On Homomorphism Indistinguishability and Hypertree Depth

Benjamin Scheidt
Humboldt-Universität zu Berlin, Germany

Abstract

GC^k is a logic introduced by Scheidt and Schweikardt (2023) to express properties of hypergraphs. It is similar to first-order logic with counting quantifiers (C) adapted to the hypergraph setting. It has distinct sets of variables for vertices and for hyperedges and requires vertex variables to be guarded by hyperedge variables on every quantification.

We prove that two hypergraphs G, H satisfy the same sentences in the logic GC^k with *guard depth* at most k if, and only if, they are homomorphism indistinguishable over the class of hypergraphs of strict hypertree depth at most k. This lifts the analogous result for tree depth $\leq k$ and sentences of first-order logic with counting quantifiers of quantifier rank at most k due to Grohe (2020) from graphs to hypergraphs. The guard depth of a formula is the quantifier rank with respect to hyperedge variables, and strict hypertree depth is a restriction of hypertree depth as defined by Adler, Gavenčiak and Klimošová (2012). To justify this restriction, we show that for every H, the strict hypertree depth of H is at most 1 larger than its hypertree depth, and we give additional evidence that strict hypertree depth can be viewed as a reasonable generalisation of tree depth for hypergraphs.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Mathematics of computing → Hypergraphs

Keywords and phrases homomorphism indistinguishability, counting logics, guarded logics, hypergraphs, incidence graphs, tree depth, elimination forest, hypertree width

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.152

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2404.10637` [26]

Acknowledgements We thank Nicole Schweikardt for helpful discussions.

1 Introduction

The (k-dimensional) Weisfeiler-Leman algorithm describes a technique to classify the vertices (or k-tuples) of a graph, by iteratively computing a colouring (i.e., a classification) of the vertices (or k-tuples), which gets refined each iteration until it stabilises. While it can be used as a way to (imperfectly) test graphs for isomorphism, it has found many other – seemingly very different – uses, e.g. reducing the cost of solving linear programs [14], as graph kernels [29] or even as an architecture for graph neural networks [30, 22, 13, 12]. For a more in-depth overview on the expressive power of the Weisfeiler-Leman algorithm itself, consult [17] as a starting point. The success of the Weisfeiler-Leman algorithm can in part be explained by its simplicity, but also by the fact that it appears to capture the structure of a graph really well. This can be explained by its connection to first-order logic with counting quantifiers and to homomorphism counts over graphs of bounded tree width. A classical result due to Cai, Fürer and Immerman [5] and Immerman and Lander [16] says that two graphs are indistinguishable by the k-dimensional Weisfeiler-Leman algorithm if, and only if, they satisfy the same sentences of first-order logic with counting quantifiers (C) and k+1 variables (C^{k+1}).

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 152; pp. 152:1–152:18


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Dvořák [9] and Dell, Grohe, Rattan [8] related the Weisfeiler-Leman algorithm to homomorphism counts over graphs of bounded tree width (this was subsequently generalised to relational structures of bounded tree width by Butti and Dalmau [3]). They showed that two graphs are homomorphism indistinguishable over the class TW_k of graphs of tree width at most k if, and only if, they are indistinguishable by the k-dimensional Weisfeiler-Leman algorithm. Here, two graphs G and H are *homomorphism indistinguishable* over a class $\mathfrak{C}$ of graphs, if the number of homomorphisms from F into G equals the number of homomorphisms from F into H for all $F \in \mathfrak{C}$. Dvořák used the previously mentioned connection to C^{k+1} and an inductive characterisation of the graphs of bounded tree width to prove this result, while Dell et al. used elaborate algebraic techniques on vectors containing homomorphism counts.

In recent years, a whole theory has emerged around homomorphism indistinguishability. There are characterisations of homomorphism indistinguishability for classes of graphs other than TW_k (cf. [7, 10, 21, 25]), among which we would like to emphasise the following: A classical result by Lovász [18], stating that two graphs are isomorphic if, and only if, they are homomorphism indistinguishable over all graphs; a well-received result by Mančinska and Roberson [20], stating that two graphs are homomorphism indistinguishable over the class of planar graphs if, and only if, they are quantum isomorphic; and, of importance for this paper, Grohe [11] showed that two graphs are homomorphism indistinguishable over the graphs of tree depth at most m if, and only if, they satisfy the same sentences of C with quantifier rank at most m (C_m). There is also work concerned with a more fundamental analysis of homomorphism counting from restricted classes (cf. [2, 15, 23, 24, 28]).

Some real-world problems can be represented by hypergraphs in a much more natural way than by graphs. The great track record of the Weisfeiler-Leman method poses the question, whether a similar algorithm exists that works on *hypergraphs*. A direct application of the Weisfeiler-Leman algorithm on the incidence structure of a hypergraph is sometimes used. But Böker noted in [4], that this approach does not capture the hypergraph structure well, since the algorithm will mix up hyperedges and vertices. Thus, a proper variant of the Weisfeiler-Leman algorithm that works on hypergraphs is, to the best of our knowledge, still missing. We believe that establishing results analogous to the ones mentioned so far can give valuable insight on how the algorithm should operate on hypergraphs. A first step from this angle is a result by Scheidt and Schweikardt [27], who lift Dvořák's result to hypergraphs by proving the following: two hypergraphs G, H are homomorphism indistinguishable over the class GHW_k of hypergraphs of generalised hypertree width at most k if, and only if, they satisfy the same sentences of the logic GC^k. GC^k is a novel logic introduced in [27]. It has distinct variables for vertices and for hyperedges and counting quantifiers for both variable types. The main feature of GC^k is that it bounds the number of variables for hyperedges by k, and it requires that vertex variables are "guarded by" (i.e., contained in) hyperedge variables on every quantification.

Contributions. As the main contribution of this work, we show that two hypergraphs satisfy the same sentences of the logic GC^k with guard depth at most k if, and only if, they are homomorphism indistinguishable over the class of hypergraphs of strict hypertree depth at most k (Theorem 6.1). The guard depth is the quantifier depth of the hyperedge variables. This theorem follows from an inductive characterisation of the class of hypergraphs of strict hypertree depth $\leq k$, combined with the main technical lemmas of Scheidt and Schweikardt [27]. We believe that this inductive characterisation is interesting on its own, since the same technique combined with the core lemmata in Dvořák's work [9] can be used to give a concise proof of the analogous result on graphs due to Grohe [11], which was

independently recognised and shown by Fluck et al. [10] recently. Strict hypertree depth is a mild restriction of hypertree depth as defined by Adler, Gavenčiak and Klimošová [1]. This (as it turns out only slight) deviation from hypertree depth is surprising at first. Because of the properties and relations between strict hypertree depth and hypertree depth we acquire in this paper, we claim that strict hypertree depth can be viewed as a reasonable generalisation of tree depth for hypergraphs too. In particular, we show that the strict hypertree depth of a hypergraph is at most 1 larger than its hypertree depth (Theorem 2.5). Moreover, we show that the distinguishing power of homomorphism counts from hypergraphs of hypertree depth at most k is different from the distinguishing power of homomorphism counts from their respective incidence graphs (Theorem 2.9). Compared to other hypergraph parameters, this is very unexpected.

Organisation. The remainder of the paper is organised as follows. Section 2 is dedicated to the introduction of the necessary notation and definitions. In particular, we introduce incidence graphs as representations of hypergraphs that will be used throughout the paper, following Böker [4] and Scheidt and Schweikardt [27]. The notions of (strict) hypertree depth are introduced in Section 2.1, followed by Section 2.2 where we handle the differences between homomorphisms between hypergraphs and homomorphisms between incidence graphs. In Section 3 we introduce k-labeled incidence graphs that were the principle tool used in [27] to achieve their result. We utilise them in Section 4 to give an inductive characterisation of the hypergraphs of strict hypertree depth at most k (Theorem 4.1). Section 5 is devoted to the logic GC^k. In Section 6 we combine the results from Section 4 and Section 2.2 with the results from [27] to obtain Theorem 6.1. Section 7 concludes the paper with a summary of the results obtained in this paper, as well as an outlook on further research directions.

2 Preliminaries

Since we heavily rely on the work by Scheidt and Schweikardt [27], we will keep our notation close to theirs. We denote the set of natural numbers *including* 0 by $\mathbb{N}$, the set of *positive* natural numbers by $\mathbb{N}_{\geq 1}$, and we write $[n]$ to denote the set $\{1, 2, \ldots, n\}$. To denote isomorphism of two objects, we use $\cong$. A tuple is denoted using a bar, e.g. $\overline{a}$. For a given ℓ-tuple $\overline{a}$, we use a_i to denote the i-th element of $\overline{a}$, i.e., $\overline{a} = (a_1, a_2, \ldots, a_\ell)$. For any set S, let $\mathcal{P}(S)$ be the set of subsets of S and let $\mathcal{P}_k(S)$ be the subsets of cardinality exactly k. If S is a set of sets, let $\bigcup S = \bigcup_{s \in S} s$.

For a finite set S of cardinality $\ell \in \mathbb{N}$, a total order $<$ on S and any number $d \in \mathbb{N}$, we say that $\langle i_{d+1}, i_{d+2} \ldots, i_{d+\ell} \rangle$ is the *$<$-enumeration of S*, if $i_{d+1} < i_{d+2} < \cdots < i_{d+\ell}$ and $\{i_{d+1}, \ldots, i_{d+\ell}\} = S$. If the order $<$ is clear from the context, we simply say that $\langle i_{d+1}, i_{d+2} \ldots, i_{d+\ell} \rangle$ is the enumeration of S. Note that we usually let $d = 0$, i.e., we usually write $\langle i_1, \ldots, i_\ell \rangle$. Furthermore, the enumeration $\langle i_{d+1}, i_{d+2} \ldots, i_{d+\ell} \rangle$ of S is empty if, and only if, S is empty.

