \geq 1$. We say that H is a* weak k-cattree *if:*

1. *H is a caterpillar, or*
2. *$k > 1$ and there is subtree E of H which is a caterpillar, and, for every subtree T cut off by E, T^* is a weak $(k-1)$-cattree.*

The difference between the above definition and the definition of a k-cattree is that here the tree T does not have to be attached to E by an extreme and that the node connecting T to E is removed before computing the parameter of the tree cut off. It can be seen directly that every k-cattree is also a weak k-cattree, and this also trivially follows from Theorems 38 and 40. It will follow from Proposition 42 that every weak k-cattree is a $2k$-cattree.

We say that a decorated weak k-cattree is a tree obtained by decorating a weak k-cattree.

Theorem 38. *Let H be a tree and $k \geq 1$. The following are equivalent:*

1. *$im(H) \leq k$*

2. *H is a decorated weak k-cattree.*

3. *$mim(H) \leq k$*

Proof. ($1 \Rightarrow 2$) The proof is by induction on k. The case $k = 1$ is given in Lemma 35. Now let $k > 1$ and assume that $im(H) = k$. Define $\mathcal{X}$ to be the set consisting of all $T \in \mathcal{T}_H$ such that $im(T^*) = k$. By Lemma 33(1), this set $\mathcal{X}$ satisfies the assumptions of Lemma 31. It now follows from Lemma 31 that there exists some caterpillar E such that, for every T cut off by E, we have $im(T^*) \leq k - 1$. By induction hypothesis, every such T^* is a decorated weak $(k-1)$-cattree, which implies that T is a decorated weak $(k-1)$-cattree as well. Define T' to be the weak $(k-1)$-cattree from which T is obtained by decorating. Finally, H is obtained by decorating the weak k-tree obtained from E and T', where T runs through $\mathcal{T}_E$, according to Definition 37.

($2 \Rightarrow 3$) The implication will follow if we can prove it for the case when H is a weak k-cattree. The proof is by induction on k. The case $k = 1$ is given by Example 29. Now let $k > 1$. Let E be the caterpillar given by the definition of a weak k-cattree and let $e_1, \ldots, e_n$ be an ordering of its hyperedges such that two consecutive hyperedges share one node. Place initially one marshal in e_1. While keeping the marshal on e_1, use the remaining $k - 1$ marshals to clear all nodes that belong to every tree T cut off by E such that $v_T \in e_1$. This is done processing all such trees one by one. If T is such a tree, we know that there exists a monotone strategy $(W_i, X_i), i = 1, \ldots, m$ that clears T^* using only $k-1$ marshals. By using such sequence (with added e_1 and with e^*_T replaced by e_T throughout) for every T, we can clear all such trees T. Once this is done, we move the marshal from e_1 to e_2 and continue in the same fashion to obtain a monotone strategy for the whole H.

The implication ($3 \Rightarrow 1$) is trivial. ■

The proof for the connected variant is slightly more involved. We first need the following intermediate result.

Proposition 39. *Let H be a tree, let e be a hyperedge of H, and $k \geq 1$. Then the following are equivalent:*

1. *$cim(H, e) \leq k$.*

2. *(H, e) is a decorated k-cattree.*

3. *$cmim(H, e) \leq k$.*

Proof. Implication ($3 \Rightarrow 1$) is trivial.

($1 \Rightarrow 2$). The proof goes by induction on k. The case $k = 1$ follows from Lemma 35.

For induction step, we claim that there exists some caterpillar E with the extreme e such that for every tree T cut off by E, $cim(T^*, e^*_T) \leq k-1$. Construct a strictly increasing sequence of subtrees of H in the following way. Set $E_1 = e$.

