



Finite Variable Counting Logics with Restricted Requantification

Simon Raßmann  

TU Darmstadt, Germany

Georg Schindling  

TU Darmstadt, Germany

Pascal Schweitzer  

TU Darmstadt, Germany

Abstract

Counting logics with a bounded number of variables form one of the central concepts in descriptive complexity theory. Although they restrict the number of variables that a formula can contain, the variables can be nested within scopes of quantified occurrences of themselves. In other words, the variables can be requantified. We study the fragments obtained from counting logics by restricting requantification for some but not necessarily all the variables.

Similar to the logics without limitation on requantification, we develop tools to investigate the restricted variants. Specifically, we introduce a bijective pebble game in which certain pebbles can only be placed once and for all, and a corresponding two-parametric family of Weisfeiler-Leman algorithms. We show close correspondences between the three concepts.

By using a suitable cops-and-robber game and adaptations of the Cai-Fürer-Immerman construction, we completely clarify the relative expressive power of the new logics.

We show that the restriction of requantification has beneficial algorithmic implications in terms of graph identification. Indeed, we argue that with regard to space complexity, non-requantifiable variables only incur an additive polynomial factor when testing for equivalence. In contrast, for all we know, requantifiable variables incur a multiplicative linear factor.

Finally, we observe that graphs of bounded tree-depth and 3-connected planar graphs can be identified using no, respectively, only a very limited number of requantifiable variables.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Complexity theory and logic

Keywords and phrases Requantification, Finite variable counting logics, Weisfeiler-Leman algorithm

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.14

Related Version *Full Version*: <https://arxiv.org/abs/2411.06944>

Funding The research of the second and third author leading to these results has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (EngageS: grant agreement No. 820148).

1 Introduction

Descriptive complexity is a branch of finite model theory that essentially aims at characterizing how difficult logical expressions need to be in order to capture particular complexity classes. While we are yet to find or rule out a logic capturing the languages in the complexity class P, there is an extensive body of work regarding the descriptive complexity of problems within P. Most notably, there is the work of Cai, Fürer, and Immerman [3] which studies a particular fragment of first-order logic. This is the fragment C^k in which counting quantifiers are introduced into the logic, but the number of variables is restricted to being at most k . The seminal result in [3] shows that this logic fails to define certain graphs up to isomorphism, which in turn proves that *inflationary fixed-point logic with counting* IFP+C fails to capture P.



© Simon Raßmann, Georg Schindling, and Pascal Schweitzer;
licensed under Creative Commons License CC-BY 4.0

33rd EACSL Annual Conference on Computer Science Logic (CSL 2025).

Editors: Jörg Endrullis and Sylvain Schmitz; Article No. 14; pp. 14:1–14:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Although the fragment C^k restricts the number of variables, it is common for variables to be reused within a single logical formula. In particular, variables can be nested within scopes of quantified occurrences of themselves. In other words, they can be requantified. In our work, we are interested in understanding what happens if we limit the ability to reuse variables through requantification. In fact, we may think of reusability as a resource (in the vein of time, space, communication, proof length, advice etc.) that should be employed economically.

It turns out that the ability to limit requantification provides us with a more detailed lens into the landscape of descriptive complexities within P, much in the fashion of fine-grained complexity theory.

Results and techniques. Let us denote by $C^{(k_1, k_2)}$ the fragment of first-order logic with counting quantifiers in which the formulas have at most k_1 variables that may be requantified and at most k_2 variables that may not be requantified.

First, we show that many of the traditional techniques of treating counting logics can be adapted to the setting of limited requantification. Specifically, it is well known that there is a close ternary correspondence between the logic C^k , the combinatorial bijective k -pebble game, and the famous $(k-1)$ -dimensional Weisfeiler-Leman algorithm [3, 22, 26]. We develop versions of the game and the algorithm that also have a limit on the reusability of resources. For the pebble game, a limit on requantification translates into pebbles that cannot be picked up anymore, once they have been placed. For the Weisfeiler-Leman algorithm, the limit on requantification translates into having some dimensions that “cannot be reused”. In fact the translation to the algorithmic viewpoint is not as straightforward as one might hope at first. Indeed, we do not know how to define a restricted version of the classical Weisfeiler-Leman algorithm that corresponds to the logic $C^{(k_1, k_2)}$. However, we circumvent this problem by employing the oblivious Weisfeiler-Leman algorithm (OWL). This variant is often used in the context of machine learning. In fact, Grohe [17] recently showed that $k+1$ -dimensional OWL is in fact exactly as powerful as k -dimensional (classical) WL. We develop a resource-reuse restricted version of the oblivious algorithm and prove equivalence to our logic. Indeed, we formally prove precisely matching correspondences between the limited requantification, limited pebble reusability, and the limited reusable dimensions (Theorem 6).

Next, we conclusively clarify the relation between the logics within the two-parametric family $C^{(k_1, k_2)}$. We show that in most cases limiting the requantifiability of a variable strictly reduces the power of the logic. We argue that no amount of requantification-restricted variables is sufficient to compensate the loss of an unrestricted variable. However, these statements are only true if at least some requantifiable variable remains. In fact, exceptionally, $C^{(1, k_2)}$ is strictly less expressive than $C^{(0, k'_2)}$ whenever $k'_2 > 2k_2$ (Theorem 13). To show the separation results, we adapt a construction of Fürer [13] and develop a cops-and-robber game similar to those in [12, 19]. In this version, some of the cops may repeatedly change their location, while others can only choose a location once and for all. Using another graph construction, we rule out various a priori tempting ideas concerning normal forms in $C^{(k_1, k_2)}$. To this end we show that formulas in the logics can essentially become as complicated as possible, having to repeatedly requantify all of the requantifiable variables an unbounded number of times, before using a non-requantifiable variable (Corollary 16). In terms of the pebble game, it seems a priori unclear when an optimal strategy would employ the non-reusable pebbles. However, the corollary says that in general one has to conserve the non-reusable pebbles for possibly many moves until a favorable position calls for them.

Having gone through the technical challenges that come with the introduction of reusability, puts us into a position to discuss the implications. Indeed, as our main result, we argue that our finer grained view on counting logics through restricted requantification has beneficial algorithmic implications. Specifically, we show that equivalence with respect to the logic $C^{(k_1, k_2)}$ can be decided in polynomial time with a space complexity of $O(n^{k_1} \log n)$, hiding quadratic factors depending only on k_1 and k_2 (Theorem 24). This shows that while the requantifiable variables each incur a multiplicative linear factor in required space, the restricted variables only incur an additive polynomial factor. In particular, equivalence with respect to the logics $C^{(0, k_2)}$ can be decided in logarithmic space. To show these statements, we leverage the fact that, because non-requantifiable variables cannot simultaneously occur free and bound, the $C^{(k_1, k_2)}$ -type of a variable assignment does not depend on the $C^{(k_1, k_2)}$ -type of assignments which disagree regarding non-requantifiable variables. Moreover, we use ideas from an algorithm of Lindell, which computes isomorphism of trees in logarithmic space [30] to implement the iteration steps of our algorithm. Generally, we believe the new viewpoint may be of interest in particular for applications in machine learning, where the WL-hierarchy appears to be too coarse for actual applications with graph neural networks (see for example [1, 2, 31, 40]). In the process of the space complexity proof, we also show that the iteration number of the resource-restricted Weisfeiler-Leman algorithm described above is at most $(k_2 + 1)n^{k_1} - 1$ (Corollary 19).

Justifying the new concepts of restricted reusability, we observe that there are interesting graph classes that are identified by the logics $C^{(k_1, k_2)}$. We argue that $C^{(0, d+1)}$ identifies all graphs of tree-depth at most d (Theorem 27) and that $C^{(2, 2)}$ identifies all 3-connected planar graphs (Theorem 33).

Outline of the paper. After briefly providing necessary preliminaries (Section 2) we formally introduce the logics $C^{(k_1, k_2)}$, the pebble game with non-reusable pebbles, the (k_1, k_2) -dimensional oblivious Weisfeiler-Leman algorithm, and prove the correspondence theorem between them (Section 3). We then relate the power of the logics to each other and rule out certain normal forms (Section 4). We then analyze the space complexity (Section 5) and finally provide two classes of graphs that are identified by our logics (Section 6).

Further related work. In addition to the references above, let us mention related investigations. Over time, a large body of work on descriptive complexity has evolved. For insights into fundamental results regarding bounded variable logics, we refer to classic texts [25, 26, 33, 34]. However, highlighting the importance of the counting logics C^k , let us at least mention the Immerman-Vardi theorem [24, 39]. It says that on ordered structures, *least fixed-point logic* LFP captures P. Since LFP has the same expressive power as IFP+C on ordered structures, also IFP+C, whose expressive power is closely related to the expressive power of the logics C^k , captures P. We should also mention the work of Hella [22] introducing the bijective k -pebble game which forms the basis for our resource restricted versions.

(Counting logics on graph classes) Because of the close correspondence between the logic C^k and the $(k - 1)$ -dimensional Weisfeiler-Leman algorithm, our investigations are closely related to the notion of the *Weisfeiler-Leman dimension* of a graph defined in [15]. Given a graph G this is the least number of variables k such that C^{k+1} identifies G . In particular, on every graph class of bounded Weisfeiler-Leman dimension, the corresponding finite variable counting logic captures isomorphism. Graph classes with bounded Weisfeiler-Leman dimension include graphs with a forbidden minor [14] and graphs of bounded rank-width (or equivalently clique width) [20], which in both cases is also shown to imply

that IFP+C captures P on these classes. For a comprehensive survey we refer to [27]. Our observations for planar graphs follow from techniques bounding the number of variables required for the identification of planar graphs [28]. Other recent classes not already captured by the results on excluded minors and rank-width include, for example, some polyhedral graphs [29], some strongly regular graphs [4], and permutation graphs [21].

(Logic and tree decompositions) In [9] and independently [8] it was shown that C^k -equivalence is characterized by homomorphism counts from graphs of tree-width at most $k - 1$. Likewise, homomorphism counts from bounded tree-depth graphs characterize equivalence in counting logic of bounded quantifier-rank [16]. Recently, these results were unified to characterize logical equivalence in finite variable counting logics with bounded quantifier-rank in terms of homomorphism counts [12].

(Space complexity) Ideas underlying Lindell’s logspace algorithm for tree isomorphism have also been used in the context of planar graphs [6] and more generally bounded genus graphs [10]. Similar results exist for classes of bounded tree-width [5, 11].

(Further recent results) Let us mention some quite recent results in the vicinity of our work that cannot be found in the surveys mentioned above. Regarding the quantifier-rank within counting logics, there is a recent superlinear lower bound [19] improving Fürer’s linear lower bound construction [13]. Further, very recent work on logics with counting includes results on rooted unranked trees [23] and inapproximability of questions on unique games [35]. Finally, there has been a surge in research on descriptive complexity within the context of machine learning (see [17, 36, 37]).