We denote a partial function f from A to B by $f \colon A \rightharpoonup B$, and we let $\mathrm{dom}(f) \coloneqq \{a \in A : f(a) \text{ is defined}\}$ and $\mathrm{img}(f) \coloneqq \{b \in B : \text{ex. } a \in A \text{ s.t. } f(a) = b\}$. We say that two functions f and g are *compatible*, if $f(x) = g(x)$ for all $x \in \mathrm{dom}(f) \cap \mathrm{dom}(g)$. We identify a (partial) function f with the set $\{(x, f(x)) : x \in \mathrm{dom}(f)\}$ whenever we are using set notation on functions. For example, we write $f \subseteq g$ to indicate $\mathrm{dom}(f) \subseteq \mathrm{dom}(g)$ and $f(x) = g(x)$ for all $x \in \mathrm{dom}(f)$. In particular, by $f \cup g$ we denote the function h with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ and $h(x) = f(x)$ for all $x \in \mathrm{dom}(f)$ and $h(x) = g(x)$ for all $x \in \mathrm{dom}(g) \setminus \mathrm{dom}(f)$. Note that f has *precedence* over g, but this only matters if f and g are not compatible. For a

(partial) function f and a set $S \subseteq \mathrm{dom}(f)$ we write $f(S)$ to denote $\{f(x) : x \in S\}$, and we call the function $g \subseteq f$ with $\mathrm{dom}(g) \coloneqq S$ the *restriction of f to S*. Finally, we define partial functions inline like this: $\{a \to 3, b \to 2, c \to 5\}$. In particular, the empty set $\varnothing$ denotes a partial function with empty domain.

Graphs, Trees and Forests. An (undirected) graph is a tuple $G = (V(G), E(G))$, where $V(G)$ is a finite set and $E(G) \subseteq \mathcal{P}_2(V(G))$. For a set $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced by S, i.e., $V(G[S]) \coloneqq S$ and $E(G[S]) \coloneqq E(G) \cap \mathcal{P}_2(S)$. A connected component of a graph is a maximal induced subgraph that is connected. A tree is a connected acyclic graph and a forest is a graph were each connected component is a tree. A rooted tree T is a tree with some distinguished node that we call its *root*, which we denote by ω_T. A rooted forest F is the disjoint union of a collection of rooted trees. It therefore has a set of roots denoted by Ω_F. We may omit the index if it is clear from the context. Note that a rooted tree is also a rooted forest and that every node n in a rooted forest is contained in a unique connected component which is a tree that we call the *tree for n* and whose root is the *root for n*.

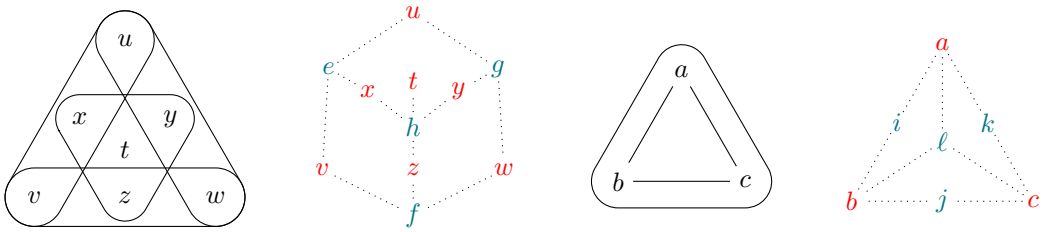
For a rooted forest F we let $\leq_F$ be the induced partial order on the nodes, i.e., the roots are the minimal elements and $s \leq_F t$ if s is on the unique path from t to its root in Ω. By $\mathrm{P}(s,t)$ we denote the set of nodes on the path from s to t (including s and t). In particular, if no path from s to t exists, $\mathrm{P}(s,t) = \varnothing$. By $\mathrm{P}(s)$ we denote the set of nodes on the path from s to the root for s and by $\wedge(s,t)$ we denote the unique element, if it exists, where the paths $\mathrm{P}(s)$, $\mathrm{P}(t)$ join, i.e., $\wedge(s,t) \coloneqq \max_{\leq_F}(\mathrm{P}(s) \cap \mathrm{P}(t))$. Notice that $\wedge(s,t)$ is undefined iff s and t are not in the same tree, and that $\wedge(s,t) = s$, iff $s \leq_F t$ (and conversely, $\wedge(s,t) = t$ iff $t \leq_F s$).

The subtree T_t induced by $t \in V(F)$ is the tree $F[V]$ with root t and $V \coloneqq \{s \in V(T) : t \leq_F s\}$. The *level* of a node $s \in V(F)$ is defined as the number of elements on the path from s to its root, i.e., $\mathrm{level}(s) \coloneqq |\mathrm{P}(s)|$. The *height* of a rooted tree T is the maximal level, i.e., $\mathrm{height}(T) \coloneqq \max\{\mathrm{level}(t) : t \in V(T)\}$ and the height of a node $t \in V(T)$ is the height of its induced subtree T_t, i.e., $\mathrm{height}(t) \coloneqq \mathrm{height}(T_t)$.

Hyper- and Incidence Graphs. A *hypergraph* is a tuple $\mathcal{H} = (V, E, \beta)$, where V and E are disjoint finite sets and β is a total function from E to $\mathcal{P}(V)$ with $V = \bigcup_{e \in E} \beta(e)$. We call the elements of V *vertices* and the elements of E *hyperedges* and for every $e \in E$, we call $\beta(e)$ its *contents*. We denote V by $V(\mathcal{H})$, E by $E(\mathcal{H})$ and β by $\beta_{\mathcal{H}}$, though we may omit the index if there is no ambiguity. Notice that, in general, multiple hyperedges with the same content and hyperedges without content are allowed. We call $\mathcal{H}$ *simple* if β is injective.

An *incidence graph* is a tuple $I = (R, B, E)$ consisting of two disjoint finite sets R and B of *red* and *blue* vertices and a relation $E \subseteq B \times R$. We denote R by $\mathcal{R}(I)$, B by $\mathcal{B}(I)$ and E by $E(I)$. For every $e \in \mathcal{B}(I)$, we let $\beta(e) \coloneqq \{v \in \mathcal{R}(I) : (e,v) \in E(I)\}$. Notice that β is equivalent in its function to the map β for a hypergraph, hence we denote them similarly. We only consider incidence graphs where for every $v \in \mathcal{R}(I)$ there is an $e \in \mathcal{B}(I)$ such that $(e,v) \in E(I)$. It is easy to see that we can assign an incidence graph to every hypergraph and the other way around: For every hypergraph $\mathcal{H}$ we let $I_{\mathcal{H}} \coloneqq (V(\mathcal{H}), E(\mathcal{H}), E)$ where $E \coloneqq \{(e,v) : e \in E(\mathcal{H}), v \in \beta(e)\}$. Conversely, for every incidence graph I we let $\mathcal{H}_I \coloneqq (\mathcal{R}(I), \mathcal{B}(I), \beta)$ where $\beta(e) \coloneqq \{v \in \mathcal{R}(I) : (e,v) \in E(I)\}$ for all $e \in \mathcal{B}(I)$.

For every set $S \subseteq E(\mathcal{H})$ we define the *induced subhypergraph* $\mathcal{H}[S]$ as $(V', S, \beta'_{\mathcal{H}})$ where $V' \coloneqq \bigcup_{e \in S} \beta(e)$ and $\beta'_{\mathcal{H}}$ is the restriction of $\beta_{\mathcal{H}}$ to S. We say that a hypergraph is connected if its incidence graph is connected. An induced subhypergraph is a connected component, if its corresponding incidence graph is a connected component.



(a) The hypergraph $\mathcal{H}$. **(b)** The IG $I_{\mathcal{H}}$ for $\mathcal{H}$. **(c)** The hypergraph $\mathcal{G}$. **(d)** The IG $I_{\mathcal{G}}$ for $\mathcal{G}$.

Figure 1 Examples for hypergraphs and their corresponding incidence graphs.

By $\mathcal{P}_n$ we denote the path of n hyperedges, where each hyperedge contains 2 vertices. I.e., we let $V(\mathcal{P}_n) = [n{+}1]$, $E(\mathcal{P}_n) = \{e_i : i \in [n]\}$ and $\beta(e_i) = \{i, i{+}1\}$ for all $i \in [n]$. We may use different names for the vertices if it is convenient.

▶ **Example 2.1.** The hypergraph $\mathcal{H}$ illustrated in Figure 1a is defined as $V(\mathcal{H}) = \{u, v, w, x, y, z, t\}$, $E(\mathcal{H}) = \{e, f, g, h\}$ and $\beta_{\mathcal{H}} = \{e \to \{u, v, x\}, f \to \{v, w, z\}, g \to \{u, w, y\}, h \to \{t, x, y, z\}\}$. Its incidence graph $I_{\mathcal{H}}$, depicted in Figure 1b, is defined by $\mathcal{R}(I_{\mathcal{H}}) = V(\mathcal{H})$, $\mathcal{B}(I_{\mathcal{H}}) = E(\mathcal{H})$ and $E(I_{\mathcal{H}}) = \{(e,u), (e,v), (e,x), (f,v), (f,w), (f,z), (g,u), (g,w), (g,y), (h,t), (h,x), (h,y), (h,z)\}$.

The hypergraph $\mathcal{G}$ depicted in Figure 1c is defined as $V(\mathcal{G}) = \{a, b, c\}$, $E(\mathcal{G}) = \{i, j, k, \ell\}$ and $\beta_{\mathcal{G}} \coloneqq \{i \to \{a, b\}, j \to \{b, c\}, k \to \{a, c\}, \ell \to \{a, b, c\}\}$. Its incidence graph $I_{\mathcal{G}}$, depicted in Figure 1d, is defined as $\mathcal{R}(I_{\mathcal{G}}) = V(\mathcal{G})$, $\mathcal{B}(I_{\mathcal{G}}) = E(\mathcal{G})$ and $E(I_{\mathcal{G}}) = \{(i,a), (i,b), (j,b), (j,c), (k,a), (k,c), (\ell,a), (\ell,b), (\ell,c)\}$.

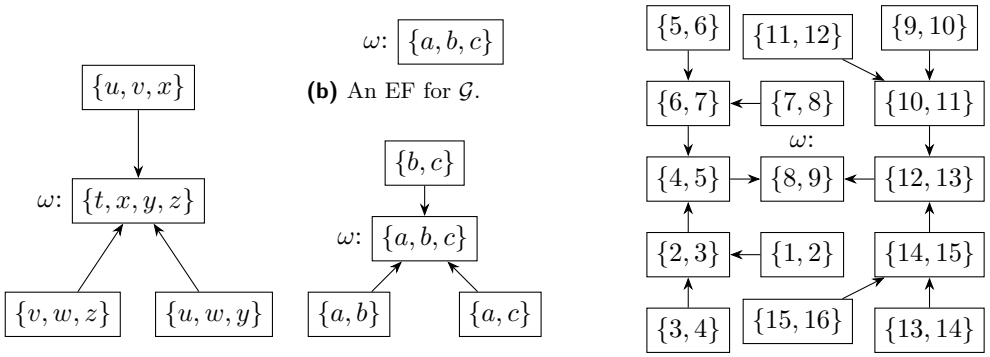
2.1 Hypertree Depth

The following definition of *elimination forest* and *hypertree depth* is due to Adler, Gavenčiak and Klimošová [1], though they refer to elimination forests as "decomposition forests". We call them elimination forests, since this reflects their conceptual similarity to elimination forests for graphs and avoids confusion with hypertree decompositions. Further, we define this notion in terms of incidence graphs, because we mainly work on those. Do notice however, that this definition easily translates to hypergraphs and that it is equivalent to the one given by Adler, Gavenčiak and Klimošová.