For $i \geq 1$, while there is a tree T_i cut off by E_i such that $cim(T_i^*, e_{T_i}^*) \geq k$, choose such tree arbitrarily and define E_{i+1} to be the hypergraph obtained by adding to E_i all hyperedges in H that contain v_{T_i}. Assume that $E_1, \ldots, E_n$ is a longest such sequence. If we can show that $v_{T_i} \notin V(E_{i-1})$ for all $i \geq 2$, it will follow that each E_i is a caterpillar and that $E = E_n$ has the desired property. Assume, for contradiction, that $v_{T_i} \in V(E_{i-1})$ for some $i \geq 2$. Choose the smallest i with this property and then the smallest $j < i$ such that $v_{T_i} \in V(E_j)$. Note that, by the choice of i and j, we have that v_{T_i} and v_{T_j} must be leaves of E_j. Then Lemma 33(3) implies that $v_{T_i} = v_{T_j}$, and hence, by construction, E_i contains all hyperedges of H containing v_{T_i}. But this contradicts the choice of T_i as being cut off by E_i, so we do have our caterpillar E.

It follows by induction hypothesis that, for every tree T cut off by E, (T^*, e_T^*) is a decorated $(k-1)$-cattree. This implies that (T, e_T) is a decorated $(k-1)$-cattree as well. Let (T', e_T) be the $(k-1)$-cattree from which (T, e_T) is obtained by decorating. Hence, (H, e) can be obtained by decorating the k-cattree obtained from E and $(T', e_T), T \in \mathcal{T}_E$ according to Definition 5.

$(2 \Rightarrow 3)$. The proof of this part is very similar to the direction $(2 \Rightarrow 3)$ of Theorem 38. It suffices to show it in the case that (H, e) is a k-cattree. We shall show, by induction on k, that there is a hyperedge e of H such that $cmim(H, e) \leq k$. The case $k = 1$ follows from Example 29. Now let E be the caterpillar containing e as an extreme that is guaranteed to exist by the definition of a k-cattree. Let $e = e_1, \ldots, e_n$ be an ordering of its hyperedges such that two consecutive hyperedges share one node. There is one marshal that performs exactly the sequence $e_1, \ldots, e_n$. Before leaving e_i, the remaining $k-1$ marshals are used to clear all subtrees cut off by E that share some element with e_i, one by one. If T is any such subtree and e_T is the edge of T connecting T to e_i then, by induction hypothesis, T can be cleared with $k-1$ marshals in such a connected monotone way so that e_T is cleared in every round. This guarantees that the whole strategy for H is connected and e is always cleared. ∎

Theorem 40. *Let H be a tree and $k \geq 1$. Then the following are equivalent:*

1. *$cim(H) \leq k$.*
2. *H is a decorated k-cattree.*
3. *$cmim(H) \leq k$.*

Proof. Implication $(3 \Rightarrow 1)$ is trivial, while $(2 \Rightarrow 3)$ follows from Proposition 39 because, it follows easily from the definition that H is a decorated k-cattree if (H, e) is such for some hyperedge e.

$(1 \Rightarrow 2)$. The case $k = 1$ follows from Example 29. Assume $k > 1$. Define $\mathcal{X}$ to be the set consisting of all $T \in \mathcal{T}_H$ such that $cim(T^*, e_T^*) \geq k$. By Lemma 33(2), this set $\mathcal{X}$ satisfies the assumptions of Lemma 31. It now follows from Lemma 31 that there exists a caterpillar E such that, for every tree T cut off by E, $cim(T^*, e^*_T) \leq k-1$. It follows by Proposition 39 that (T^*, e^*_T) is a decorated $(k-1)$-cattree, which implies that (T, e_T) is a decorated $(k-1)$-cattree as well. Let (T', e_T) be the $(k-1)$-cattree from which (T, e_T) is obtained by decorating. Hence (H, e) is a decorated k-cattree (for any extreme e of E) as it can be obtained by decorating the k-cattree obtained from E and $(T', e_T), T \in \mathcal{T}_E$ according to Definition 5. ■