2 Preliminaries

General notation. For $n \in \mathbb{N}_+$ we use $[n]$ to denote the n -element set $\{1, \dots, n\}$. We use the notation $\{\{v_1, \dots, v_n\}\}$ for *multisets*. For $k_1, k_2 \in \mathbb{N}_+$, we fix the variable sets $[x_{k_1}] := \{x_1, \dots, x_{k_1}\}$, $[y_{k_2}] := \{y_1, \dots, y_{k_2}\}$, and $[x_{k_1}, y_{k_2}] := \{x_1, \dots, x_{k_1}, y_1, \dots, y_{k_2}\}$. Given a set V , a *partial function* $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V$ assigns to every variable $z \in [x_{k_1}, y_{k_2}]$ at most one element $\alpha(z) \in V$. If α does not assign an element to z , we write $\alpha(z) = \perp$. Also, we write $\text{im}(\alpha)$ for the *image* of α . With a finite set V and $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V$ we associate the total function $\bar{\alpha}: [x_{k_1}, y_{k_2}] \rightarrow V \cup \{\perp\}$, which we also view as a $[x_{k_1}, y_{k_2}]$ -indexed $(k_1 + k_2)$ -tuple. For $z \in [x_{k_1}, y_{k_2}]$ and $v \in V$, the function $\alpha[z/v]$ is defined as α but with $\alpha(z)$ replaced by v .

Graphs. A graph is a pair $G = (V(G), E(G))$ consisting of a finite set $V(G)$ of *vertices* and a set $E(G) \subseteq \binom{V(G)}{2}$ of *edges*. We write $|G|$ for the number of vertices, called the *order* of G . For a vertex $v \in V(G)$ we define the *neighborhood* $N_G(v) := \{w \in V(G) : \{v, w\} \in E(G)\}$ and the *degree* $d_G(v) := |N_G(v)|$ of v in G . We call v *universal* in G if $N_G(v) = V(G) \setminus \{v\}$. A *colored graph* consists of a graph G and a coloring function $\chi: V(G) \rightarrow C$ with a finite, ordered set C of *colors*. For a colored graph G and vertices $v_1, \dots, v_n \in V(G)$ the graph $G_{(v_1, \dots, v_n)}$ is obtained by assigning new and distinct colors to the vertices v_1, \dots, v_n in G . The vertices v_1, \dots, v_n are then called *individualized*. An isomorphism of (colored) graphs G and H is a bijection $\varphi: V(G) \rightarrow V(H)$ that preserves edges, non-edges, and vertex-colors.

We denote the complete graph on n vertices by K_n , that is, the graph with vertex set $[n]$ and all possible edges included. A star of degree n is a graph consisting of one universal vertex of degree n and its neighbors of degree 1.

First-order logic with counting. First-order logic with counting C is an extension of first-order logic by counting quantifiers $\exists^{\geq k}$ for all $k \in \mathbb{N}$. These intuitively state that there exist at least k distinct vertices satisfying the formula that follows.

Over the language of colored graphs with variable set \mathcal{V} , formulas are inductively built up from atomic formulas (for stating equality or adjacency of vertices as well as for stating that a vertex has a given vertex color) via negation (\neg), conjunction (\wedge), disjunction (\vee), implication (\rightarrow), and quantification over vertices via \forall , \exists and the counting quantifiers $\exists^{\geq k}$. For a colored graph G , a variable assignment $\alpha: \mathcal{V} \rightarrow V(G)$, and a formula $\varphi \in \mathcal{C}$, we write $G, \alpha \models \varphi$ if the graph G together with the variable assignment α *satisfies* the formula φ .

The *quantifier-rank* $\text{qr}(\varphi)$ of a formula φ is the maximum depth of nested quantifiers in the formula. The set of *free variables* $\text{free}(\varphi)$ of a formula φ is defined as the set of all variables that occur outside the scope of a corresponding quantifier in φ . If $\text{free}(\varphi) = \emptyset$ the formula φ is called a *sentence*. The set of *bound variables* $\text{bound}(\varphi)$ is the set of all variables that occur quantified in φ . If we restrict to a finite set of $k \in \mathbb{N}_+$ variables, the resulting logic is called *k-variable counting logic* and denoted by \mathcal{C}^k . For $r \in \mathbb{N}$ the *quantifier-rank-r counting logic* \mathcal{C}_r is obtained by restricting formulas in \mathcal{C} to quantifier-rank at most r . The *k-variable quantifier-rank-r counting logic* is defined as $\mathcal{C}_r^k := \mathcal{C}^k \cap \mathcal{C}_r$.

As a general reference on finite variable logics, we refer to [33].

The Weisfeiler-Leman algorithm. Let $k \geq 1$ and G be a colored graph. The *k-dimensional Weisfeiler-Leman algorithm* (short *k-WL*) iteratively computes a coloring of the k -tuples of vertices of G . We also view the k -tuples as total functions $\bar{\alpha}: \{x_1, \dots, x_k\} \rightarrow V(G)$. Initially, each tuple $\bar{\alpha} \in V(G)^k$ is colored by $\text{wl}_k^{(0)}(G, \bar{\alpha}) := \text{atp}_k(G, \bar{\alpha})$. Here, $\text{atp}_k(G, \bar{\alpha})$ is the *atomic type* of $\bar{\alpha}$ in G , i.e., the set of all atomic formulas with variables in $\{x_1, \dots, x_k\}$ satisfied by the colored graph $G[\text{im}(\alpha)]$ together with the assignment α . For every $r \in \mathbb{N}$, we then inductively set

$$\text{wl}_k^{(r+1)}(G, \bar{\alpha}) := (\text{wl}_k^{(r)}(G, \bar{\alpha}); \{\{\text{wl}_k^{(r)}(G, \bar{\alpha}[x_i/u])\}_{i \in [k]} : u \in V(G)\})$$

whenever $k \geq 2$, but for the case $k = 1$ we set

$$\text{wl}_k^{(r+1)}(G, \bar{\alpha}) := (\text{wl}_k^{(r)}(G, \bar{\alpha}); \{\text{wl}_k^{(r)}(G, u) : u \in N_G(\bar{\alpha})\}).$$

We write $\text{wl}_k^{(r)}(G)$ for the coloring of all k -tuples of vertices of G assigning $\text{wl}_k^{(r+1)}(G, \bar{\alpha})$ to $\bar{\alpha}$. Since the definition of $\text{wl}_k^{(r+1)}(G, \bar{\alpha})$ includes the color $\text{wl}_k^{(r)}(G, \bar{\alpha})$ of the previous iteration, the coloring $\text{wl}_k^{(r+1)}(G)$ *refines* the coloring $\text{wl}_k^{(r)}(G)$. That is, whenever $\text{wl}_k^{(r+1)}(G, \bar{\alpha}_1) = \text{wl}_k^{(r+1)}(G, \bar{\alpha}_2)$ for $\bar{\alpha}_1, \bar{\alpha}_2 \in V(G)^k$, then we also have $\text{wl}_k^{(r)}(G, \bar{\alpha}_1) = \text{wl}_k^{(r)}(G, \bar{\alpha}_2)$. Since there are exactly $|V(G)|^k$ -many k -tuples of vertices, there exists an $r < |V(G)|^k$ such that $\text{wl}_k^{(r)}(G)$ induces the same partition of color classes as $\text{wl}_k^{(r+1)}(G)$. It follows from the definition of the refinement that this implies $\text{wl}_k^{(r)}(G)$ induces the same partition as $\text{wl}_k^{(r')}(G)$ for all $r' \geq r$. In this case we say that *k-WL stabilizes* after at most r iterations on the graph G . If r is minimal with this property, we write $\text{wl}_k^{(\infty)}(G) := \text{wl}_k^{(r+1)}(G)$ and call $\text{wl}_k^{(\infty)}(G)$ the *stable coloring*. For a second colored graph H , we say that *k-WL distinguishes* G and H after r iterations if there exists a color c such that $|\{\bar{\alpha} \in V(G)^k : \text{wl}_k^{(r)}(G, \bar{\alpha}) = c\}| \neq |\{\bar{\beta} \in V(H)^k : \text{wl}_k^{(r)}(H, \bar{\beta}) = c\}|$.

Besides the classical k -dimensional Weisfeiler-Leman algorithm, there also exists a variant, called the *(k + 1)-dimensional oblivious Weisfeiler-Leman algorithm* (short *(k + 1)-OWL*). This variant colors $(k + 1)$ -tuples of vertices, which we again view as total functions $\bar{\alpha}: \{x_1, \dots, x_{k+1}\} \rightarrow V(G)$. Initially, all tuples are colored by their atomic type: $\text{owl}_{k+1}^{(0)}(G, \bar{\alpha}) := \text{atp}_{k+1}(G, \bar{\alpha})$. Then, this coloring is iteratively refined by setting

$$\text{owl}_{k+1}^{(r+1)}(G, \bar{\alpha}) := (\text{owl}_{k+1}^{(r)}(G, \bar{\alpha}); \{\{\text{owl}_{k+1}^{(r)}(G, \bar{\alpha}[x_i/u])\}_{i \in [k+1]}\}).$$

The stable coloring $\text{owl}_{k+1}^{(\infty)}(G)$ and the notion of distinguishing graphs is defined as for *k-WL*.

It turns out that k -WL and $(k + 1)$ -OWL have the same distinguishing power.

► **Lemma 1** ([17, Lemma A.1, Corollary V.7]). *Let G and H be graphs, $\bar{\alpha} \in V(G)^{k+1}$ and $\bar{\beta} \in V(H)^{k+1}$. Then the following are equivalent for every $r \in \mathbb{N}$:*

1. $\text{owl}_{k+1}^{(r)}(G, \bar{\alpha}) = \text{owl}_{k+1}^{(r)}(H, \bar{\beta})$,
2. $\text{atp}_{k+1}(G, \bar{\alpha}) = \text{atp}_{k+1}(H, \bar{\beta})$ and for all $i \in [k+1]$, we have $\text{wl}_k^{(r)}(G, \bar{\alpha}_{\neq i}) = \text{wl}_k^{(r)}(H, \bar{\beta}_{\neq i})$, where $\bar{\alpha}_{\neq i}$ is the k -tuple obtained from α by deleting the i -th entry.

Moreover, two graphs are distinguished by k -WL if and only if they are distinguished by $(k + 1)$ -OWL.

3 Finite Variable Counting Logics with Restricted Requantification

When working in the logic C^k , it is often necessary to *requantify* variables in order to express certain properties. We introduce finite variable first-order logic with counting quantifiers and *restricted quantification* to study this issue. We then define an Ehrenfeucht-Fraïssé-style game as an important tool for the analysis of the newly introduced logic by game-theoretic arguments. Finally, we devise a variant of k -OWL that precisely captures the expressive power of the logic and game and prove a characterization that closely ties the reusable and non-reusable resources among these objects. First, we give a precise definition of *requantification*.