▶ **Definition 2.2** (Hypertree Depth and Elimination Forests, [1])**.** Let I be an incidence graph. An *elimination forest* (F, Γ) for I consists of a forest F and a mapping $\Gamma\colon V(F) \to \mathcal{B}(I)$ such that conditions 1–3 hold. We write $\widehat{\Gamma}(t)$ as shorthand for $\beta(\Gamma(t))$.

1. *Completeness for vertices*: For every red vertex $v \in \mathcal{R}(I)$, there is a $t \in V(F)$ such that $v \in \widehat{\Gamma}(t)$.
2. *Hyperedge-Containment*: For every blue vertex $e \in \mathcal{B}(I)$ there are nodes $s, t \in V(F)$ such that $s \leq_F t$ and $\beta(e) \subseteq \bigcup \widehat{\Gamma}(\mathrm{P}(s,t))$.
3. *Shared heritage*: For all $s, t \in V(F)$, if $\widehat{\Gamma}(s) \cap \widehat{\Gamma}(t) \neq \varnothing$, then $\wedge(s,t)$ is defined and $\widehat{\Gamma}(s) \cap \widehat{\Gamma}(t) \subseteq \bigcup \widehat{\Gamma}(\mathrm{P}(\wedge(s,t)))$.

The intuition behind condition 3 is that hyperedges can only share the vertices contained in their common ancestors in the elimination forest. The height of an elimination forest (F, Γ) is simply the height of F. The *hypertree depth of* I is defined as the minimal height over all elimination forests for I, and we denote it by $\mathrm{hd}(I)$. Analogously, we let $\mathrm{hd}(\mathcal{H}) \coloneqq \mathrm{hd}(I_{\mathcal{H}})$ for all hypergraphs. We write IHD_k to denote the class of incidence graphs of hypertree depth at most k and HD_k to denote the corresponding class of hypergraphs. ⌟



(a) A (strict) EF for $\mathcal{H}$. **(b)** An EF for $\mathcal{G}$. **(c)** A strict EF for $\mathcal{G}$. **(d)** A (strict) elimination forest for $\mathcal{P}_{15}$.

Figure 2 Elimination forests for various (hyper)graphs.

We call an elimination forest (F, Γ) *strict* if Γ is bijective. It is easy to see that the first and second condition are trivially satisfied when Γ is bijective, thus we can redefine the notion of strict elimination forest in a more succinct manner.

▶ **Definition 2.3** (Strict Elimination Forest)**.** Let I be an incidence graph. A *strict elimination forest* (F, Γ) for I consists of a forest F and a *bijective* function $\Gamma \colon V(F) \to \mathcal{B}(I)$ satisfying condition 3 of Definition 2.2. The *strict hypertree depth of* I, denoted by $\mathrm{shd}(I)$, is defined as the minimal height over all strict elimination forests for I. Again, we let $\mathrm{shd}(\mathcal{H}) \coloneqq \mathrm{shd}(I_{\mathcal{H}})$ for every hypergraph $\mathcal{H}$. We write ISHD_k to denote the class of incidence graphs of strict hypertree depth at most k and SHD_k to denote the corresponding class of hypergraphs. ⌟

▶ **Example 2.4.** Consider the hypergraphs $\mathcal{G}$ and $\mathcal{H}$ as well as their incidence graphs $I_{\mathcal{G}}$, $I_{\mathcal{H}}$ from Example 2.1 (see Figure 1). Let (F_1, Γ_1) be defined as follows. F_1 is a tree defined by $V(F_1) = \{t_1, t_2, t_3, t_4\}$ and $E(F_2) \coloneqq \{\{t_1, t_2\}, \{t_1, t_3\}, \{t_1, t_4\}\}$ with root t_1. Γ_1 is a map defined as $\{t_1 \to h, t_2 \to e, t_3 \to f, t_4 \to g\}$. (F_1, Γ_1) is an elimination forest of height 2 for $I_{\mathcal{H}}$. It is depicted in Figure 2a, where we labeled the node t_i with $\widehat{\Gamma}_1(t_i)$. Notice, that (F_1, Γ_1) is strict.

Analogously, we depicted two elimination forests for $I_{\mathcal{G}}$ of height 1 and 2 in Figures 2b and 2c. Notice how the elimination forest depicted in Figure 2b, witnessing $\mathrm{hd}(\mathcal{G}) = 1$, is not strict. It is easy to see that $\mathrm{shd}(\mathcal{G}) \geq 2$ since a bijective map implies that there are as many nodes in the forest as there are edges. Thus, Figure 2c witnesses that $\mathrm{shd}(\mathcal{G}) = 2$.

Finally, Figure 2d depicts a (strict) elimination forest for $\mathcal{P}_{15}$ witnessing $\mathrm{shd}(\mathcal{P}_{15}) \leq 4$. Recall that $\mathcal{P}_{15}$ is defined by $V(\mathcal{P}_{15}) = [16]$ and $E(\mathcal{P}_{15}) = \{\{i, i{+}1\} : i \in [15]\}$. ⌟

One can show that the strict hypertree depth of a hypergraph is at most its hypertree depth increased by one. The main idea is that we can turn every elimination forest into a strict one, if we add a leaf for every hyperedge that is currently not being mapped to below the path that contains it according to condition 2 in Definition 2.2. The proof can be found in the full version.

▶ **Theorem 2.5.** *For all hypergraphs* $\mathcal{H}$, $\mathrm{hd}(\mathcal{H}) \leq \mathrm{shd}(\mathcal{H}) \leq \mathrm{hd}(\mathcal{H}){+}1$.

Finally, it is easy to see that, due to condition 3 (shared heritage), every elimination forest of a connected incidence graph is also an *elimination tree*.

▶ **Lemma 2.6.** *Let* I *be a connected incidence graph. For every elimination forest* (F, Γ) *of* I*,* F *is a tree.*

2.2 Homomorphisms

While hypergraphs and incidence graphs are conceptually close, their "natural" notions of homomorphisms are not the same. Since our interest lies in hypergraphs, but we are mainly working on incidence graphs in this paper, we have to relate these notions. Following Scheidt and Schweikardt, we use the same definitions as Böker [4].

A homomorphism from a hypergraph $\mathcal{H}$ into another hypergraph $\mathcal{G}$ is a pair of functions $(h_V\colon V(\mathcal{H}) \to V(\mathcal{G}), h_E\colon E(\mathcal{H}) \to E(\mathcal{G}))$ such that for every $e \in E(\mathcal{H})$ the equality $h_V(\beta(e)) = \beta(h_E(e))$ holds.

A homomorphism from an incidence graph I into another incidence graph J is a pair of mappings $(h_V\colon \mathcal{R}(I) \to \mathcal{R}(J), h_E\colon \mathcal{B}(I) \to \mathcal{B}(J))$ such that $(h_E(e), h_V(v)) \in E(J)$ holds for every edge $(e, v) \in E(I)$. This is equivalent to the requirement $h_V(\mathrm{p}(e)) \subseteq \mathrm{p}(h_E(e))$. Thus, the equality that we require for a hypergraph homomorphism is relaxed to an inclusion for incidence graphs.

Let A, B and $\mathfrak{C}$ be two hypergraphs and a class of hypergraphs or two incidence graphs and a class of incidence graphs. We denote the number of homomorphisms from A to B by $\hom(A, B)$, and we let $\mathrm{Hom}(\mathfrak{C}, A)$ be the "vector" that has a row for every $F \in \mathfrak{C}$ containing $\hom(F, A)$. We say that A and B are *homomorphism indistinguishable* over $\mathfrak{C}$ ($A \equiv_{\mathfrak{C}} B$), if $\mathrm{Hom}(\mathfrak{C}, A) = \mathrm{Hom}(\mathfrak{C}, B)$, i.e., if $\hom(F, A) = \hom(F, B)$ for all $F \in \mathfrak{C}$.

The following crucial theorem relates homomorphism indistinguishability over a class of hypergraphs to homomorphism indistinguishability over the corresponding class $\mathfrak{C}_{\mathsf{I}}$ of incidence graphs. As noted in [27], this theorem is implicit in [4], consult Appendix A of the full version of [27] for details.

▶ **Theorem 2.7** ([4, 27]). *Let $\mathfrak{C}$ be a class of hypergraphs and let $\mathfrak{C}_{\mathsf{I}}$ be its corresponding class of incidence graphs. If $\mathfrak{C}$ is closed under* pumping *and* local merging*, then $\mathrm{Hom}(\mathfrak{C}, \mathcal{G}) = \mathrm{Hom}(\mathfrak{C}, \mathcal{H})$ if, and only if, $\mathrm{Hom}(\mathfrak{C}_{\mathsf{I}}, I_{\mathcal{G}}) = \mathrm{Hom}(\mathfrak{C}_{\mathsf{I}}, I_{\mathcal{H}})$ for all hypergraphs $\mathcal{G}$ and $\mathcal{H}$.*

$\mathfrak{C}$ is closed under pumping, if $H' \in \mathfrak{C}$ for every $H \in \mathfrak{C}$, where H' is created from H by inserting a new vertex into *one* arbitrary hyperedge of H; and closed under local merging, if $H' \in \mathfrak{C}$ for every $H \in \mathfrak{C}$, where H' is created from H by choosing an arbitrary hyperedge e and then merging two vertices u, v that are both contained in e.

It is easy to see that SHD_k is closed under both pumping and local merging, whereas HD_k is only closed under local merging.

▶ **Proposition 2.8.** *Let $k \in \mathbb{N}_{\geq 1}$. The class SHD_k is closed under pumping and local merging, the class HD_k is closed under local merging but* ***not*** *under pumping.*

The following theorem shows that homomorphism indistinguishability over HD_k is not equal to homomorphism indistinguishability over SHD_k. Because $\mathsf{SHD}_k \subseteq \mathsf{HD}_k$, counting homomorphisms from HD_k is more powerful in the sense that it distinguishes more hypergraphs. But we also show that it is unequal to homomorphism indistinguishability over IHD_k, which is unexpected. Since this prohibits us from relating HD_k to any fragment of GC, it is conceivable that this could pose a problem in other scenarios too. Thus, we argue that the notion of strict hypertree depth can be viewed as a reasonable generalisation of tree depth, especially when we recall that $\mathsf{HD}_{k-1} \subseteq \mathsf{SHD}_k \subseteq \mathsf{HD}_k$.