A close inspection to the proof of Theorem 38 reveals that every decorated weak k-cattree has a monotone strategy in which only one marshal moves in every round. Also, the proof of Proposition 39 shows that every decorated k-cattree has a monotone connected strategy in which only one marshal moves at a time. This implies that, for trees, there is not any gain in allowing several marshals to move at once under any of the variants of the invisible game with marshals considered in this paper. For hypergraphs that are not trees this is not longer true. For example, it is easy to see that the hypergraph H with $V(H) = \{1, \ldots, 8\}$ and $E(H) = \{\{1,2,3\}, \{4,5,6\}, \{1,4,7\}, \{2,5,8\}, \{3,6\}, \{7,8\}\}$ has $im(H) = 2$ (the two marshals occupy the first pair of hyperedges first, and then simultaneously move to the second pair), but there is no strategy for H with 2 marshals when we require that only one marshal moves at once (since every transition between rounds would leave at least one of the nodes 1,2,4,5 unblocked). This is again one aspect in which marshals and cops differ substantially, as in almost every conceivable variant with cops, and certainly the ones considered in this paper, there is not any gain in moving several cops at once.

5.2. Pathwidth and cattrees

From Proposition 30 and Theorem 40, we know that the *mic* and *cmim* parameters of a hypergraph are connected with pathwidth (of Gaifman graph) and cattree parameters, respectively. The goal of this subsection is to link them between themselves (which we achieve in Corollary 43), thus proving Theorem 6.

Proposition 41. *Let H be a hypergraph and let r be the maximum among the cardinality of its hyperedges. Then $mim(H) \leq mic(H) \leq r \cdot mim(H)$.*

Proof. ($mim(H) \leq mic(H)$) Let $k \geq 1$ and let $(W_i, Y_i), 1 \leq i \leq m$ be a winning strategy in the monotone game with k cops for H. Every hyperedge e must be contained entirely in some set Y_i, $1 \leq i \leq m$ because otherwise the fugitive could always hide in e. Let the *index* of e be the minimal i such that $e \subseteq Y_i$. For any node v, arbitrarily pick one hyperedge e_v with minimal index among those containing v, and say that the index of v is that of the edge e_v. Observe that we can assume that if v is a node and i is its index then v does not appear in any Y_j with $j < i$ as otherwise we could for every $j < i$ remove v from Y_j and add it to W_j obtaining again a winning strategy.

It is not difficult to verify that $(W'_i, X_i), i = 1, \ldots, m$ with $X_i = \{e_v \mid v \in Y_i\}$ (and sets W'_i determined by the rules of the game) is a strategy in the monotone game with k marshals. Indeed, by the construction of the sequence $(W'_i, X_i), i = 1, \ldots, m$ we have $Y_i \subseteq \bigcup X_i$ for every $i = 1, \ldots, m$ and hence this sequence will clear all nodes. It remains to show that no node can be recontaminated once it has been cleared. To this end, observe that when a marshal is first placed on a hyperedge e, say in round i, then by the choice of e, we have that $e \subseteq Y_i$. Then monotonicity of the strategy for the fugitive and cops game guarantees that no node in e will be contaminated again.

($mic(H) \leq r \cdot mim(H)$) Let $k \geq 1$ and let $(W_i, X_i), i = 1, \dots, m$ be a strategy in the monotone game with k marshals for H. Then, the sequence $(W_i, Y_i), 1 \leq i \leq m$ with $Y_i = \bigcup X_i$ (and the same sets W_i) for all i is a strategy for the monotone game with $r \cdot k$ cops. ■

Proposition 42. *Let H be a tree, let e be a hyperedge of H, and let $k \geq 1$. If $im(H) \leq k$ then $cmim(H, e) \leq 2k$.*

Proof. If $im(H) \leq k$ then, by Theorem 38, H is a decorated weak k-cattree. Note that it is enough to prove the proposition assuming that H is a weak k-cattree, since the same strategy would always work in the decorated case. We prove it by induction on k.

The base case $k = 1$ (that is, when H is a caterpillar) follows easily. If e is an extreme of H then $cmim(H, e) = 1$ and we are done. Otherwise, let $e_1, \dots, e_n$ be an ordering of the hyperedges of H such that two consecutive nodes share exactly one node and let $1 \leq i \leq n$ such that $e = e_i$. Place one marshal in e_i and while keeping this marshal in e_i use the other marshal to clean one "half" of the caterpillar by playing it along the sequence $e_{i+1}, \dots, e_n$. At this point clear the other half by moving the marshal that was originally kept in e_i to $e_{i-1}, e_{i-1}, \dots, e_1$ successively.