► **Definition 2.** *Consider the counting logic C over a set of variables \mathcal{V} . A variable $x \in \mathcal{V}$ is said to be requantified in a formula $\varphi \in C$ if either $x \in \text{free}(\varphi) \cap \text{bound}(\varphi)$ or if there exist a subformula $Qx\psi$ of φ and in turn a subformula $Q'\chi$ of ψ with $Q, Q' \in \{\forall, \exists\} \cup \{\exists^{\geq n} : n \in \mathbb{N}\}$. We define the logic $C^{(k_1, k_2)}$ as the fragment of C over the fixed variable set $\mathcal{V} = [x_{k_1}, y_{k_2}]$ consisting of those formulas in which the variables from $\{y_1, \dots, y_{k_2}\}$ are not requantified. The fragment of $C^{(k_1, k_2)}$ with quantifier-rank at most $r \in \mathbb{N}$ is denoted by $C_r^{(k_1, k_2)}$.*

► **Example 3.** Consider the following $C_3^{(2,1)}$ formula:

$$(\exists y_1 \neg E(x_2, y_1)) \wedge \exists^{\geq 4} x_1 (E(x_2, x_1) \wedge \exists y_1 (\neg E(x_1, y_1)) \wedge \forall x_2 (\neg E(x_2, x_1) \rightarrow \exists^{\geq 3} x_1 E(x_1, x_2)))$$

expressing that the vertex x_2 is not universal and has at least four non-universal neighbors such that every non-neighbor of those has degree at least three. The variable x_2 is requantified in this formula since it occurs free and bound. The variable x_1 is requantified because the subformula $\exists^{\geq 3} x_1 E(x_1, x_2)$ occurs within the scope of the outermost quantification $\exists^{\geq 4} x_1$. The variable y_1 however is not requantified since neither of its quantifications occurs in the scope of the other.

The central question we will investigate in the following is how the non-requantifiability restriction affects the expressive power of the logic $C^{(k_1, k_2)}$. To this end, we use the notation $C^{(k_1, k_2)} \preceq C^{(k'_1, k'_2)}$ if every pair of graphs distinguished by $C^{(k_1, k_2)}$ is also distinguished by $C^{(k'_1, k'_2)}$. We also write $C^{(k_1, k_2)} \equiv C^{(k'_1, k'_2)}$ if the two logics distinguish exactly the same pairs of graphs and $C^{(k_1, k_2)} \prec C^{(k'_1, k'_2)}$ if the relation is strict. In the case of unrestricted quantification (i.e. $k_2 = 0$) it is clear that $C^{(k_1, 0)} \preceq C^{(k_1+1, 0)}$. In this terminology, the central result of [3] is that this relation is strict for all $k_1 \in \mathbb{N}$. For the case of restricted quantification we make the simple observation that having more variables is at least as expressive as having fewer variables (independent of their ability to be requantified). We also observe that requantifiable variables are at least as expressive as non-requantifiable variables. That is, for all $k_1, k_2 \in \mathbb{N}$ with $k_1 + k_2 \geq 1$ it holds that $C^{(k_1, k_2)} \preceq C^{(k_1, k_2+1)} \preceq C^{(k_1+1, k_2)}$.

Also, observe that having only non-requantifiable variables (i.e. $k_1 = 0$) bounds the quantifier-rank to at most k_2 and in turn every sentence of quantifier-rank at most k_2 can be rewritten using at most k_2 non-requantifiable variables. More precisely, we have $C^{(0,k_2)} \equiv C_{k_2}$.

Next, we establish an Ehrenfeucht-Fraïssé-style game which closely corresponds to the power of the previously defined logics with respect to distinguishing graphs. The game is a variant of the bijective pebble game introduced in [22] with the additional restriction that some pebbles may not be picked up again once placed.

► **Definition 4.** *Suppose $k_1, k_2 \in \mathbb{N}$ and $k_1 + k_2 \geq 1$. For colored graphs G and H , we define the bijective (k_1, k_2) -pebble game $\text{BP}_{(k_1, k_2)}(G, H)$ as follows:*

The game is played by the players Spoiler, denoted by (S) , and Duplicator, denoted by (D) , with one pair of pebbles for each variable in $[x_{k_1}, y_{k_2}]$. The pebble pairs in $[x_{k_1}]$ are called reusable and the pebble pairs in $[y_{k_2}]$ are called non-reusable.

The game proceeds in rounds, each of which is associated with a pair of partial functions $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V(G)$, $\beta: [x_{k_1}, y_{k_2}] \rightarrow V(H)$ with $\text{dom}(\alpha) = \text{dom}(\beta)$. We call such a pair of partial functions a (k_1, k_2) -configuration on the pair G, H . These functions indicate the placement of the pebble pairs on the graphs. For a pebble pair $z \in [x_{k_1}, y_{k_2}]$ and vertices $v \in V(G), w \in V(H)$ we have $\alpha(z) = v, \beta(z) = w$ whenever the two pebbles of the pair z are placed on v and w , respectively. If not specified otherwise, both games start from the empty configuration given by $\text{dom}(\alpha) = \text{dom}(\beta) = \emptyset$. One round of the game with current configuration (α, β) consists of the following steps:

1. *(S) picks up a pebble pair $z \in [x_{k_1}, y_{k_2}]$ such that $z \in [x_{k_1}]$ or $\alpha(z)$ is undefined. If no such z exists, the winning condition is checked directly.*
2. *(D) chooses a bijection $f: V(G) \rightarrow V(H)$.*
3. *(S) chooses $w \in V(G)$ and $f(w) \in V(H)$ to be pebbled with the pair z .*
4. *The new configuration is given by $(\alpha[z/w], \beta[z/f(w)])$.*

The winning conditions are as follows:

- *(S) wins immediately, if the initial configuration (α, β) does not induce a partial isomorphism. That is, the function $h: \text{im}(\alpha) \rightarrow \text{im}(\beta), \alpha(z) \mapsto \beta(z)$ is not a graph isomorphism from $G[\text{im}(\alpha)]$ to $H[\text{im}(\beta)]$.*
- *(S) wins if (D) cannot choose a bijection f , i.e., if $|G| \neq |H|$.*
- *(S) wins after the current round if the configuration (α, β) does not induce a partial isomorphism. Otherwise, the game continues and (D) wins the game if (S) never wins a round.*

For $r \in \mathbb{N}_+$ we define the game variant $\text{BP}_{(k_1, k_2)}^r$, which has the additional winning condition that (D) wins the game if (S) does not win after r rounds.

We now turn to devise an algorithmic counterpart of the logic $C^{(k_1, k_2)}$ and the game $\text{BP}_{(k_1, k_2)}$. It is an adaptation of the oblivious Weisfeiler-Leman algorithm k -OWL.

Indeed, to capture $C^{(k_1, k_2)}$ -equivalence, we iteratively color (partial) $(k_1 + k_2)$ -tuples of vertices of a given graph with the previous color, and a sequence of multisets corresponding to variables as in k_1 -OWL. We deviate from the classical oblivious Weisfeiler-Leman algorithm by treating some entries of the tuple as *non-reusable*: For $y \in [y_{k_2}]$ with $\alpha(y) \neq \perp$, the variable y is already assigned in the logic, respectively the non-reusable pebble is already placed in the game. Thus, the entry in α corresponding to this variable should not be replaced by other vertices, but be kept fixed. For this reason we utilize the advantage of oblivious Weisfeiler-Leman that each multiset corresponds to exactly one variable and pebble pair respectively.

Recall that we can view a variable assignment $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V(G)$ for a graph G as a $[x_{k_1}, y_{k_2}]$ -indexed $(k_1 + k_2)$ -tuple over $V(G) \dot{\cup} \{\perp\}$, which we denote by $\bar{\alpha}$. For a variable assignment α , we set $J(\alpha) := \{j \in [k_2] : \bar{\alpha}(y_j) = \perp\}$.

► **Definition 5.** Let G be a graph and $k_1, k_2 \in \mathbb{N}$ with $k_1 + k_2 \geq 1$. The (k_1, k_2) -dimensional oblivious Weisfeiler-Leman algorithm (short (k_1, k_2) -OWL) iteratively computes a coloring of $[x_{k_1}, y_{k_2}]$ -indexed $(k_1 + k_2)$ -tuples over $V(G) \dot{\cup} \{\perp\}$.

Initially, each tuple $\bar{\alpha}$ is colored by its atomic type in G : $\text{owl}_{(k_1, k_2)}^{(0)}(G, \bar{\alpha}) := \text{atp}_{k_1 + k_2}(G, \bar{\alpha})$. This coloring is then refined recursively: for every $r \in \mathbb{N}$, we define

$$\text{owl}_{(k_1, k_2)}^{(r+1)}(G, \bar{\alpha}) := (\text{owl}_{(k_1, k_2)}^{(r)}(G, \bar{\alpha}), \{\{\text{owl}_{(k_1, k_2)}^{(r)}(G, \bar{\alpha}[x_i/w]) : w \in V(G)\}_{i \in [k_1]}, \{\{\text{owl}_{(k_1, k_2)}^{(r)}(G, \bar{\alpha}[y_j/w]) : w \in V(G)\}_{j \in J(\alpha)}\})$$

Just as in the classical case, the coloring $\text{owl}_{(k_1, k_2)}^{(r+1)}(G)$ refines $\text{owl}_{(k_1, k_2)}^{(r)}(G)$ and eventually stabilizes. We denote the stable coloring by $\text{owl}_{(k_1, k_2)}^{(\infty)}(G)$.

The correspondence of counting logic, pebble game, and algorithm for restricted reusability now is as follows:

► **Theorem 6.** Let G, H be colored graphs and $k_1, k_2 \in \mathbb{N}$ with $k_1 + k_2 \geq 1$. Then for all (k_1, k_2) -configurations (α, β) and $r \in \mathbb{N}$ the following are equivalent:

1. For every $\varphi \in \mathcal{C}_r^{(k_1, k_2)}$ with $\text{free}(\varphi) \subseteq \text{dom}(\alpha)$ and $\text{free}(\varphi) \cap [y_{k_2}] = \text{dom}(\alpha) \cap [y_{k_2}]$ it holds that $G, \alpha \models \varphi \Leftrightarrow H, \beta \models \varphi$.
2. (D) has a winning strategy for $\text{BP}_{(k_1, k_2)}^r(G, H)$ with initial configuration (α, β) .
3. It holds that $\text{owl}_{(k_1, k_2)}^{(r)}(G, \bar{\alpha}) = \text{owl}_{(k_1, k_2)}^{(r)}(H, \bar{\beta})$.

The theorem can be proved by carefully adapting the proof of [3, Theorem 5.2] by treating non-requantifiable variables separately.