▶ **Theorem 2.9.** *For every $k \in \mathbb{N}_{\geq 1}$ there exist pairs of hypergraphs $(\mathcal{G}_k, \mathcal{H}_k)$ and $(\mathcal{G}'_k, \mathcal{H}'_k)$, such that:*

1. $\mathrm{Hom}(\mathsf{SHD}_k, \mathcal{G}_k) = \mathrm{Hom}(\mathsf{SHD}_k, \mathcal{H}_k)$, *but* $\mathrm{Hom}(\mathsf{HD}_k, \mathcal{G}_k) \neq \mathrm{Hom}(\mathsf{HD}_k, \mathcal{H}_k)$;
2. $\mathrm{Hom}(\mathsf{HD}_k, \mathcal{G}'_k) = \mathrm{Hom}(\mathsf{HD}_k, \mathcal{H}'_k)$, *but* $\mathrm{Hom}(\mathsf{IHD}_k, I_{\mathcal{G}'_k}) \neq \mathrm{Hom}(\mathsf{IHD}_k, I_{\mathcal{H}'_k})$.

(a) $\mathcal{G}_1$. **(b)** $\mathcal{H}_1$. **(c)** $\mathcal{G}'_1$. **(d)** $\mathcal{H}'_1$.

Figure 3 $(\mathcal{G}_1, \mathcal{H}_1)$, $(\mathcal{G}'_1, \mathcal{H}'_1)$ witness Theorem 2.9 for $k = 1$. Circles denote singleton hyperedges.

For $k = 1$ this is easy to see: A connected hypergraph has strict hypertree depth 1 iff it consists of a single hyperedge, whereas a connected hypergraph has hypertree depth 1 if one hyperedge contains all vertices. It is therefore not hard to see that the statement of the theorem holds for $k = 1$ using the hypergraphs depicted in Figure 3. For $k \geq 2$ a similar idea for the construction of $(\mathcal{G}_k, \mathcal{H}_k)$ and $(\mathcal{G}'_k, \mathcal{H}'_k)$ works, but we had to defer the details to the full version due to space constraints.

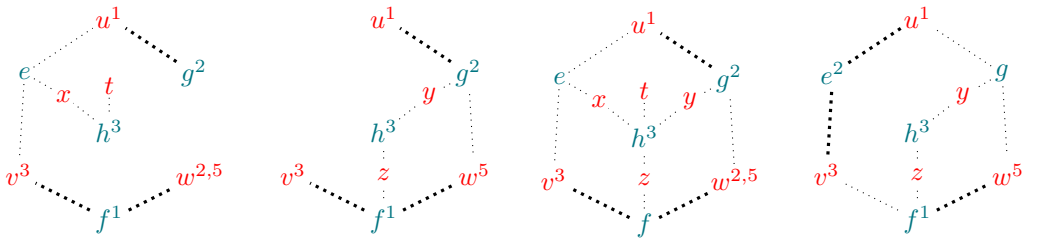
3 k-Labeled Incidence Graphs

Our goal is to give an inductive characterisation of the incidence graphs of strict hypertree depth at most k (and thus also of hypergraphs of strict hypertree depth at most k). The concepts presented in this section were first defined in [27], and we adopt their notation and phrasing for the most part. Note that the k-labeled incidence graphs defined here are inspired by the concept of k-labeled graphs as they are used in [6, 9, 10, 19] and elsewhere. In particular, k-labeled graphs are the main tool used by Dvořák [9] to prove his result. In principle, a *k-labeled incidence graph* is an incidence graphs that has *labels* attached to some of its red and blue vertices. We have an unbounded number of red labels that can be attached to red vertices (though the number of labels actually used must always be finite), but we only have k labels that we can attach to blue vertices. We are allowed to attach multiple labels to the same vertex, but we are not required to use all of them. Every red label has an assigned "guard", which is a blue vertex with a label on it. In practice, we will require every red labeled vertex to be a neighbour of its guard (i.e., we want it to have a *real* guard, as defined in the next paragraph), though it makes the proofs easier if we do not enforce this in the definition itself. This idea is formalised as follows.

A k-labeled incidence graph is a tuple $L = (I, r, b, g)$, where I is an incidence graph, $r\colon \mathbb{N}_{\geq 1} \rightharpoonup \mathcal{R}(I)$, $b\colon [k] \rightharpoonup \mathcal{B}(I)$ and $g\colon \mathbb{N}_{\geq 1} \rightharpoonup [k]$ are partial mappings such that $\mathrm{dom}(r)$ is finite and $\mathrm{dom}(g) = \mathrm{dom}(r)$. We use I_L, r_L, ... to denote the components of L. But to keep the indices from getting overly complicated, we may write $I', r', \ldots$ and $I_i, r_i, \ldots$ instead of $I_{L'}, r_{L'}$ and $I_{L_i}, r_{L_i}, \ldots$, respectively. If it is clear from the context, we may omit the index altogether and simply write I, r, b, g.

We say that L *has real guards (w.r.t. g)*, if for every $i \in \mathrm{dom}(r)$ we have $g(i) \in \mathrm{dom}(b)$ and $(b(g(i)), r(i)) \in E(I)$. A k-labeled incidence graph L is *label-free* if $r_L = b_L = g_L = \varnothing$. We call I the *skeleton* of L. Next, we define some operations on k-labeled incidence graphs.

For any set $X_r \subseteq \mathbb{N}_{\geq 1}$ of finite size ℓ and any tuple $\overline{v} = (v_1, v_2, \ldots, v_\ell) \in \mathcal{R}(I_L)^n$ we write $L[X_r \to \overline{v}]$ to denote a copy of L where we modified r such that $r(i_j) = v_j$ for all j in the enumeration $\langle i_1, \ldots, i_\ell \rangle$ of X_r, i.e., we introduce, and change the placement of, some red labels. Similarly, for any $X_b = \{i_1, \ldots, i_\ell\} \subseteq [k]$ and any $\overline{e} = (e_1, e_2, \ldots, e_\ell) \in \mathcal{B}(I_L)^\ell$ we write $L\langle X_b \to \overline{e} \rangle$ to denote a copy of L where we modified b accordingly. We write $L[X_r \to \bullet]$ ($L\langle X_b \to \bullet \rangle$) to denote a copy of L where we *removed* the red (blue) labels in X_r (X_b). Note, that we remove *just* the labels and *not* the vertices that carry them.



(a) The 3-labeled incidence graph L_1.

(b) The 3-labeled incidence graph L_2.

(c) An example of glueing: $(L_1 \cdot L_2)$.

(d) An example of applying a transition: $L_1[\rightsquigarrow f]$.

Figure 4 3-labeled incidence graphs and operations on them. Labels are encoded as exponents and the guard function is encoded using thicker edges between the red vertex and its guard.

Intuitively, the "product" $(L_1 \cdot L_2)$ or *glueing* of two k-labeled incidence graphs L_1, L_2 is the k-labeled incidence graph L that is created by first taking the disjoint union of L_1 and L_2, followed by repeatedly merging pairs of red (blue) vertices, that carry a shared red (blue) label. By merging we mean that we replace these vertices by a single fresh vertex, which inherits their neighbourhoods and labels. We apply this procedure until there are no more such pairs. The guard function of $(L_1 \cdot L_2)$ is simply $g_{L_1} \cup g_{L_2}$, i.e., in theory, g_{L_1} has precedence over g_{L_2}. In practice, we will require that g_{L_1} and g_{L_2} are compatible, which means the precedence of g_{L_1} will be irrelevant. Note that the order in which we merge vertices does not matter, and that if a vertex carries two or more labels, all vertices carrying any one of these labels will be replaced by a single fresh vertex that carries *all* those labels and inherits *all* neighbourhoods. Finally, for $i \in [2]$ we define mappings $\mathrm{succ}^R_{L_i} \colon \mathcal{R}(I_{L_i}) \to \mathcal{R}(I_L)$ and $\mathrm{succ}^B_{L_i} \colon \mathcal{B}(I_{L_i}) \to \mathcal{B}(I_L)$ such that $\mathrm{succ}^R_{L_i}(v)$ is the red vertex of I_L that corresponds to $v \in \mathcal{R}(I_i)$, and $\mathrm{succ}^B_{L_i}(e)$ is the blue vertex of I_L that corresponds to $e \in \mathcal{B}(I_i)$.

▶ **Example 3.1.** Consider the k-labeled incidence graphs L_1, L_2 according to Figures 4a and 4b. In particular, we have

i	1	2	3	5
$r_1(i)$	u	w	v	w
$r_2(i)$	u	–	v	w

i	1	2	3
$b_1(i)$	f	g	h
$b_2(i)$	f	g	h

and

i	1	2	3	5
$g_1(i)$	2	1	1	1
$g_2(i)$	2	–	1	1

.

The product $(L_1 \cdot L_2)$ is depicted in Figure 4c.

So far, we should not be allowed to remove a blue label from a vertex, if it serves as the guard of a red label. But sometimes we want to *transition* from one (real) guard assignment to another (real) guard assignment. I.e., we want to remove blue labels even if they still guard some red labels, because we guarantee that we introduce new guards for these labels right away. We formalise this operation as a special partial function, that assigns new guards to existing red labels: We call $f \colon \mathbb{N}_{\geq 1} \rightharpoonup [k]$ a *transition for L (for g)*, if $\varnothing \neq \mathrm{dom}(f) \subseteq \mathrm{dom}(g)$ and for all $i \in \mathrm{dom}(g)$ we have that if $g(i) \in \mathrm{img}(f)$, then $i \in \mathrm{dom}(f)$. This means that if f reassigns the blue label guarding the red label i, then f provides a new guard for i. Applying a transition, denoted by $L[\rightsquigarrow f]$, means modifying a copy of L as follows: we want to insert fresh vertices with these blue labels, thus we must first remove all blue labels, that are currently in use, i.e., we must first remove the labels in the set $X_b \coloneqq \mathrm{img}(f) \cap \mathrm{dom}(b_L) \cap \mathrm{img}(g_L)$ from b. Notice that we have to intersect with $\mathrm{dom}(b_L)$ since we do not require L to have real guards. After removing the labels in X_b, we insert $|X_b|$ new blue vertices into I_L, each carrying one of the blue labels in X_b, and introduce an edge between $b(f(i))$ and $r(i)$ for all

$i \in \mathrm{dom}(f)$. Finally, we redefine the guard function as $f \cup g_L$. Note that this procedure can be easily expressed as the product $(M_f \cdot L\langle X_b \to \bullet\rangle)$ for a suitably defined k-labeled incidence graph M_f.