Assume $k > 1$ and fix an arbitrary hyperedge e of H. Let E be the caterpillar given by Definition 37. If T is any tree cut off by E then T^* is a weak $(k-1)$-cattree, which implies that T is a weak $(k-1)$-cattree as well. Hence, by induction hypothesis, $cmim(T, e') \leq 2k - 2$ for any hyperedge e' of T.

Let $e_1, e_2, \dots, e_n$ be an ordering of the hyperedges of E such that two consecutive hyperedges share one node.

Assume first that e is in E, say $e = e_i$ for some i. Place a marshal in e_i and for every subtree T cut off by E such that $v_T \in e_i$, clear T in a connected manner with $2k-2$ marshals according to the connected monotone strategy for (T, e_T). Remove all $2k-2$ marshals from hyperedges of T after all nodes in T have been cleared. This sort of operation, that of clearing all trees cut off by E that share some node with a given hyperedge e_j of E, will be referred to as *clearing all side routes to e_j*. Once all side routes to e_i have been cleared, place a new marshal in e_{i+1}, clear all side routes to e_{i+1}, move the marshal in e_{i+1} to e_{i+2} and proceed in this way until there is a marshal in e_n and all side routes to e_n have been cleared. Move the marshal from e_n to e_{i-1} and start again the process this time walking back from e_{i-1} to e_1. The total number of marshals required is $2k$.

If e is not in E then it must belong to a subtree T cut off by E that can be cleared with $2k-2$ marshals in a connected monotone manner starting at

e. During this process, once the edge e_T has been cleared, we keep an extra marshal in it to avoid recontamination from the rest of H until T has been completely cleared. Once T is cleared, move the marshal in e_T to a hyperedge e_i in E with which it shares a node. From this point proceed as in the previous case. ■

Corollary 43. *Let H be a tree and let r be the maximum of the cardinality of its hyperedges. Then $\frac{1}{2} \cdot cmim(H) \leq mic(H) \leq r \cdot cmim(H)$.*

Proof. The inequality $(mic(H) \leq r \cdot cmim(H))$ is a consequence of Proposition 41 and the obvious fact that $mim(H) \leq cmim(H)$. Now, for an arbitrary hyperedge e of H, we have

$$\frac{1}{2} \cdot cmim(H) \leq \frac{1}{2} \cdot cmim(H, e) \leq im(H) = mim(H) \leq mic(H)$$

where the first inequality is trivial, the second follows from Proposition 42, and the last from Proposition 41, while the equality follows from Theorem 38. ■

6. Conclusion

We have characterised, in terms of algebra, logic, and combinatorics, structures **B** which have an obstruction set consisting of trees of bounded pathwidth. As we mentioned after Theorem 8, these structures provide a new class of problems CSP(**B**) belonging to the complexity class **NL**. Admittedly, our algebraic characterisation (k-layered m-ABS polymorphisms) is not easy to use. However, we hope that this type of operations may lead to identifying a new type of operations that would be useful for general results putting (co-)CSPs in linear Datalog, in the same way as the operations described in [31] led to a description of (co-)CSPs definable in Datalog [4].

Obviously, k-cattree duality for a structure implies both tree duality and bounded pathwidth duality. It is an open question whether the converse holds, that is, whether every structure that has both tree duality and bounded pathwidth duality also has an obstruction set consisting of trees of bounded pathwidth.

Acknowledgments

The authors would like to thank Isolde Adler and the anonymous referee for their valuable comments. The first author is supported by FCT grant SFRH/BPD/26216/2006 and also by FCT and FEDER within the project ISFL-1-143 of the Centre for Algebra of the University of Lisbon. The second author is supported by grants TIN2006-15387-C03-03 and TIN2007-68005-C04 funded by the MCyT. The third author is supported by the UK EPSRC grants EP/G011001/1 and EP/C543831/1.