4 The Role of Reusability

We investigate the interplay of quantifiable and non-quantifiable variables in $\mathcal{C}^{(k_1, k_2)}$ using the game-theoretic characterization provided by Theorem 6. To this end, we utilize the CFI construction from [3] in the variant employed in [13]. The construction starts from a so-called *base graph*, that is, a connected and colored graph such that every vertex receives a unique natural number as color. By our convention, the coloring induces a linear ordering on the vertices of the base graph. The vertices and edges of the base graph are called *base vertices* and *base edges*, respectively. From a base graph G the *CFI graph* $X(G)$ is constructed by replacing each base vertex in G by a *gadget* consisting of *gadget vertices* but not edges. Gadgets corresponding to adjacent base vertices are then connected by adding edges between gadget vertices. To *twist* a base edge $\{u, v\} \in E$ in $X(G)$ means to replace every edge between the corresponding gadgets by a non-edge and every non-edge by an edge. The *twisted CFI graph* $\tilde{X}(G)$ is obtained by twisting an arbitrary base edge $e \in E(G)$ in $X(G)$.

We introduce a variant of the cops-and-robber game used in [19] to simulate the game $\text{BP}_{(k_1, k_2)}$ on CFI graphs via a game played only on the base graph. Our variant involves non-reusable cops as a way of restricting reusability of resources.

► **Definition 7.** The cops-and-robber game $\text{CR}_{(k_1, k_2)}(G)$ is played on a base graph G between a group of $k_1 + k_2$ cops and one robber. The cops are denoted by the elements of $[x_{k_1}, y_{k_2}]$ and a cop x_i is called reusable while a cop y_j is called non-reusable. Each round of the game is associated with a partial function $\gamma: [x_{k_1}, y_{k_2}] \rightarrow V(G)$ and an edge $e \in E(G)$. The function γ encodes the current positions of the cops while the edge e is the position of the robber. Initially, there are no cops on the vertices and the robber is placed on some edge of the base graph. One round of the game with current position (γ, e) consists of the following steps:

1. The cops choose $z \in [x_{k_1}, y_{k_2}]$ such that $z \in [x_{k_1}]$ or $\gamma(z)$ is undefined. If no such z exists, the winning condition is checked directly. Then a destination $w \in V(G)$ for z is declared.
2. The robber chooses an edge e' in the connected component of e in $G - \text{im}(\gamma[z/\perp])$.
3. The cop z is placed on the vertex w .
4. The new position of the game is given by $(\gamma[z/w], e')$.

The winning condition is as follows:

- The cops win the game if at the end of the current round both vertices incident to the robber edge e' hold cops. The robber wins if the cops never win.

We also introduce the game $\text{CR}_{(k_1, k_2)}^r(G)$ with the additional winning condition that the robber wins if the cops do not win in r rounds.

Intuitively, in the game $\text{BP}_{(k_1, k_2)}(X(G), \tilde{X}(G))$ Spoiler has to catch the twist in $\tilde{X}(G)$ with pebbles to show the difference of the graphs. This corresponds to moving the cops (according to the reusability of the used pebbles) in $\text{CR}_{(k_1, k_2)}(G)$. Duplicator, however, moves the twist in $\tilde{X}(G)$ using automorphisms of the graph to hide the difference, which corresponds to moving the robber in $\text{CR}_{(k_1, k_2)}(G)$. Following similar arguments from [7, 13], this yields the following lemma, stating that the bijective pebble game on CFI graphs can be simulated appropriately.

► **Lemma 8.** Let $k_1 + k_2 \geq 2$ and $r \in \mathbb{N}$. Then the robber has a winning strategy in $\text{CR}_{(k_1, k_2)}^r(G)$ if and only if (D) has a winning strategy in $\text{BP}_{(k_1, k_2)}^r(X(G), \tilde{X}(G))$.

We now prove a strict hierarchy for the logics $\mathcal{C}^{(k_1, k_2)}$ by providing, for every pair of logics we want to separate, two CFI graphs $X(G), \tilde{X}(G)$ that are distinguished by one of the logics, but not the other. To show this, it now suffices to provide strategies for the game $\text{CR}_{(k_1, k_2)}(G)$ by Lemma 8 and Theorem 6. The idea for the choice of the base graphs G is inspired by [13] where grid graphs were chosen as base graphs.

► **Definition 9.** The graph

$$G_{h \times \ell} := (\{v_{i,j} : i \in [h], j \in [\ell]\}, \{\{v_{i,j}, v_{r,s}\} : |i - r| + |j - s| = 1\})$$

is called the grid graph with h rows and ℓ columns. We also call h the height and ℓ the length of the grid. We say that the cops build a barrier in the game $\text{CR}_{(k_1, k_2)}$ played on a graph G containing a grid if they are placed on a separator of the graph G disconnecting the first column from the last column of the grid.

First, we show the advantages of reusability: When the cops-and-robber game is played on the grid graph $G_{h \times \ell}$ with at least $h + 1$ cops, the cops can be placed on a column to form a barrier and move it through the grid maintaining this formation by using the additional cop. To show the separation, we choose the base graph as a grid of sufficient length such that the cops are required to move or build a barrier repeatedly. This makes reusability necessary as all non-reusable cops are placed at some point and cannot be used to move a barrier any further.

14:10 Finite Variable Counting Logics with Restricted Requantification

► **Lemma 10.** *For all $k_1, k_2, k'_1, k'_2 \geq 0$, if $k_1 > \max(k'_1, 1)$, then $C^{(k_1, k_2)} \not\leq C^{(k'_1, k'_2)}$.*

Proof. First, consider the game $CR_{(k_1, k_2)}(G_{(k_1-1) \times (k_1 2^{2k'_2+1})})$. The cops can build a barrier in the middle of the grid and move it towards the robber only using reusable cops. This is a winning strategy for the cops since the size of the component containing the robber is decreased by a constant in each round and eventually vanishes. On the other hand, we show that the robber has a winning strategy in the game $CR_{(k'_1, k'_2)}(G_{(k_1-1) \times (k_1 2^{2k'_2+1})})$ by induction on k'_2 . The base case for $k'_2 = 0$ is the game $CR_{(k'_1, 0)}(G_{(k_1-1) \times (k_1+1)})$, for which the robber has a winning strategy as a barrier can be built, but not moved. For the inductive step assume $k'_2 > 0$ and consider the game $CR_{(k'_1, k'_2+1)}(G_{(k_1-1) \times (k_1 2^{2k'_2+2})})$. Using only reusable cops, the cops can reduce the size of the robber component with a barrier. However, the size remains at least $(k_1 - 1) \cdot (k_1 2^{2k'_2+1} + 1)$, since the robber can choose the larger induced component. When a reusable cop is reused before a non-reusable cop was used to build another barrier, the reusable barrier breaks down and the robber as an escape strategy. Using non-reusable cops, the cops can build another wall to reduce the size of the robber component to $(k_1 - 1) \cdot (k_1 2^{2k'_2} + 1)$. Again, the robber chooses the larger induced component. But now the remaining game is $CR_{(k'_1, k'_2-k_1+2)}(G_{(k_1-1) \times (k_1 2^{2k'_2+1})})$ and we have $k'_2 \geq k'_2 - k_1 + 2$. Thus, by the inductive hypothesis the robber has a winning strategy for the remaining game. ◀

Second, we show the advantages of mere capacity: When the base graph is chosen as a complete graph, the robber can choose any edge independently of the choice of vertices by the cops and the only possibility to win for the cops is to have sufficient capacity. In this case, capacity is more valuable than reusability.

► **Lemma 11.** *For all $k_1, k_2, k'_1, k'_2 \geq 0$, if $k_1 + k_2 > k'_1 + k'_2$, then $C^{(k_1, k_2)} \not\leq C^{(k'_1, k'_2)}$.*

Proof. In the game $CR_{(k_1, k_2)}(K_{k_1+k_2})$, the cops have a winning strategy just by covering all base vertices. In contrast, in the game $CR_{(k'_1, k'_2)}(K_{k_1+k_2})$ the robber has a winning strategy. Whenever a cop is picked up there is one edge that is not incident to a cop and thus yields a safe escape for the robber. ◀

Third, we treat the special case of a single requantifiable variable: Intuitively, at least two reusable cops are needed to move a barrier for an arbitrarily large distance in a base graph. When only one single reusable cop is available, the distance that can be covered by a barrier of cops is bounded by $2k_1 + 1$ because for every other move a non-reusable cop must be used. The *perfect binary tree of depth d* is the binary tree B^d such that all interior vertices have two children and all leaves have the same depth.

► **Lemma 12.** *For all $k_2, k'_2 \geq 1$ it holds that $C^{(1, k_2)} \not\leq C^{(0, k'_2)}$ if and only if $k'_2 \leq 2k_2$.*

Proof. In the game $CR_{(1, k_2)}(B^{2k_2})$, the cops have a winning strategy by alternately using non-reusable cops and the reusable cop. In the game $CR_{(0, 2k_2)}(B^{2k_2})$ the robber has a winning strategy as the non-reusable cops are exhausted before the robber is caught. This yields $C^{(1, k_2)} \not\leq C^{(0, 2k_2)}$. For $k'_2 \leq 2k_2$ we get $C^{(1, k_2)} \not\leq C^{(0, k'_2)}$ since the robber wins with the same strategy in $CR_{(0, k'_2)}(B^{2k_2})$. For $k'_2 > 2k_2$, let (S) have a winning strategy for $BP_{(1, k_2)}(G, H)$. Then (S) has a winning strategy for $BP_{(1, k_2)}^{2k_2+1}(G, H)$ since consecutive moves involving the pebble pair x_1 can be replaced by a single move instead. The winning strategy for (S) in $BP_{(1, k_2)}^{2k_2+1}(G, H)$ directly yields a winning strategy for (S) in $BP_{(0, k'_2)}(G, H)$: For every pebble pair played by (S) in $BP_{(1, k_2)}^{2k_2+1}(G, H)$, the player (S) can use a new pair in $BP_{(0, k'_2)}(G, H)$. ◀

With the previous lemmas we can determine the relation of the logics $C^{(k_1, k_2)}$ and $C^{(k'_1, k'_2)}$ for any given combination of parameters.

► **Theorem 13.** *For all $k_1, k_2 \in \mathbb{N}$ and $k'_1, k'_2 \in \mathbb{N}$ with $k_1 + k_2, k'_1 + k'_2 \geq 2$ it holds that $C^{(k_1, k_2)} \prec C^{(k'_1, k'_2)}$ if and only if one of the following assertions holds:*

1. $k_1 < k'_1$ and $k_1 + k_2 \leq k'_1 + k'_2$,
2. $k_1 \leq k'_1$ and $k_1 + k_2 < k'_1 + k'_2$, or
3. $k_1 = 1, k'_1 = 0$, and $k'_2 > 2k_2$.