▶ **Example 3.2.** Consider the k-labeled incidence graph L_1 from Example 3.1 and Figure 4a. The partial function $f = \{1 \to 2, 3 \to 2\}$ is a transition for L_1. The result $L_1[\rightsquigarrow f]$ of the application of f on L_1 is depicted in Figure 4d.

We define the class GLI_k^i of k-labeled incidence graphs that can be constructed in a way that at most i blue labels are removed "in series".

▶ **Definition 3.3.** For $k \in \mathbb{N}_{\geq 1}$ and $i \in \mathbb{N}$ we define the set GLI_k^i inductively as follows.

Base case. $L \in \mathsf{GLI}_k^0$ for all k-labeled incidence graphs L with $\mathrm{dom}(r) = \mathcal{R}(I)$, $\mathrm{dom}(b) = \mathcal{B}(I)$ and real guards.
For all $i \in \mathbb{N}$, if $L \in \mathsf{GLI}_k^i$, then $L \in \mathsf{GLI}_k^{i+1}$.

Glueing. Let $L_1 \in \mathsf{GLI}_k^{i_1}$, $L_2 \in \mathsf{GLI}_k^{i_2}$ have compatible guard functions and $L = (L_1 \cdot L_2)$.
Then, $L \in \mathsf{GLI}_k^i$ where $i := \max\{i_1, i_2\}$.

Transitioning. Let $L \in \mathsf{GLI}_k^i$, let f be a transition for L and $L' = L_1[\rightsquigarrow f]$.
Then, $L' \in \mathsf{GLI}_k^{i'}$ where $i' := i + |\,\mathrm{img}(f) \cap \mathrm{img}(b_L) \cap \mathrm{img}(g_L)|$.

Label-Removal. Let $L \in \mathsf{GLI}_k^i$.

(a) For $X_r \subseteq \mathrm{dom}(r)$, $L[X_r \to \bullet] \in \mathsf{GLI}_k^i$.

(b) For $X_b \subseteq \mathrm{dom}(b) \setminus \mathrm{img}(g)$, $L\langle X_b \to \bullet\rangle \in \mathsf{GLI}_k^{i'}$ where $i' := i + |X_b|$.

Finally, we let $\mathsf{GLI}^k := \mathsf{GLI}_k^k$ for every $k \in \mathbb{N}$. ⌟

4 Characterising Hypergraphs of Strict Hypertree Depth at most k

In this section we prove that the inductively defined class GLI^k corresponds precisely to the class ISHD_k.

▶ **Theorem 4.1.** *An incidence graph J has strict hypertree depth at most k if, and only if, there exists a label-free $L \in \mathsf{GLI}^k$ such that $I_L \cong J$.*

In the following, we first show how to construct an incidence graph of strict hypertree depth at most k as the skeleton of a label-free k-labeled incidence graph in GLI^k (Lemma 4.2). Then we show that every label-free $L \in \mathsf{GLI}^k$ has strict hypertree depth at most k (Lemma 4.3). Theorem 4.1 follows directly from the combination of these two Lemmata.

For the rest of this section, let $J \in \mathsf{ISHD}_k$ and let (T, Γ) be a strict elimination forest of height $\leq k$ for J. We can w.l.o.g. assume that J is connected and that T is a tree (Lemma 2.6). Let $\mathcal{R}(J) = \{v_1, v_2, \ldots, v_m\}$ where $m \geq 1$.

(T, Γ) will help us decide when to remove (i.e., eliminate) which label from which blue vertex in the following sense. The core idea is to start with a trivial k-labeled incidence graph for every path from a leaf to the root in the elimination tree. Then we walk bottom-up along these paths and whenever several paths join in a node, we apply red and blue vertex removals in a suitable way on their k-labeled incidence graphs, such that afterwards we can glue them together and receive a k-labeled incidence graph that is isomorphic to the incidence graph induced by the union of said paths.

For this we need the following notions: For a node n in a tree T, the *subtree with stem induced by* n is the tree $\dot{T}_n$ induced on T by the set $\mathrm{P}(n) \cup \{t \in V(T) : n \leq_T t\}$. Recall that $\mathrm{P}(n)$ is the set of nodes on the path from n to the root ω (including n and ω), and

notice that the subtree with stem induced by the root is T, i.e., $T_\omega = T$, and for every leaf n it is the path from the root ω to n. For every node $n \in V(T)$ we define the set $\text{labels}(n) \coloneqq \{i \in [m] : v_i \in \widehat{\Gamma}(n)\}$. To avoid an overload of notation, we will write $\text{labels}(N)$ to denote the set $\bigcup_{n\in N} \text{labels}(n)$ and write $J[\dot{T}_n]$ to abbreviate $J[\widehat{\Gamma}(V(\dot{T}_n))]$. With these notions, we can prove the following lemma via induction.

▶ **Lemma 4.2.** *For every $n \in V(T)$ of level d where $\langle t_1, \dots, t_d\rangle$ is a $\leq_T$-enumeration of $\mathrm{P}(n)$ (i.e., in particular $t_1 = \omega$, $t_d = n$), there exists an $L_n \in \mathsf{GLI}_k^{k-d}$ of the form (I, r, b, g) such that*

(A) $\text{dom}(b) = [d]$ *and* $\text{dom}(r) = \text{labels}(\mathrm{P}(n))$*;*

(B) $g(i) \coloneqq \min\{j \in [d] : v_i \in \widehat{\Gamma}(t_j)\}$ *for every* $i \in \text{dom}(g)$*;*

(C) *There exists an isomorphism (π_R, π_B) between I and $J[\dot{T}_n]$ such that*

- **(i)** $\pi_R(r(i)) = v_i$ *for all* $i \in \text{dom}(r)$*, and*
- **(ii)** $\pi_B(b(j)) = \Gamma(t_j)$ *for all* $j \in \text{dom}(b)$*.*

Notice that, in particular, this lemma states $J \cong I_{L_\omega}$ for the root ω of T. But L_ω is not label-free, since $\text{dom}(b_{L_\omega}) = \{1\}$ and $\text{dom}(r_{L_\omega}) = \text{labels}(\mathrm{P}(\omega)) = \text{labels}(\omega)$. But, the lemma also states that $L_\omega \in \mathsf{GLI}_k^{k-1}$, since $\text{level}(\omega) = 1$. Thus, $J \cong I_{L'}$ for $L' = L_\omega[\text{labels}(\omega)\to\bullet]\langle\{1\}\to\bullet\rangle$, and in particular, $L' \in \mathsf{GLI}_k^k$ is label-free. Thus, this lemma shows the forward direction of Theorem 4.1.

The following lemma can be shown by induction. On a high level, the idea of the proof is to only modify the elimination forest, if blue labels are removed. At that point, we prepend a chain of new nodes to the root(s) of the elimination forest. If we take the product of two k-labeled incidence graphs, we take the union of the forests, and if we remove red labels, we do not alter the forest at all.

▶ **Lemma 4.3.** *For every $L \in \mathsf{GLI}_k^d$ of the form (I, r, b, g) there is a tuple (F, Γ), where F is a forest of height $\leq d$ and Γ is a bijective function from $V(F)$ to $\mathcal{B}(I) \setminus \text{img}(b)$ satisfying condition A. We write $\widehat{\Gamma}(t)$ as a shorthand for $\beta(\Gamma(t))$ and $\widetilde{\Gamma}(t)$ as a shorthand for $\widehat{\Gamma}(t) \setminus \text{img}(r)$.*

(A) *For all $s, t \in V(F)$, and all $v \in \widetilde{\Gamma}(s) \cap \widetilde{\Gamma}(t) \neq \varnothing$ it holds that:*
$v \in \beta(b(j))$ for a $j \in \text{dom}(b)$ or $\wedge(s, t)$ is defined and $v \in \bigcup \widetilde{\Gamma}(\mathrm{P}(\wedge(s, t)))$.

Notice that, if $L \in \mathsf{GLI}_k^d$ is label-free, this guarantees a strict elimination forest (F, Γ) of height d for I_L. This shows the backward direction of Theorem 4.1.

5 The Logic GC^k

This section introduces the logic GC^k as defined in [27] and its restricted fragment GC_k, consisting of all formulas of guard depth at most k. Let k be a positive natural number, that is fixed for this section.

Variables. GC^k uses two different kinds of variables: $\mathsf{VAR}_\mathsf{v} \coloneqq \{\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3 \dots\}$ to address vertices and $\mathsf{VAR}_\mathsf{e} \coloneqq \{\mathsf{e}_1, \mathsf{e}_2, \dots, \mathsf{e}_k\}$ to address hyperedges. Notice that the number of variables for hyperedges is bounded by k, but unbounded for vertices. We say that a tuple of the form $\overline{\mathsf{v}} = (\mathsf{v}_{i_1}, \dots, \mathsf{v}_{i_\ell}) \in \mathsf{VAR}_\mathsf{v}^\ell$ or $\overline{\mathsf{e}} = (\mathsf{e}_{i_1}, \dots, \mathsf{e}_{i_\ell}) \in \mathsf{VAR}_\mathsf{e}^\ell$ is a v- *or* e-*tuple*, if $i_1 < i_2 < \dots < i_\ell$. We let $\text{vars}(\overline{\mathsf{v}}) \coloneqq \{\mathsf{v}_{i_1}, \dots, \mathsf{v}_{i_\ell}\}$ and $\text{vars}(\overline{\mathsf{e}}) \coloneqq \{\mathsf{e}_{i_1}, \dots, \mathsf{e}_{i_\ell}\}$ respectively. We call $\{i_1, \dots, i_\ell\}$ the *index set* of $\overline{v}$ and $\overline{e}$, respectively.

Logical Guards. The key idea behind GC^k is that on quantification, vertex variables must be *guarded* by hyperedge variables. This is formalised by a partial function $g\colon \mathbb{N}_{\geq 1} \rightharpoonup [k]$ with finite domain (similar to the guard function of a k-labeled incidence graph, cf. Section 3) and its corresponding *logical guard* $\Delta_g \coloneqq \bigwedge_{i \in \mathrm{dom}(g)} E(\mathsf{e}_{g(i)}, \mathsf{v}_i)$. For the special partial function g with empty domain, we let $\Delta_g \coloneqq \top$, which is a special formula that always evaluates to true.