References

[1] I. Adler. Marshals, monotone marshals, and hypertree width. *Journal of Graph Theory*, 47(4):275–296, 2004.

[2] F. Afrati and S. Cosmodakis. Expressiveness of restricted recursive queries. In *STOC'89*, pages 113–126, 1989.

[3] L. Barrière, P. Fraigniaud, N. Santoro, and D. Thilikos. Searching is not jumping. In *WG'03*, volume 2880 of *LNCS*, pages 34–45, 2003.

[4] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *FOCS'09*, pages 595–603, 2009.

[5] A. Bulatov. Tractable conservative constraint satisfaction problems. In *LICS'03*, pages 321–330, 2003.

[6] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[7] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[8] A. Bulatov, A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 93–124, 2008.

[9] A. Bulatov and M. Valeriote. Recent results on the algebraic approach to the CSP. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 68–92, 2008.

[10] C. Carvalho, V. Dalmau, and A. Krokhin. Caterpillar duality for constraint satisfaction problems. In *LICS'08*, pages 307–316, 2008.

[11] 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.

[12] N. Creignou, S. Khanna, and M. Sudan. Complexity Classifications of Boolean Constraint Satisfaction Problems. Volume 7 of SIAM Monographs on Discrete Mathematics and Applications, 2001.

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

[14] V. Dalmau and A. Krokhin. Majority constraints have bounded pathwidth duality. *European Journal of Combinatorics*, 29(4):821–837, 2008.

[15] V. Dalmau, A. Krokhin, and B. Larose. First-order definable retraction problems for reflexive graphs and posets. *Journal of Logic and Computation*, 17(1):31–51, 2007.

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

[17] J. Ellis, I.H. Sudborough, and J. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[18] T. Feder and M.Y. 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.

[19] F. Fomin and D. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 41(2):381–393, 2008.

[20] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66:775–808, 2003.

[21] E. Grädel and G. McColm. Hierarchies in transitive closure logic, stratified Datalog and infinitary logic. *Annals of Pure and Applied Logic*, 77(2):169–199, 1996.

[22] N. Kinnersley. The vertex separation number of a graph is equal to its path-width. *Information Processing Letters*, 42(6):345–350, 1992.

[23] L.M. Kirousis and C.H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47(3):205–218, 1986.

[24] Ph.G. Kolaitis. On the expressive power of logics on finite models. In *Finite Model Theory and its Applications*, pages 27–124. Springer, 2007.

[25] Ph.G. Kolaitis and M.Y. Vardi. A logical approach to constraint satisfaction. In *Finite Model Theory and its Applications*, pages 339–370. Springer, 2007.

[26] A.S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40:224–245, 1993.

[27] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoretical Computer Science*, 410(18):1629–1647, 2009.

[28] B. Larose and L. Zádori. Algebraic properties and dismantlability of finite posets. *Discrete Mathematics*, 163:89–99, 1997.

[29] B. Larose and L. Zádori. The complexity of the extendibility problem for finite posets. *SIAM Journal on Discrete Mathematics*, 17(1):114–121, 2003.

[30] B. Larose and L. Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.

[31] M. Maróti and R. McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463–489, 2008.

[32] N. Meggido, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, 1988

[33] R.H. Möhring. Graph problems related to gate matrix layout and PLA folding. *Comput.Suppl.*, 7:17–51, 1990.

[34] J. Nešetřil and V. Rödl. Chromatically optimal rigid graphs. *Journal of Combinatorial Theory, Ser.B*, 46:133–141, 1989.

[35] P.D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Ser.B*, 58:22–33, 1993.

[36] Cs. Szabó and L. Zádori. Idempotent totally symmetric operations on finite posets. *Order*, 18:39–47, 2001.

[37] L. Zádori. Posets, near-unanimity functions and zigzags. *Bulletin of the Australian Mathematical Society*, 47:79–93, 1993.

[38] L. Zádori. Monotone Jónsson operations and near unanimity functions. *Algebra Universalis*, 33(2):216–236, 1995.