Furthermore, it holds that $C^{(k_1, k_2)} \equiv C^{(k'_1, k'_2)}$ if and only if $(k_1, k_2) = (k'_1, k'_2)$.

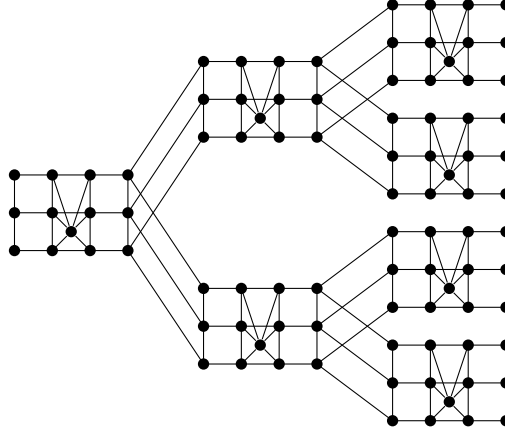
This settles the question of how the use of non-requantifiable variables affects the expressive power of the logic. For the investigation of the logic $C^{(k_1, k_2)}$ for fixed parameters, it is also of interest how the non-requantifiable variables behave in concrete formulas of the logic. This relates closely to asking whether there are normal forms for the logic $C^{(k_1, k_2)}$ with respect to reusability. We give a precise answer to this question that rules out many such normal forms. The idea is to construct a new family of base graphs that allow the use of non-reusable cops only after all reusable cops have been used a certain number of times.

► **Definition 14.** *We construct the graph $\dot{G}_{h \times \ell}$ from the grid graph $G_{h \times \ell}$ by adding one additional vertex b , which we call bridge vertex, and the edges $\{\{v_{i, \lfloor \frac{\ell}{2} \rfloor}, b\} : i \in [h]\} \cup \{\{b, v_{i, \lfloor \frac{\ell}{2} \rfloor + 1}\} : i \in [h]\}$ to $G_{h \times \ell}$. Using this modified grid, we define the following base graph:*

For $\ell \geq 2, h \geq 1$ and $d \geq 1$ we obtain the graph $B_{h \times \ell}^d$ by replacing every vertex of a perfect binary tree B^d of depth d by a grid $\dot{G}_{h \times \ell}$ and connect adjacent grids row-wise (see Figure 1).

► **Theorem 15.** *For all $k_1, k_2 \geq 1$ and $r \geq 1$ there exist graphs G and H such that (S) has a winning strategy for $\text{BP}_{(k_1, k_2)}(G, H)$ and in every winning strategy (S) must reuse every reusable pebble pair at least r times (once if $k_1 = 1$) before using a new non-reusable pebble pair.*

Proof. For $k_1 > 1$, we consider the game $\text{CR}_{(k_1, k_2)}(B_{(k_1-1) \times 2r}^{k_2+1})$, see Figure 1. The cops have the following winning strategy: First, they build a barrier in the root grid (behind the bridge vertex) by occupying one full column using k_1 reusable cops. The barrier can then be moved towards the robber using the additional reusable cop. When the barrier reaches the last column, the two subtrees induced by the children of the root grid are disconnected components with respect to the cops. Thus, the robber has to choose an edge in one of these components to escape to. The cops move the barrier into the corresponding subtree, which essentially results in the game $\text{CR}_{(k_1, k_2)}(B_{(k_1-1) \times 2r}^{k_2})$. In every grid of the tree, the cops encounter a bridge vertex that can be covered by one of the k_2 non-reusable cops. The game continues inductively for $\Omega(rk_2)$ rounds, until the cops use the last remaining non-reusable cop to cover the bridge vertex in a leaf grid. The barrier can be moved to the end of the grid and the robber will be caught. Now assume a new non-reusable cop y has been used before all reusable cops have been (re)used r times at some point of the game. Then the barrier was not moved out of the current grid at this point, since all reusable cops have to be moved at least r times to achieve this. Hence, the robber has not chosen a new subtree so far and can pick an edge in a subtree that does not contain y . The cops need to move the barrier into that subtree and use non-reusable cops for the bridge vertices. Since y was used in another subtree, at some point there will be no non-reusable cops left to cover a bridge vertex and the barrier cannot be moved further without breaking down. Thus, the robber can escape indefinitely. For $k_1 = 1$ we consider the game $\text{CR}_{(1, k_2)}(B^{2k_2})$. The cops have the following winning strategy: First, the reusable cop x_1 is placed in the root node. This disconnects



■ **Figure 1** A drawing of the base graph $B_{3 \times 4}^3$ for Theorem 15.

the subtrees induced by the two children of the root node for the robber, and the robber has to choose an edge in one of the subtrees of depth $2k_2 - 1$. Accordingly, a non-reusable cop y_1 is placed on the node inducing that subtree, which again disconnects two subtrees of depth $2k_2 - 2$. The cop x_1 can be picked up again from the root node to be placed on the corresponding subtree. Inductively, the cops alternately use non-reusable cops y_j and the reusable cop x_1 to cover the next child node. After $2k_2$ moves, the induced subtree is of depth 0 and the robber is caught in the edge to a leaf node. If two non-reusable cops are used consecutively, similar to the case $k_1 > 1$, there are no non-reusable cops left at depth $2k_2 - 1$ and the remaining reusable cop does not suffice to catch the robber. ◀

Again using Theorem 6 this result translates into the language of logic as follows:

► **Corollary 16.** *For all $k_1, k_2 \geq 1$ and $r \geq 1$ there exist graphs G, H such that G and H are not $\mathcal{C}^{(k_1, k_2)}$ -equivalent and for every formula $\varphi \in \mathcal{C}^{(k_1, k_2)}$ that distinguishes G and H the following holds: There exists a sequence of subformulas $\exists^{\geq n_1} y_{j_1} \psi_1, \dots, \exists^{\geq n_{k_2}} y_{j_{k_2}} \psi_{k_2}$ of φ such that $\text{qr}(\psi_{k_2}) = k_1$ and for $\ell \in [k_2 - 1]$ the formula $\psi_{\ell+1}$ is a subformula of ψ_ℓ with $\text{qr}(\psi_\ell) \geq \text{qr}(\psi_{\ell+1}) + k_1 r$ if $k_1 > 1$ and $\text{qr}(\psi_\ell) \geq \text{qr}(\psi_{\ell+1}) + 1$ if $k_1 = 1$. Moreover, between the quantifications $\exists^{\geq n_\ell} y_{j_\ell} \psi_\ell$ and $\exists^{\geq n_{\ell+1}} y_{j_{\ell+1}} \psi_{\ell+1}$ all requantifiable variables have to be requantified r times (once if $k_1 = 1$) in ψ_ℓ .*

The necessity of this pattern of (re)quantification rules out various normal forms with respect to requantification for $\mathcal{C}^{(k_1, k_2)}$ one might have hoped to have. In particular, it is not sufficient to quantify all non-requantifiable variables directly one after the other.

Regarding classical logics without restricted requantification, Theorem 13 and Corollary 16 yield that for $k_1, k_2 \geq 1$ the power of $\mathcal{C}^{(k_1, k_2)}$ to distinguish graphs is not identical to that of \mathcal{C}^k , \mathcal{C}_r , or \mathcal{C}_r^k for all $k, r \in \mathbb{N}$.

5 Space Complexity

In this section we investigate the space complexity of deciding whether two given graphs are $\mathcal{C}^{(k_1, k_2)}$ -equivalent. In principle, this can be achieved by testing $\text{owl}_{(k_1, k_2)}^{(\infty)}(G) = \text{owl}_{(k_1, k_2)}^{(\infty)}(H)$ by Theorem 6. However, a naive implementation of (k_1, k_2) -OWL requires space $\Omega(n^{k_1+k_2})$ and hence provides no improvement compared to the situation with unrestricted reusability. We seek to improve the space complexity to $O(n^{k_1} \log n)$ when both requantifiable and non-requantifiable variables are involved. Here the O notation hides factors depending on k_1 and k_2 but not on n .

To achieve this, we observe that the color $\text{owl}_{(k_1, k_2)}^{(r)}(G, \bar{\alpha})$ only depends on the colors $\text{owl}_{(k_1, k_2)}^{(s)}(G, \bar{\beta})$ with $s < r$ where $\beta|_{[y_{k_2}]}$ is an extension of $\alpha|_{[y_{k_2}]}$. This allows us to compute these colorings, while only ever remembering colors of assignments with a few distinct $[y_{k_2}]$ -parts. Moreover, we show that the (k_1, k_2) -dimensional oblivious Weisfeiler-Leman algorithm can equivalently be implemented by alternatingly refining with respect to the reusable dimensions until the coloring stabilizes, and refining with respect to the first unassigned non-quantifiable variable. We use this to show that the iteration number of (k_1, k_2) -OWL is at most $(k_2 + 1)n^{k_1} - 1$.

► **Definition 17.** For colorings χ and χ' on a set S , we say that χ refines χ' , written $\chi \preceq \chi'$, if every χ' -color class is a union of χ -color classes. If $\chi \preceq \chi'$ and $\chi \succeq \chi'$, we write $\chi \equiv \chi'$.

In the context of colorings, it is natural to understand the oblivious Weisfeiler-Leman algorithm as a *refinement operator*, i.e., a function that maps every coloring to a refined coloring. To make this formal, we define for every coloring χ on assignments $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V(G)$ the OWL-refinement

$$\text{owl-ref}_{(k_1, k_2)}(\chi)(\alpha) := \left(\chi(\alpha), \left\{ \left\{ \chi(\alpha[x_i/w]) : w \in V(G) \right\}_{i \in [k_1]}, \right. \right. \\ \left. \left. \left\{ \left\{ \chi(\alpha[y_j/w]) : w \in V(G) \right\}_{j \in J(\alpha)} \right\} \right).$$

This refinement is precisely the refinement that is applied by OWL in each iteration. In particular, applying it r times to the initial coloring by atomic types yields precisely the r -round OWL-coloring. That is,

$$\text{owl-ref}_{(k_1, k_2)}^{(r)}(\text{atp}_{k_1+k_2}(G)) = \text{owl}_{(k_1, k_2)}^{(r)}(G).$$

Note that OWL-refinement is *monotone* in the sense that for all colorings χ and χ' with the property $\chi \preceq \chi'$ it also holds that $\text{owl-ref}_{(k_1, k_2)}(\chi) \preceq \text{owl-ref}_{(k_1, k_2)}(\chi')$.

In order to space-efficiently deal with these refinements, we want to separate the refinements with respect to reusable dimensions from those with respect to non-reusable dimensions. To do this, note that the definition of $\text{owl-ref}_{(k_1, k_2)}$ still makes sense for colorings assignments $\alpha: [x_{k'_1}, y_{k'_2}] \rightarrow V(G)$ for $k'_1 \geq k_1$ or $k'_2 \geq k_2$, where the refinement just refines with respect to some but not all of the dimensions.