▶ **Definition 5.1.** The logic GC^k is inductively defined along with the *free vertex variables*, the *free hyperedge variables* and the *guard depth*, as formalised by the functions

$$\mathrm{free}_{\mathsf{v}}\colon \mathsf{GC}^k \to \mathcal{P}(\mathsf{VAR}_{\mathsf{v}}), \quad \mathrm{free}_{\mathsf{e}}\colon \mathsf{GC}^k \to \mathcal{P}(\mathsf{VAR}_{\mathsf{e}}), \quad \text{and} \quad \mathrm{gd}\colon \mathsf{GC}^k \to \mathbb{N}.$$

Atomic Formulas. For all $i, i' \in \mathbb{N}_{\geq 1}$ and all $j, j' \in [k]$ the following formulas are in GC^k:

- $\varphi = \mathsf{v}_i = \mathsf{v}_{i'}$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \{\mathsf{v}_i, \mathsf{v}_{i'}\}$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \varnothing$;
- $\varphi = \mathsf{e}_j = \mathsf{e}_{j'}$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \varnothing$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \{\mathsf{e}_j, \mathsf{e}_{j'}\}$;
- $\varphi = E(\mathsf{e}_j, \mathsf{v}_i)$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \{\mathsf{v}_i\}$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \{\mathsf{e}_j\}$.

In all the above cases, $\mathrm{gd}(\varphi) \coloneqq 0$.

Inductive Rules. Let χ, ψ be formulas of GC^k. The following formulas are in GC^k.

- $\varphi = \neg\chi$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{v}}(\chi)$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{e}}(\chi)$, and $\mathrm{gd}(\varphi) \coloneqq \mathrm{gd}(\chi)$;
- $\varphi = (\chi \wedge \psi)$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{v}}(\chi) \cup \mathrm{free}_{\mathsf{v}}(\psi)$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{e}}(\chi) \cup \mathrm{free}_{\mathsf{e}}(\psi)$, and $\mathrm{gd}(\varphi) \coloneqq \max\{\mathrm{gd}(\chi), \mathrm{gd}(\psi)\}$.

Note that by the rules defined so far, $\mathrm{gd}(\Delta_g) = 0$ for all logical guards Δ_g.

We say that $g\colon \mathbb{N}_{\geq 1} \rightharpoonup [k]$ *is a guard function for* φ if $\mathrm{dom}(g) = \{i : \mathsf{v}_i \in \mathrm{free}_{\mathsf{v}}(\varphi)\}$.

Let $n \in \mathbb{N}_{\geq 1}$, let g be a guard function for ψ and $\chi = (\Delta_g \wedge \psi)$. The following formulas are in GC^k for every v-tuple $\overline{\mathsf{v}}$ with $\mathrm{vars}(\overline{\mathsf{v}}) \subseteq \mathrm{free}_{\mathsf{v}}(\chi)$ and every e-tuple $\overline{\mathsf{e}}$ with $\mathrm{vars}(\overline{\mathsf{e}}) \subseteq \mathrm{free}_{\mathsf{e}}(\chi)$:

- $\varphi = \exists^{\geq n} \overline{\mathsf{v}}\,.\,\chi$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{v}}(\chi) \setminus \mathrm{vars}(\overline{\mathsf{v}})$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{e}}(\chi)$, and $\mathrm{gd}(\varphi) \coloneqq \mathrm{gd}(\chi)$;
- $\varphi = \exists^{\geq n} \overline{\mathsf{e}}\,.\,\chi$ with $\mathrm{free}_{\mathsf{v}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{v}}(\chi)$ and $\mathrm{free}_{\mathsf{e}}(\varphi) \coloneqq \mathrm{free}_{\mathsf{e}}(\chi) \setminus \mathrm{vars}(\overline{\mathsf{e}})$, and $\mathrm{gd}(\varphi) \coloneqq \mathrm{gd}(\chi) + |\mathrm{vars}(\overline{\mathsf{e}})|$. ⌟

For convenience, we let $\mathrm{free}(\varphi) \coloneqq \mathrm{free}_{\mathsf{v}}(\varphi) \cup \mathrm{free}_{\mathsf{e}}(\varphi)$ for all $\varphi \in \mathsf{GC}^k$. Formulas of GC^k are evaluated over a hypergraph $\mathcal{H}$ via interpretations $\mathcal{I} = (I_{\mathcal{H}}, \nu_{\mathsf{v}}, \nu_{\mathsf{e}})$ that consist of $\mathcal{H}$'s incidence graph $I_{\mathcal{H}}$ and assignments $\nu_{\mathsf{v}}\colon \mathsf{VAR}_{\mathsf{v}} \to \mathcal{R}(I_{\mathcal{H}})$ and $\nu_{\mathsf{e}}\colon \mathsf{VAR}_{\mathsf{e}} \to \mathcal{B}(I_{\mathcal{H}})$. The semantics of GC^k are as expected and a definition can be found in Section 6 of the full version of [27], thus we do not give one here. A *sentence* is a formula $\varphi \in \mathsf{GC}^k$ that has neither free vertex, nor free hyperedge variables, i.e., $\mathrm{free}(\varphi) = \varnothing$. By GC^k_d we denote the fragment $\{\varphi \in \mathsf{GC}^k : \mathrm{gd}(\varphi) \leq d\}$, and we let $\mathsf{GC}_k \coloneqq \mathsf{GC}^k_k$. We write $\mathcal{G} \equiv_{\mathsf{L}} \mathcal{H}$ to denote that $\mathcal{G}$ and $\mathcal{H}$ satisfy the same sentences in the fragment $\mathsf{L} \subseteq \mathsf{GC}^k$.

For simplicity, we omit logical guards if they are empty or equal to the formula they are guarding. I.e., we may abbreviate subformulas of the form $(\top \wedge \varphi)$ or $(\varphi \wedge \varphi)$ as φ. We may also omit parentheses in the usual way. We write $\exists^{=n}(\overline{\mathsf{x}})\,.\,(\Delta \wedge \varphi)$ as shorthand for $\exists^{\geq n}(\overline{\mathsf{x}})\,.\,(\Delta \wedge \varphi) \wedge \neg\exists^{\geq n+1}(\overline{\mathsf{x}})\,.\,(\Delta \wedge \varphi)$. Clearly, these shorthands change neither the semantics, nor the free variables, nor the guard depth of a formula.

▶ **Example 5.2.** The sentence $\varphi_{\mathcal{G}} \coloneqq \psi_1 \wedge \psi_2 \wedge \psi_3$ describes $\mathcal{G}$ from Example 2.1 up to isomorphism, where

$$\chi_n \coloneqq \bigwedge_{1\leq i<j\leq n} \neg \mathtt{v}_i = \mathtt{v}_j \ \wedge\ \neg\exists^{\geq 1}(\mathtt{v}_{n+1}) \,.\, \Big(E(\mathtt{e},\mathtt{v}_{n+1}) \ \wedge\ \bigwedge_{i\in[n]} \neg \mathtt{v}_{n+1} = \mathtt{v}_i\Big)\,,$$
$$\psi_1 \coloneqq \exists^{=4}(\mathtt{e}) \,.\, \mathtt{e}=\mathtt{e}\,,$$
$$\psi_2 \coloneqq \exists^{=1}(\mathtt{e}) \,.\, \exists^{\geq 1}(\mathtt{v}_1,\mathtt{v}_2,\mathtt{v}_3) \,.\, \Big(\bigwedge_{i\in[3]} E(\mathtt{e},\mathtt{v}_i) \ \wedge\ \chi_3 \ \wedge\ \bigwedge_{i\in[3]} \exists^{=3}(\mathtt{e}) \,.\, E(\mathtt{e},v_i)\Big)\,,$$
$$\psi_3 \coloneqq \exists^{=3}(\mathtt{e}) \,.\, \exists^{\geq 1}(\mathtt{v}_1,\mathtt{v}_2) \,.\, \Big(\bigwedge_{i\in[2]} E(\mathtt{e},\mathtt{v}_i) \ \wedge\ \chi_2 \ \wedge\ \bigwedge_{i\in[3]} \exists^{=3}(\mathtt{e}) \,.\, E(\mathtt{e},v_i)\Big)\,.$$

It is easily verified that $\varphi_{\mathcal{G}} \in \mathsf{GC}^1_2$. χ_n is a helper formula, describing that there are precisely n vertices $\mathtt{v}_1,\dots,\mathtt{v}_n$ in the hyperedge $\mathtt{e}$. ψ_1 describes that there are precisely four hyperedges, ψ_2 describes that precisely one hyperedge contains precisely three vertices, each being contained in precisely 3 hyperedges. Finally, ψ_3 describes that there are exactly 3 hyperedges containing precisely 2 vertices, each being contained in precisely 3 hyperedges. It is not hard to see that, in total, this describes $\mathcal{G}$ up to isomorphism.

Scheidt and Schweikardt [27] prove their result only for the following restricted variant of GC^k, called RGC^k. They mention in the conclusion, that RGC^k and GC^k are equivalent and show in the full version of the paper (Theorem 7.2) how a formula in GC^k can be translated into one in RGC^k. We still need RGC^k since it is used in the formulation and the proof of the two core lemmata of [27] that we want to borrow.

▶ **Definition 5.3** ([27])**.** The restriction RGC^k is inductively defined as follows:

Atomic Formulas. $(\Delta_g \wedge \varphi)$ is in RGC^k for all atomic formulas $\varphi \in \mathsf{GC}^k$ and all guard functions for φ, i.e., all $g\colon \mathbb{N}_{\geq 1} \rightharpoonup [k]$ with $\mathrm{dom}(g) = \{i : \mathtt{v}_i \in \mathrm{free}_{\mathtt{v}}(\varphi)\}$.

Inductive Rules.

- For every formula $(\Delta_g \wedge \varphi) \in \mathsf{RGC}^k$, the formula $(\Delta_g \wedge \neg\varphi)$ is also in RGC^k.
- For $i \in [2]$ and formulas $(\Delta_{g_i} \wedge \psi_i) \in \mathsf{RGC}^k$, the formula $(\Delta_{(g_1 \cup g_2)} \wedge (\psi_1 \wedge \psi_2))$ is in RGC^k, if g_1 and g_2 are compatible.

Let $n \in \mathbb{N}_{\geq 1}$, $(\Delta_g \wedge \varphi) \in \mathsf{RGC}^k$.