Moreover, in order to handle the distinguishing power of (k_1, k_2) -OWL on two different graphs, we note that we can also simultaneously apply these refinement operators to colorings on assignments over two different graphs.

With this terminology at hand, we can clarify the intuition we may gain from Section 4 regarding the employment of non-reusability. That is, in order to distinguish graphs, it suffices to alternatingly refine with respect to all quantifiable variables and a non-quantifiable variable.

► **Lemma 18.** For all $k_1 + k_2 \geq 1$, we have

$$\text{owl}_{(k_1, k_2)}^{(\infty)}(G) \equiv (\text{owl-ref}_{(k_1, 0)}^{(\infty)} \circ \text{owl-ref}_{(0, k_2)}^{(k_2)}) (\text{owl}_{(k_1, 0)}^{(\infty)}(G))$$

Proof. Because $\text{owl}_{(k_1, k_2)}$ is a finer refinement than $\text{owl-ref}_{(k_1, 0)}$ and than $\text{owl-ref}_{(0, k_2)}$, the direction \preceq is immediate. For the other direction, set

$$\chi_r := \left(\text{owl-ref}_{(k_1, 0)}^{(\infty)} \circ \text{owl-ref}_{(0, k_2)} \right)^{(r)} \left(\text{owl}_{(k_1, 0)}^{(\infty)}(G) \right).$$

We use the bijective pebble game and show, by induction on r , that for every (k_1, k_2) -configuration (α, β) over G with $|\text{dom}(\alpha) \cap [y_{k_2}]| = k_2 - r$ such that α and β have equal χ_r -colors, (D) has a winning strategy in the game $\text{BP}_{(k_1, k_2)}(G, G)$ with initial position (α, β) . This then implies the equality of owl-ref $_{(k_1, k_2)}^{(\infty)}(\chi)$ -colors.

For $r = 0$, the colors are precisely the colors computed by (k_1, k_2) -OWL. Thus, (D) has a winning strategy by Theorem 6.

Now, assume the claim is true for some r and let (α, β) be a (k_1, k_2) -configuration with $|\text{dom}(\alpha) \cap [y_{k_2}]| = k_2 - (r + 1)$ such that α and β have equal χ_{r+1} -colors. By the construction of $(k_1, 0)$ -OWL refinement, (D) can preserve the equality of χ_{r+1} -colors as long as (S) picks up reusable pebble pairs. When (S) picks up a non-reusable pebble pair, (D) can play such that the resulting positions have the same χ_r -colors. But then, (D) has a winning strategy by the induction hypothesis. \blacktriangleleft

Because classical owl $_{(k_1, 0)}$ -refinement stabilizes after at most $n^{k_1} - 1$ rounds, this scheme yields an upper bound on the iteration number of the oblivious Weisfeiler-Leman algorithm.

► **Corollary 19.** *The sequence of colorings owl $_{(k_1, k_2)}^{(r)}$ computed on a graph G stabilizes after at most $(k_2 + 1)n^{k_1} - 1$ rounds.*

We will now turn to the computation of the OWL-colorings. Because the names of the OWL-colors consist of nested multisets, they can become exponentially long. The usual way to deal with this is to either replace after each iteration round all color names by numbers of logarithmic length, or to not compute the colors at all but only consider the order on the variable assignments induced by the lexicographic ordering of their OWL-colors. We will switch between these two viewpoints depending on suitability to the task at hand. Accordingly, we use two different encodings of the colorings we deal with. Consider a coloring $\chi: M \rightarrow C$ and an order \leq on the set of colors C . We say that an algorithm is given *oracle access to the ordering of χ -colors* if the algorithm has access to a function that, given two elements $m, m' \in M$, returns whether $\chi(m) \leq \chi(m')$. For the second way that our algorithms interact with colorings, we call a coloring $\chi': M \rightarrow [|M|]$ a *normalization of χ* if for all $m, m' \in M$ we have $\chi(m) \leq \chi(m')$ if and only if $\chi'(m) \leq \chi'(m')$. Now, we say that an algorithm is given a *function table for χ* if for some normalization χ' of χ the algorithm is given an array A with $A[m] = \chi'(m)$ for all $m \in M$ suitably encoded as numbers in $[|M|]$. Similarly, we say that an algorithm *computes a function table for χ* if it outputs such an array. Note that a function table can be stored in space $|M| \cdot \lceil \log_2 |M| \rceil \in O(|M| \log |M|)$.

The main technical tool needed for the implementation of owl-ref $_{(k_1, k_2)}$ -refinements, is the ability to compare multisets of previously computed colors when given oracle access to a function comparing these previous colors.

► **Lemma 20.** *Given a natural number n in unary, oracle access to a total order \leq on $[n]$, and two multisets M and M' on $[n]$ of order at most n , the lexicographic order of M and M' can be decided in logarithmic space using quadratic time. Also, the lexicographical order of tuples of colors can be computed in logarithmic space.*

Proof. Consider two multisets $M = \{s_1, \dots, s_n\}$ and $M' = \{s'_1, \dots, s'_n\}$. Note that we have enough space to store a constant number of elements of $[n]$.

For a number $i \in [n]$, we denote the number of occurrences of i in M or M' by $M(i)$ and $M'(i)$ respectively. Note that these numbers can be computed in logarithmic space and linear time by simply comparing i to all elements in either set.

We start by finding the minimal element m_1 of M and m'_1 of M' . If $m_1 \neq m'_1$, we return their order. Otherwise, if $M(m_1) \neq M'(m_1)$, we return this order.

Thus, assume $m_1 = m'_1$ and that they occur in both multisets the same number of times. Next, we find the second-smallest elements m_2 and m'_2 of both sets, and can now forget about m_1 and m'_1 . We again compare m_2 and m'_2 and their number of occurrences and possibly return the order accordingly. Iteratively, we only need to remember the i -th smallest elements to find the $(i + 1)$ -th smallest elements, and we iteratively compare these elements and their number of occurrences. ◀

This allows us to compute the order of owl-ref $_{(k_1, k_2)}(\chi)$ -colors in logarithmic space when we are given oracle access to the order of χ -colors. Using the bound on the iteration number of (k_1, k_2) -OWL from Corollary 19, and the fact that we can perform one iteration using only logarithmic additional space, we immediately obtain an algorithm that can compare owl $_{(k_1, k_2)}^{(\infty)}$ -colors using space at most $O(n^{k_1} \log n)$, where we again dropped multiplicative factors depending on k_1 and k_2 . However, this naive implementation will not run in polynomial time. Indeed, because there are already $n^{k_1 + k_2}$ many variable assignments, we do not have enough space to store even the (order of) colors computed in the previous round. Instead, this naive algorithm recomputes polynomially many previous colors in every step, which leads to a polynomially branching algorithm with exponential running time.

To remedy this, we make full use of the scheme from Lemma 18. While performing owl $_{(k_1, 0)}$ -refinements, we are able to store a function table with the previously computed colors for all assignments with the same $[y_{k_2}]$ -part. This allows us to perform a full owl $_{(k_1, 0)}^{(\infty)}$ -refinement in polynomial time and the required space. Only when performing one of the k_2 many owl $_{(0, k_2)}$ -refinement steps do we need to consider variable assignments with different $[y_{k_2}]$ -parts. In this latter case, we cannot circumvent needing to compute colors for these assignments polynomially many times. While this does again lead to a polynomially branching algorithm, the depth of this branching is bounded by k_2 , which leads to a polynomial running time increase of n^{k_2} .

For a fixed assignment $\eta: [y_{k_2}] \rightarrow V(G)$, we denote by $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_\eta$ the set of assignments $\alpha: [x_{k_1}, y_{k_2}] \rightarrow V(G)$ whose $[y_{k_2}]$ -part is η . Because we only ever need to compare the colors of two assignments at a time, it will always be sufficient to compute the OWL-coloring on sets of the form

$$[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}$$

for a $(0, k_2)$ -configuration (η_G, η_H) over G and H . When restricting ourselves to assignments in such a set, the coloring computed by $(k_1, 0)$ -OWL can be computed as usual:

► **Lemma 21.** *Let $k_1 + k_2 \geq 1$, G, H be graphs and (η_G, η_H) be a $(0, k_2)$ -configuration over G, H . Given a function table for a coloring χ on $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}$, a function table for owl-ref $_{(k_1, 0)}(\chi)$ can be computed in time $n^{O(k_1)}$ and space $O(k_1 n^{k_1} \log n)$.*

Proof. Note that $|[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}| = 2(n + 1)^{k_1}$, which means that we can store a function tables for χ and owl-ref $_{(k_1, 0)}(\chi)$ in space

$$O(2(n + 1)^{k_1} \cdot \log(2(n + 1)^{k_1})) = O(k_1 n^{k_1} \log n).$$

In addition to these two function tables, we will only need logarithmic space.

In order to compute the function table for owl-ref $_{(k_1, 0)}(\chi)$, we need to refine the coloring χ with respect to the multisets

$$\{\{\chi(\alpha[x_i/w]) : w \in V(G)\}\} \quad \text{or} \quad \{\{\chi(\alpha[x_i/w]) : w \in V(H)\}\}$$

for all $i \in [k_1]$. By using the function table for χ as an oracle, we can compare these multisets in logarithmic additional space using Lemma 20.

14:16 Finite Variable Counting Logics with Restricted Requantification

This allows us to compare owl-ref_(k₁,0)(χ)-colors in the required space. Now, we simply start to compare each variable assignment α with all other assignments and count the number of assignments whose color is less than or equal to α. Then, we use this count as the new color of α and insert it into our function table. ◀

By applying Lemma 21 repeatedly until the coloring stabilizes, and only ever storing the function table from the previous and current iteration round we get the following:

► **Corollary 22.** *Let $k_1 + k_2 \geq 1$, G and H be graphs, and (η_G, η_H) be a $(0, k_2)$ -configuration over G and H . Given a function table for a coloring χ on $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}$, a function table for owl-ref_(k₁,0)^(∞)(χ) can be computed in time $n^{O(k_1)}$ space $O(k_1 n^{k_1} \log n)$.*

Now, we turn to refinements with respect to non-requantifiable variables.

► **Lemma 23.** *Let $k_1 + k_2 \geq 1$, G and H be graphs, and χ a coloring on $[[x_{k_1}, y_{k_2}] \rightarrow V(G)] \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]$.*

Given oracle access to the order of χ-colors, we can compute for every $(0, k_2)$ -configuration (η_G, η_H) the function table of owl-ref_(0,k₂)(χ) on $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}$ using space $O(k_1 n^{k_1} \log n + k_2 \log n)$ and time $n^{O(k_1)}$.

Proof. Note that we have enough space to hold the function table. In addition, we will only need logarithmic space.