- For every $\mathtt{v}$-tuple $\overline{\mathtt{v}}$ with $\mathrm{vars}(\overline{\mathtt{v}}) \subseteq \mathrm{free}_{\mathtt{v}}(\varphi)$ and index set S, the formula $(\Delta_{\tilde{g}} \wedge \chi)$ is in RGC^k, where

  $$\chi \coloneqq \exists^{\geq n}\overline{\mathtt{v}} \,.\, (\Delta_g \wedge \varphi) \quad \text{and } \tilde{g} \text{ is the restriction of } g \text{ to } \mathrm{dom}(g) \setminus S.$$

- For every $\mathtt{e}$-tuple $\overline{\mathtt{e}}$ with $\mathrm{vars}(\overline{\mathtt{e}}) \subseteq \mathrm{free}_{\mathtt{e}}(\Delta_g \wedge \varphi)$ and index set S, the formula

  $$(\Delta_{\tilde{g}} \wedge \exists^{\geq n}\overline{\mathtt{e}} \,.\, (\Delta_g \wedge \varphi))$$

  is in RGC^k, if $\mathrm{dom}(\tilde{g}) = \mathrm{dom}(g)$ and all $i \in \mathrm{dom}(g)$ satisfy

  $$\tilde{g}(i) = g(i) \quad \text{or} \quad \tilde{g}(i) \in S \quad \text{or} \quad \tilde{g}(i) \notin \mathrm{img}(g). \tag{1}$$

Intuitively, formulas in RGC^k always carry the information, which hyperedge variable currently guards which vertex variable and the logical guards are in a certain sense "consistent" (1) along the syntax tree. ⌟

A simple inspection of the inductive proof for Theorem 7.2 in the full version of [27] shows that the guard depth is unaffected by the translation, thus it gives us the following refined result.

▶ **Lemma 5.4.** *For every formula $\varphi \in \mathsf{GC}^k$ and every guard function g for φ, there exists a formula $(\Delta_g \wedge \varphi_g) \in \mathsf{RGC}^k$ such that*

1. $(\Delta_g \wedge \varphi) \equiv (\Delta_g \wedge \varphi_g)$,
2. $\mathrm{free}(\varphi) = \mathrm{free}(\varphi_g)$, *and* $\mathrm{gd}(\varphi) = \mathrm{gd}(\varphi_g)$.

6 Main Result

We are now ready to plug everything together, which yields our main result.

▶ **Theorem 6.1.** *Let $\mathcal{G}$ and $\mathcal{H}$ be hypergraphs and let $k \in \mathbb{N}_{\geq 1}$.*

$$\begin{aligned} \mathcal{G} \equiv_{\mathsf{GC}_k} \mathcal{H} &\iff \mathrm{Hom}(\mathsf{ISHD}_k, I_{\mathcal{G}}) = \mathrm{Hom}(\mathsf{ISHD}_k, I_{\mathcal{H}}) \\ &\iff \mathrm{Hom}(\mathsf{SHD}_k, \mathcal{G}) = \mathrm{Hom}(\mathsf{SHD}_k, \mathcal{H}). \end{aligned}$$

We use the fact that the proofs for the core Lemmata 8.1 and 8.2 in the work by Scheidt and Schweikardt [27] actually give us the following refined results. This is easy to see on inspection of the original proofs (consult Appendix E in the full version of [27]), since there is a one-to-one correspondence between the blue label i and the hyperedge variable e_i in the proofs for both lemmas: whenever a blue label i is removed, the corresponding variable e_i is quantified and vice-versa.

For a k-labeled incidence graph L of the form (I, r, b, g), we let $\mathcal{I}_L := (I, \nu_{\mathsf{v}}, \nu_{\mathsf{e}})$ be defined by $\nu_{\mathsf{v}}(\mathsf{v}_i) := r(i)$ for all $i \in \mathrm{dom}(r)$ and $\nu_{\mathsf{e}}(\mathsf{e}_j) := b(j)$ for all $j \in \mathrm{dom}(b)$.

▶ **Lemma 6.2** (implicit in [27])**.** *Let $L = (I, r, b, g) \in \mathsf{GLI}_k^i$. For every $m \in \mathbb{N}$ there is a formula $\varphi_{L,m}$ with $(\Delta_g \wedge \varphi_{L,m}) \in \mathit{RGC}^k$, $\mathrm{free}_{\mathsf{v}}(\Delta_g \wedge \varphi_{L,m}) = \{\mathsf{v}_i \,:\, i \in \mathrm{dom}(r)\}$, $\mathrm{free}_{\mathsf{e}}(\Delta_g \wedge \varphi_{L,m}) = \{\mathsf{e}_j \,:\, j \in \mathrm{dom}(b)\}$, and $\mathrm{gd}(\varphi_{L,m}) \leq i$, such that for every k-labeled incidence graph L' with $\mathrm{dom}(b_{L'}) \supseteq \mathrm{dom}(b)$, $\mathrm{dom}(r_{L'}) \supseteq \mathrm{dom}(r)$, and with real guards w.r.t. g we have: $\mathcal{I}_{L'} \models \Delta_g$, and $\mathrm{hom}(L, L') = m \iff \mathcal{I}_{L'} \models \varphi_{L,m}$.*

▶ **Lemma 6.3** (implicit in [27])**.** *Let $\chi := (\Delta_g \wedge \psi) \in \mathit{RGC}^k$ with $\mathrm{gd}(\chi) = \ell$ and let $m, d \in \mathbb{N}$ with $m \geq 1$. There exists a linear combination $Q := \sum_{i \in [q]} \alpha_i L_i$, and sets $\mathrm{dr}_Q = \{i \,:\, \mathsf{v}_i \in \mathrm{free}_{\mathsf{v}}(\chi)\}$ and $\mathrm{db}_Q = \{i \,:\, \mathsf{e}_i \in \mathrm{free}_{\mathsf{e}}(\chi)\}$, where for all $i \in [q]$:*

$$\alpha_i \in \mathbb{R}, \quad L_i \in \mathsf{GLI}_k^{\ell}, \quad g_i = g, \quad \mathrm{dom}(b_i) = \mathrm{db}_Q, \quad \text{and} \quad \mathrm{dom}(r_i) = \mathrm{dr}_Q;$$

such that for all k-labeled incidence graphs L' with $|\mathcal{B}(I')| = m$, $\max\{|\,\beta(e)| \,:\, e \in \mathcal{B}(I')\} \leq d$ and $\mathrm{dom}(b') \supseteq \mathrm{db}_Q$, $\mathrm{dom}(r') \supseteq \mathrm{dr}_Q$, $g' \supseteq g$, and with real guards w.r.t. g we have: $\mathcal{I}_{L'} \models \Delta_g$, and

$$\sum_{i \in [q]} \alpha_i \cdot \mathrm{hom}(L_i, L') = \begin{cases} 1, & \text{if } \mathcal{I}_{L'} \models \chi \\ 0, & \text{if } \mathcal{I}_{L'} \not\models \chi. \end{cases}$$

The proof of Theorem 6.1 works the same way as the one for Theorem 6.1 in [27, Section 8]: the second biimplication is provided by Theorem 2.7 and Proposition 2.8. The first biimplication is shown via contraposition, where the contraposition of the forward direction uses Lemma 6.2 and the one for the backward direction uses Lemma 6.3.

Proof of Theorem 6.1. Let $I = I_{\mathcal{G}}$ and $J = I_{\mathcal{H}}$. If $|\mathcal{B}(I)| \neq |\mathcal{B}(J)|$ then $\mathrm{hom}(I', I) \neq \mathrm{hom}(I', J)$ for the incidence graph $I' \in \mathsf{ISHD}_1$ that consists of a single blue vertex and no red vertices. Similarly, I and J are distinguished by a suitable GC_1-sentence of the form $\exists^{\geq n} \mathsf{e} \,.\, (\mathsf{e} = \mathsf{e})$. If $|\mathcal{B}(I)| = |\mathcal{B}(J)|$, consider their corresponding label-free k-labeled incidence graphs $L_I = (I, \varnothing, \varnothing, \varnothing)$ and $L_J = (J, \varnothing, \varnothing, \varnothing)$.

Assume there is an $I' \in \mathsf{ISHD}_k$ such that $\mathrm{hom}(I', I) = m_1 \neq m_2 = \mathrm{hom}(I', J)$. According to Theorem 4.1, there is a label-free $L \in \mathsf{GLI}^k$ such that $I' \cong I_L$, which means $\mathrm{hom}(L, L_I) = m_1 \neq m_2 = \mathrm{hom}(L, L_J)$. By Lemma 6.2 there exists a formula $(\top \wedge \varphi_{L,m_1}) \in \mathsf{RGC}^k$ with $\mathrm{gd}(\varphi_{L,m_1}) \leq k$ such that $\mathcal{I}_{L_I} \models (\top \wedge \varphi_{L,m_1})$ and $\mathcal{I}_{L_J} \not\models (\top \wedge \varphi_{L,m_1})$. Hence, $\mathcal{I}_{L_I} \models \varphi_{L,m_1}$ and $\mathcal{I}_{L_J} \not\models \varphi_{L,m_1}$, and since $\varphi_{L,m_1} \in \mathsf{GC}_k$, $\mathcal{G} \not\equiv_{\mathsf{GC}_k} \mathcal{H}$.

Assume there is a sentence $\varphi \in \mathsf{GC}^k$ with $\mathrm{gd}(\varphi) = k$ such that $\mathcal{I}_{L_I} \models \varphi$ and $\mathcal{I}_{L_J} \not\models \varphi$. By Lemma 5.4 there exists a formula $(\top \wedge \psi) \in \mathsf{RGC}^k$ with $\mathrm{gd}(\psi) = k$ such that $\mathcal{I}_{L_I} \models (\top \wedge \psi)$ and $\mathcal{I}_{L_J} \not\models (\top \wedge \psi)$. Let $m \coloneqq |\mathcal{B}(I)| = |\mathcal{B}(J)|$ be the number of hyperedges and let $n \in \mathbb{N}$ such that $|\,\beta(e)| \leq n$ for all $e \in \mathcal{B}(I)$ and all $e \in \mathcal{B}(J)$. According to Lemma 6.3 there exists a linear combination $Q = \sum_{i\in[q]} \alpha_i L_i$ such that $\sum_{i\in[q]} \alpha_i \cdot \hom(L_i, L_I) = 1$ and $\sum_{i\in[q]} \alpha_i \cdot \hom(L_i, L_J) = 0$ and $L_i \in \mathsf{GLI}_k^k$ for all $i \in [q]$. This means there must be an $i \in [q]$ such that $\alpha_i \cdot \hom(L_i, L_I) \neq \alpha_i \cdot \hom(L_i, L_J)$, which means $\hom(L_i, L_I) \neq \hom(L_i, L_J)$. Since $\mathrm{dr}_Q = \mathrm{db}_Q = \varnothing$, L_i is label-free. According to Theorem 4.1, there exists an $I' \in \mathsf{ISHD}_k$ such that $I' \cong I_{L_i}$. Thus, $\hom(I', I) \neq \hom(I', J)$, i.e., $\mathrm{Hom}(\mathsf{ISHD}_k, I) \neq \mathrm{Hom}(\mathsf{ISHD}_k, J)$.