Using Lemma 20, we can compute the lexicographic ordering of owl-ref_(0,k₂)(χ)-colors with logarithmic additional space. We can then compute the function table by assigning to each assignment α as the new color the number of assignments β in $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_G} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(H)]_{\eta_H}$ such that

$$\text{owl-ref}_{(0,k_2)}(\chi)(\beta) \leq_{\text{lex}} \text{owl-ref}_{(0,k_2)}(\chi)(\alpha),$$

which is a number in $[2(n+1)^{k_1}]$. ◀

Together, these two statements allow us to compare the colors computed by the (k_1, k_2) -dimensional oblivious Weisfeiler-Leman algorithm in a time- and space-efficient manner.

► **Theorem 24.** *Let $k_1 + k_2 \geq 1$ be fixed. For all (k_1, k_2) -configurations (α, β) over graphs G and H , we can decide whether $G, \alpha_1 \equiv_{\mathcal{C}^{(k_1, k_2)}} H, \alpha_2$ using space $O(k_1(k_2 + 1)n^{k_1} \log n + (k_2)^2 \log n)$ and polynomial time.*

Proof. By Lemma 18, we have

$$\text{owl}_{(k_1, k_2)}^{(\infty)}(G) \equiv \left(\text{owl-ref}_{(k_1, 0)}^{(\infty)} \circ \text{owl-ref}_{(0, k_2)} \right)^{(k_2)} \left(\text{owl}_{(k_1, 0)}^{(\infty)}(G) \right).$$

and similarly for H . We show by induction on r that we can compute for every pair of graphs $G_1, G_2 \in \{G, H\}$ and every $(0, k_2)$ -configuration (η_1, η_2) over G_1 and G_2 , a function table of

$$\chi_r := \left(\text{owl-ref}_{(k_1, 0)}^{(\infty)} \circ \text{owl-ref}_{(0, k_2)} \right)^{(r)} \left(\text{owl}_{(k_1, 0)}^{(\infty)} \right)$$

on $[[x_{k_1}, y_{k_2}] \rightarrow V(G_1)]_{\eta_1} \dot{\cup} [[x_{k_1}, y_{k_2}] \rightarrow V(G_2)]_{\eta_2}$ using time $n^{O((k_1+1)(r+1))}$ and space $O(k_1(r+1)n^{k_1} \log n + k_2(r+1) \log n)$, where $\text{owl}_{(k_1, 0)}^{(\infty)}(G_1, G_2)$ is the common coloring computed by (k_1, k_2) -OWL on both G_1 and G_2 .

If $r = 0$, we only need to compute the classical OWL-coloring. To do this, we first note that we can compute a function table listing the atomic types of assignments, each encoded as numbers in $[2(n+1)^{k_1}]$. Then, the claim follows from Corollary 22.

For the induction step, assume we can compute function tables for (restrictions of) the coloring χ_r for every fixed $(0, k_2)$ -configuration (η_1, η_2) over G_1 and G_2 in the required time and space. This in particular implies that we can compute the order of χ_r -colors of arbitrary assignments in time $n^{O((k_1+1)(r+1))}$ and space $O((r+1)(k_1 n^{k_1} \log n + k_2 \log n))$.

For every fixed $(0, k_2)$ -configuration (η_1, η_2) , we can thus compute a function table for the refined coloring $\text{owl-ref}_{(0, k_2)}(\chi_r)$ on $[[x_{k_1}, y_{k_2}] \rightarrow V(G)]_{\eta_1, \eta_2}$ using space $O((r+2)(k_1 n^{k_1} \log n + k_2 \log n))$ by Lemma 23. Because the algorithm from Lemma 23 runs in time $n^{O(k_1)}$, it can make at most $n^{O(k_1)}$ comparisons of previously computed colors, which means that this step takes time at most $n^{O(k_1)} \cdot n^{O((k_1+1)r)} = n^{O((k_1+1)(r+2))}$.

Using Corollary 22, we can refine this to a function table of

$$\chi_{r+1} = \text{owl-ref}_{(k_1, 0)}^{(\infty)} \circ \text{owl-ref}_{(0, k_2)}(\chi_r)$$

For $r = k_2$, this yields the claim. ◀

6 Graphs Identified by Logics with Restricted Requantification

We finally give two classes of graphs where very few reusable variables already suffice for identification. We say that a logic *identifies* a graph G if it distinguishes G from every non-isomorphic graph.

First, we show that the logic $C^{(0, d+1)}$ identifies all graphs of tree-depth at most d . Second, we refine previous work in which it was shown that $C^{(4, 0)}$ identifies all planar graphs [28]. By closely inspecting the arguments, we show that already $C^{(2, 2)}$ suffices to identify all 3-connected planar graphs.

Graphs of bounded tree-depth

Tree-depth is a graph parameter that intuitively measures how close a graph is to a star [32]. Let G be a graph and let G_1, \dots, G_p be the connected components of G . Then the *tree-depth* of G is inductively defined as

$$\text{td}(G) := \begin{cases} 1 & \text{if } |G| = 1 \\ 1 + \min_{v \in V(G)} \text{td}(G - v) & \text{if } p = 1 \text{ and } |G| > 1 \\ \max_{i \in [p]} \text{td}(G_i) & \text{otherwise.} \end{cases}$$

Note that a graph has tree-depth 1 if and only if it is an independent set, and tree-depth 2 if and only if it is a disjoint union of isolated vertices and stars. Intuitively, graphs of bounded tree-depth are those which, by repeatedly deleting one vertex in each connected component, can be eliminated in a bounded number of rounds. We start by showing how to simulate this deletion of vertices by pebbling them instead.

► **Definition 25.** *Let G be a graph with vertex coloring χ_G and let $v \in V(G)$. We define the colored graph $G \setminus v$ as the graph $G - v$ with the new coloring $\chi_{G \setminus v}$ defined by setting*

$$\chi_{G \setminus v}(w) := \begin{cases} (\chi_G(w), 1) & \text{if } \{v, w\} \in E(G) \\ (\chi_G(w), 0) & \text{if } \{v, w\} \notin E(G) \end{cases}$$

for all $w \in V(G) \setminus \{v\}$.

14:18 Finite Variable Counting Logics with Restricted Requantification

► **Lemma 26** ([16, Lemma 4.5]). *Let $k_2 \geq 2$, G, H be graphs with $|G| = |H| \geq 2$ and $v \in V(G), w \in V(H)$ with $\chi_G(v) = \chi_H(w)$. Then the following are equivalent:*

1. *(D) has a winning strategy for $\text{BP}_{(0, k_2)}(G, H)$ with initial position given by $\alpha(y_1) = v, \beta(y_1) = w$ and $\text{dom}(\alpha) = \text{dom}(\beta) = \{y_1\}$.*
2. *(D) has a winning strategy for $\text{BP}_{(0, k_2-1)}(G \wr v, H \wr w)$.*

We can now prove our result on identification of graphs of bounded tree-depth:

► **Theorem 27.** *For all $d \geq 1$, the logic $C^{(0, d+1)}$ identifies all colored graphs of tree-depth at most d .*

Proof. For two graphs G and C , denote by $\text{noc}(G, C)$ the number of connected components of G that are isomorphic to C . Using the equivalence of the logic and bijective pebble game, it suffices to prove the following claim, which lends itself better to an inductive proof:

▷ **Claim 28.** Let G and H be colored graphs, and C a connected, colored graph of tree-depth at most k_2 . If $\text{noc}(G, C) \neq \text{noc}(H, C)$, then (S) has a winning strategy for the game $\text{BP}_{(0, k_2+1)}(G, H)$.

Proof. We argue by induction on k_2 . If $k_2 = 1$, then C is a single vertex. Thus, G and H differ in the number of isolated vertices of some specific color, which allows (S) to win in 2 rounds.

For the induction step, assume that the claim is true for k_2 . Now, consider a graph C with $\text{td}(C) \leq k_2 + 1$ and assume w.l.o.g. that $\text{noc}(G, C) > \text{noc}(H, C)$. By the definition of tree-depth, C contains a vertex c such that $\text{td}(C - c) \leq k_2$. Let s be the number of such vertices.

We call a vertex v in either G or H *C-shrinking* if it is contained in a connected component C_v isomorphic to C , and $\text{td}(C_v - v) \leq k_2$. The number of *C-shrinking* vertices in G and H is $s \cdot \text{noc}(G, C)$ and $s \cdot \text{noc}(H, C)$ respectively. In particular, G contains more *C-shrinking* vertices than H .

Now, we describe the winning strategy for (S) in the game $\text{BP}_{(0, k_2+2)}(G, H)$. First, (S) picks up the (unused) pebble pair y_1 , and (D) picks a bijection $f: V(G) \rightarrow V(H)$. Then there exist some *C-shrinking* vertex $v \in V(G)$ such that its image $f(v)$ is not *C-shrinking* in H . Then (S) places the pebble pair on these two vertices. By Lemma 26, it now suffices to argue that (D) has a winning strategy for the game $\text{BP}_{(0, k_2+1)}(G \wr v, H \wr f(v))$. For this, let C_v be the connected component of v in G , and consider the connected components $C_v^{(1)}, \dots, C_v^{(\ell)}$ of $C_v \wr v \subseteq G \wr v$. If for some $i \in [\ell]$, we have $\text{noc}(G \wr v, C_v^{(i)}) \neq \text{noc}(H \wr f(v), C_v^{(i)})$, then we are done by the induction hypothesis. Thus, we are left with the case that $\text{noc}(G \wr v, C_v^{(i)}) = \text{noc}(H \wr f(v), C_v^{(i)})$. Note that the vertex-colorings of $G \wr v$ and $H \wr f(v)$ ensure that all connected components isomorphic to $C_v^{(i)}$ for some i are incident to v or $f(v)$ respectively. Thus, all copies of $C_v^{(i)}$ in G lie in C_v .

In this case, there is an isomorphism φ between the subgraphs induced by $G \wr v$ and $H \wr f(v)$ on the union of connected components isomorphic to $C_v^{(i)}$ for some $i \in [\ell]$. The vertex-colorings of $G \wr v$ and $H \wr f(v)$ further ensure that φ can be extended by $\varphi(v) := f(v)$, so that its domain is all of C_v . Thus, φ now embeds C_v into the connected component of $f(v)$, which might, however, have additional vertices attached to $f(v)$. If the image of C_v under φ was the whole connected component of $f(v)$, then $f(v)$ would be *C-shrinking*, which contradicts our assumption. Thus, there are additional vertices attached to $f(v)$. Thus, v and $f(v)$ have distinct degrees, which allows (S) to win in one further round. ◀

The theorem now follows by applying the claim to all connected components of G . ◀

Note that using Theorem 24 on the space complexity of $C^{(k_1, k_2)}$ -equivalence, the theorem in particular reproves the statement that isomorphism of graphs of bounded tree-depth can be decided in logarithmic space [5].