This finishes the proof for the first "iff". The second is provided by the combination of Theorem 2.7 and Proposition 2.8. ◀

7 Final Remarks

This paper solves one of the open questions of Scheidt and Schweikardt [27], who lift a result by Dvořák [9] from graphs to hypergraphs. Dvořák shows that homomorphism indistinguishability over the graphs of tree width at most k is equivalent to indistinguishability over first-order logic with counting quantifiers (C) and k+1 variables (C^{k+1}). Scheidt and Schweikardt show that homomorphism indistinguishability over the class GHW_k of hypergraphs of generalised hypertree width at most k is equivalent to indistinguishability over the logic GC with k guards (GC^k). Grohe [11] gave a result complementing Dvořák's: C with quantifier depth at most m (C_m) matches homomorphism indistinguishability over graphs of tree depth at most m. An obvious expectation was that the distinguishing power of GC_m would match homomorphism indistinguishability over the class HD_m of hypergraphs of hypertree depth at most m as it is defined by Adler et al. [1]. However, this expectation did not manifest in this exact way. Instead, we proved that the distinguishing power of GC_m matches homomorphism indistinguishability over hypergraphs of *strict* hypertree depth at most m, which is a (mild) restriction of hypertree depth. Combining Theorem 6.1 with the main result of [27] yields the following combined result.

▶ **Theorem 7.1.** *For all hypergraphs $\mathcal{G}$ and $\mathcal{H}$, the following equivalences hold:*

$$\mathcal{G} \equiv_{\mathsf{GC}_k} \mathcal{H} \iff \mathcal{G} \equiv_{\mathsf{SHD}_k} \mathcal{H} \iff I_{\mathcal{G}} \equiv_{\mathsf{ISHD}_k} I_{\mathcal{H}} \quad \text{and}$$
$$\mathcal{G} \equiv_{\mathsf{GC}^k} \mathcal{H} \iff \mathcal{G} \equiv_{\mathsf{GHW}_k} \mathcal{H} \iff I_{\mathcal{G}} \equiv_{\mathsf{IGHW}_k} I_{\mathcal{H}}.$$

We took this unexpected mismatch between GC_k and HD_k as an opportunity to investigate the relationship between HD_k and SHD_k. In Theorem 2.5 we showed that the strict hypertree depth of a hypergraph is at most 1 larger than its hypertree depth.

▶ **Theorem 2.5.** *For all hypergraphs $\mathcal{H}$,* $\mathrm{hd}(\mathcal{H}) \leq \mathrm{shd}(\mathcal{H}) \leq \mathrm{hd}(\mathcal{H})+1$.

To show that homomorphism counts from the class SHD_k are just as expressive as homomorphism counts from the class ISHD_k, which was necessary to prove Theorem 6.1, we used an implicit result by Böker [4], who gives a sufficient set of properties for a class $\mathfrak{C}$ of hypergraphs, such that homomorphism indistinguishability over $\mathfrak{C}$ is the same as homomorphism indistinguishability over the corresponding class $\mathfrak{C}_{\mathsf{I}}$ of incidence graphs. Since HD_k does not have these properties, Böker's result cannot be applied with respect to HD_k and IHD_k. In fact, we showed in Theorem 2.9 that homomorphism indistinguishability over HD_k is *not* the same as homomorphism indistinguishability over IHD_k and furthermore, that it is also not the same as homomorphism indistinguishability over SHD_k.

▶ **Theorem 2.9.** *For every $k \in \mathbb{N}_{\geq 1}$ there exist pairs of hypergraphs $(\mathcal{G}_k, \mathcal{H}_k)$ and $(\mathcal{G}'_k, \mathcal{H}'_k)$, such that:*

1. $\mathrm{Hom}(\mathsf{SHD}_k, \mathcal{G}_k) = \mathrm{Hom}(\mathsf{SHD}_k, \mathcal{H}_k)$, *but* $\mathrm{Hom}(\mathsf{HD}_k, \mathcal{G}_k) \neq \mathrm{Hom}(\mathsf{HD}_k, \mathcal{H}_k)$;
2. $\mathrm{Hom}(\mathsf{HD}_k, \mathcal{G}'_k) = \mathrm{Hom}(\mathsf{HD}_k, \mathcal{H}'_k)$, *but* $\mathrm{Hom}(\mathsf{IHD}_k, I_{\mathcal{G}'_k}) \neq \mathrm{Hom}(\mathsf{IHD}_k, I_{\mathcal{H}'_k})$.

Further Research. It would be very interesting to see if the result by Böker (Theorem 2.7) is tight in the sense that closure under pumping and local merging are sufficient *and required* properties. I.e., whether for every class $\mathfrak{C}$ that misses one of these properties, homomorphism counts over $\mathfrak{C}$ differ from homomorphism counts over the corresponding class $\mathfrak{C}_{\mathsf{I}}$ of incidence graphs in their distinguishing power.

As mentioned in the introduction, this work can be seen as one more step in the search of a "proper" lifting of the k-dimensional Weisfeiler-Leman algorithm to hypergraphs. Given the relationship between Weisfeiler-Leman, C and homomorphism indistinguishability on graphs [5, 7, 8, 9, 10, 11], we believe that the proper lifting should admit a similar relationship to the corresponding hypergraph parameters. Hence, we believe that the distinguishing power of such an algorithm should match homomorphism indistinguishability over the class GHW_k of hypergraphs of generalised hypertree width at most k and thus also indistinguishability by the logic GC^k. Since we believe that GC^k is the natural lifting of C^k in this setting, this paper adds to this picture: The k-dimensional Weisfeiler-Leman algorithm restricted to m iterations should have the same distinguishing power as the intersection of the classes $\mathsf{GHW}_k \cap \mathsf{SHD}_m$. Hence, the mismatch we uncovered in this work might propagate to the Weisfeiler-Leman algorithm.

References

1 Isolde Adler, Tomáš Gavenčiak, and Tereza Klimošová. Hypertree-depth and minors in hypergraphs. *Theoretical Computer Science*, 463:84–95, 2012. doi:10.1016/j.tcs.2012.09.007.

2 Jan Böker, Yijia Chen, Martin Grohe, and Gaurav Rattan. The Complexity of Homomorphism Indistinguishability. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2019.54.

3 Silvia Butti and Víctor Dalmau. Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2021.27.

4 Jan Böker. Color Refinement, Homomorphisms, and Hypergraphs. In Ignas Sau and Dimitrios M. Thilikos, editors, *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19-21, 2019, Revised Papers*, volume 11789 of *Lecture Notes in Computer Science*, pages 338–350. Springer, 2019. doi:10.1007/978-3-030-30786-8_26.

5 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

6 Bruno Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In Neil Robertson and Paul Seymour, editors, *Graph Structure Theory, Proceedings of a AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors held June 22 to July 5, 1991, at the University of Washington, Seattle, USA*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1993.

7 Anuj Dawar, Tomáš Jakl, and Luca Reggio. Lovász-type theorems and game comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

8 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.40.

9 Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, 2010. doi:10.1002/jgt.20461.

10 Eva Fluck, Tim Seppelt, and Gian Luca Spitzer. Going deep and going wide: Counting logic and homomorphism indistinguishability over graphs of bounded treedepth and treewidth. In Aniello Murano and Alexandra Silva, editors, *32nd EACSL Annual Conference on Computer Science Logic, CSL 2024, February 19-23, 2024, Naples, Italy*, volume 288 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2024.27.

11 Martin Grohe. Counting Bounded Tree Depth Homomorphisms. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 507–520, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394739.

12 Martin Grohe. Word2vec, Node2vec, Graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, pages 1–16. ACM, 2020. doi:10.1145/3375395.3387641.

13 Martin Grohe. The Logic of Graph Neural Networks. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–17. IEEE, 2021. doi:10.1109/LICS52264.2021.9470677.

14 Martin Grohe, Kristian Kersting, Martin Mladenov, and Erkal Selman. Dimension Reduction via Colour Refinement. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 505–516, Berlin, Heidelberg, 2014. Springer. doi:10.1007/978-3-662-44777-2_42.

15 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism tensors and linear equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.70.

16 Neil Immerman and Eric Lander. Describing Graphs: A First-Order Approach to Graph Canonization. In Alan L. Selman, editor, *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81. Springer, New York, NY, 1990. doi:10.1007/978-1-4612-4478-3_5.

17 Sandra Kiefer. The Weisfeiler-Leman Algorithm: An Exploration of Its Power. *ACM SIGLOG News*, 7(3):5–27, 2020. doi:10.1145/3436980.3436982.

18 László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18(3):321–328, 1967.

19 László Lovász and Balázs Szegedy. Contractors and connectors of graph algebras. *Journal of Graph Theory*, 60(1):11–30, 2009. doi:10.1002/jgt.20343.

20 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In Sandy Irani, editor, *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672. IEEE, 2020. doi:10.1109/FOCS46700.2020.00067.

21 Yoàv Montacute and Nihil Shah. The Pebble-Relation Comonad in Finite Model Theory. In Christel Baier and Dana Fisman, editors, *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, number 13 in LICS '22, pages 1–11, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533335.

22 Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33014602.

23 Daniel Neuen. Homomorphism-distinguishing closedness for graphs of bounded tree-width. *CoRR*, abs/2304.07011, 2023. doi:10.48550/arXiv.2304.07011.

24 Gaurav Rattan and Tim Seppelt. Weisfeiler-Leman and Graph Spectra. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 2268–2285. SIAM, 2023. doi:10.1137/1.9781611977554.ch87.

25 David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, 2022. doi:10.48550/arXiv.2206.10321.

26 Benjamin Scheidt. On Homomorphism Indistinguishability and Hypertree Depth, 2024. Full version. doi:10.48550/arXiv.2404.10637.

27 Benjamin Scheidt and Nicole Schweikardt. Counting Homomorphisms from Hypergraphs of Bounded Generalised Hypertree Width: A Logical Characterisation. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 79:1–79:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Full version available at arXiv: arXiv:2303.10980 [cs.LO]. doi:10.4230/LIPIcs.MFCS.2023.79.

28 Tim Seppelt. Logical equivalences, homomorphism indistinguishability, and forbidden minors. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.82.

29 Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011. URL: https://www.jmlr.org/papers/volume12/shervashidze11a/shervashidze11a.pdf, doi:10.5555/1953048.2078187.

30 Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.