Note moreover that because $C^{(1,1)}$ can count the number of connected components isomorphic to a fixed colored star, the above inductive proof also shows that for $d \geq 2$, the logic $C^{(1, d-1)}$ identifies all colored graphs of tree-depth at most d .

3-connected planar graphs

The next class of graphs we consider are 3-connected planar graphs. These naturally appear in the proof that C^4 identifies all planar graphs, which starts by reducing the claim for arbitrary planar graphs to 3-connected planar graphs via the decomposition into triconnected components [28]. We recall their proof that C^4 identifies every 3-connected planar graph and show that it actually already yields that $C^{(2,2)}$ suffices.

The underlying technical lemma is the following:

► **Lemma 29** ([28, Lemma 23]). *Let G be a 3-connected planar graph and let $v_1, v_2, v_3 \in V(G)$. If v_1, v_2, v_3 lie on a common face of G , then $wl_1^{(\infty)}(G_{(v_1, v_2, v_3)})$ is a discrete coloring.*

In the classical setting, from this lemma one can obtain that 4-WL, i.e., C^5 identifies all 3-connected planar graphs. We make use of our framework and show that instead it suffices to use non-reusable resources to cover the individualized vertices.

► **Corollary 30** (compare [28, Corollary 24]). *The logic $C^{(2,3)}$ identifies all 3-connected planar graphs.*

Proof. Let G be a 3-connected planar graph. By Lemma 29, there are $v_1, v_2, v_3 \in V(G)$ such that $wl_1^{(\infty)}(G_{(v_1, v_2, v_3)})$ is a discrete coloring. Then by Lemma 1, also $owl_{(2,0)}^{(\infty)}(G_{(v_1, v_2, v_3)})$ is a discrete coloring, which implies that $G_{(v_1, v_2, v_3)}$ is identified by $C^{(2,0)}$. Thus, there exists a formula $\varphi(y_1, y_2, y_3) \in C^{(2,3)}$ which defines the graph G with three individualized vertices represented by y_1, y_2 and y_3 up to isomorphism. But then, the formula $\exists y_1 \exists y_2 \exists y_3 \varphi(y_1, y_2, y_3)$ identifies G . ◀

In order to improve the identification result from C^5 to C^4 and from $C^{(2,3)}$ to $C^{(2,2)}$, one observes that for almost all 3-connected graphs, the individualization of just 2 vertices already suffices for the above claim, and the exceptions, where indeed, 3 vertices are necessary are somewhat rare.

► **Definition 31.** *A 3-connected planar graph is called an exception if there are no two vertices $v_1, v_2 \in V(G)$ such that $wl_1^{(\infty)}(G_{(v_1, v_2)})$ is discrete.*

The crucial tool in lowering the number of variables needed is an explicit classification of all exceptions [28]. This allows to prove the following:

► **Lemma 32.** *All exceptions are identified by $C^{(2,2)}$.*

Proof. The exceptions are the following:

- all bipyramids, i.e., cycles with two additional non-adjacent but otherwise universal vertices,
- all platonic solids besides the dodecahedron, and the rhombic dodecahedron,
- the triakis tetrahedron, the tetrakis hexahedron and the triakis octahedron, i.e., the graphs obtained from the tetrahedron, the hexahedron and the octahedron by adding one vertex per face, whose neighborhood consists of the vertices on that face.

Because even color refinement identifies every graph with at most 5 vertices, and the bipyramid of order 6 is the octahedron, we start with bipyramids of order at least 7. We individualize two adjacent vertices of the cycle. Then, color refinement computes a discrete coloring on the underlying cycle, while the two pyramid tips get the same color, which is, however, distinct from all other colors. This graph is identified by color refinement.

Next, consider the platonic solids and the rhombic dodecahedron. All of these are distance-regular graphs of diameter at most 4, with at most 14 vertices. Note that for every $d \in \mathbb{N}$, there exists a formula $\varphi_d(y_1, y_2) \in \mathcal{C}^{(2,2)}$ stating that y_1 and y_2 have distance d . This implies that in $\mathcal{C}^{(2,2)}$ we can express that a graph G is distance-regular with a given parameter set. Because every distance-regular graph of order at most 14 is determined by its parameters [38], $\mathcal{C}^{(2,2)}$ identifies all platonic solids and the rhombic dodecahedron.

For the last case, note that the vertices of the platonic solid and the added vertices for every face have distinct degrees. As moreover, adding these vertices does not change the distance between any two original vertices, $\mathcal{C}^{(2,2)}$ can still express that the underlying platonic solid is of the correct type. Additionally, we can express that the neighborhood of each added face vertex is a cycle of the correct length, and that no two face vertices share more than 2 common neighbors. This identifies the last class of exceptions. ◀

This finally allows us to prove identification by $\mathcal{C}^{(2,2)}$:

► **Theorem 33.** *Every 3-connected planar graph is identified by $\mathcal{C}^{(2,2)}$.*

Proof. Let G be a 3-connected planar graph. If G is an exception, this follows from Lemma 32. If G is not an exception, the claim can be proven as in Corollary 30, where we instead only need to individualize 2 instead of 3 vertices. ◀

It is unclear how precisely this result generalizes to all planar graphs. Moreover, it is known that all graphs of Euler genus g are identified by \mathcal{C}^{4g+4} [18], and we would expect that also here, for sufficiently connected graphs only a very small number of the variables must be requantified.

7 Outlook

In this work, we establish a refined framework for the logical description of graphs by means of the requantification of variables. We indicate some open questions for future work in vastly different directions.

Towards structural graph theory, the newly defined cops-and-robber game $\text{CR}^{(k_1, k_2)}$ defines a two-parametric family of graph classes, which contain the tree-width and tree-depth graphs as subclasses. It will be interesting to obtain graph-theoretic characterizations for these classes and to study them from an algorithmic and logical point of view.

From a practical viewpoint, the space complexity is generally the roadblock to a use of higher-dimensional Weisfeiler-Leman in isomorphism testing and graph neural networks. The introduction of non-requantifiable variables is a technique to limit space complexity, so it needs to be investigated whether problems that arise in practice can be solved by this technique.

In another direction, it will be interesting to investigate other classes of graphs that are known to have bounded Weisfeiler-Leman dimension. Using the new variant with restricted reusability, it may be possible to obtain a more fine-grained complexity measure and more space-efficient algorithms for such graph classes. In particular, it seems highly plausible that (sufficiently connected) bounded genus graphs require only a limited number of requantifiable

variables. This might allow to design easier fixed-parameter tractable results for graph isomorphism, for example on bounded genus graphs (see [18]). However, for this there are further restrictions, as non-reusable variables must be choosable from FPT-size bounded sets.

References

- 1 Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25280–25293, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/d4d8d1ac7e00e9105775a6b660dd3cbb-Abstract.html>.
- 2 Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant sub-graph aggregation networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=dFbKQaRk15w>.
- 3 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Comb.*, 12(4):389–410, 1992. doi:10.1007/BF01305232.
- 4 Jinzhuan Cai, Jin Guo, Alexander L. Gavrilyuk, and Ilia Ponomarenko. A large family of strongly regular graphs with small Weisfeiler-Leman dimension. *arXiv:2312.00460 [math.CO]*, 2023. arXiv. doi:10.48550/arXiv.2312.00460.
- 5 Bireswar Das, Murali Krishna Enduri, and I. Vinod Reddy. Logspace and FPT algorithms for graph isomorphism for subclasses of bounded tree-width graphs. In M. Sohel Rahman and Etsuji Tomita, editors, *WALCOM: Algorithms and Computation - 9th International Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings*, volume 8973 of *Lecture Notes in Computer Science*, pages 329–334. Springer, 2015. doi:10.1007/978-3-319-15612-5_30.
- 6 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. *ACM Trans. Comput. Theory*, 14(2):8:1–8:33, 2022. doi:10.1145/3543686.
- 7 Anuj Dawar and David Richerby. The power of counting logics on restricted classes of finite structures. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic*, pages 84–98, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-74915-8_10.
- 8 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamani, 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 Zdenek Dvorák. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory*, 64(4):330–342, 2010. doi:10.1002/JGT.20461.
- 10 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 383–392. ACM, 2014. doi:10.1145/2591796.2591865.
- 11 Michael Elberfeld and Pascal Schweitzer. Canonizing graphs of bounded tree width in logspace. *ACM Trans. Comput. Theory*, 9(3):12:1–12:29, 2017. doi:10.1145/3132720.
- 12 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 *LIPICs*, pages 27:1–27:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.CSL.2024.27.

- 13 Martin Fürer. Weisfeiler-Lehman refinement requires at least a linear number of iterations. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 322–333. Springer, 2001. doi:10.1007/3-540-48224-5_27.
- 14 Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM*, 59(5):27:1–27:64, 2012. doi:10.1145/2371656.2371662.
- 15 Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, volume 47 of *Lecture Notes in Logic*. Cambridge University Press, 2017. doi:10.1017/9781139028868.
- 16 Martin Grohe. Counting bounded tree depth homomorphisms. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 507–520. ACM, 2020. doi:10.1145/3373718.3394739.
- 17 Martin Grohe. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE, 2021. doi:10.1109/LICS52264.2021.9470677.
- 18 Martin Grohe and Sandra Kiefer. A linear upper bound on the Weisfeiler-Leman dimension of graphs of bounded genus. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPICs*, pages 117:1–117:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.117.
- 19 Martin Grohe, Moritz Lichter, Daniel Neuen, and Pascal Schweitzer. Compressing CFI graphs and lower bounds for the Weisfeiler-Leman refinements. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 798–809. IEEE, 2023. doi:10.1109/FOCS57990.2023.00052.
- 20 Martin Grohe and Daniel Neuen. Canonisation and definability for graphs of bounded rank width. *ACM Trans. Comput. Log.*, 24(1):6:1–6:31, 2023. doi:10.1145/3568025.
- 21 Jin Guo, Alexander L. Gavriluk, and Iliia Ponomarenko. On the Weisfeiler-Leman dimension of permutation graphs. *arXiv:2305.15861 [math.CO]*, May 2023. arXiv. URL: <https://arxiv.org/abs/2305.15861>, doi:10.48550/arXiv.2305.15861.
- 22 Lauri Hella. Logical hierarchies in PTIME. *Inf. Comput.*, 129(1):1–19, 1996. doi:10.1006/INCO.1996.0070.
- 23 Jelle Hellings, Marc Gyssens, Jan Van den Bussche, and Dirk Van Gucht. Expressive completeness of two-variable first-order logic with counting for first-order logic queries on rooted unranked trees. In *Proceedings of the 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175828.
- 24 Neil Immerman. Relational queries computable in polynomial time. *Inf. Control.*, 68(1-3):86–104, 1986. doi:10.1016/S0019-9958(86)80029-8.
- 25 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.
- 26 Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. *Complexity Theory Retrospective*, pages 59–81, 1990. doi:10.1007/978-1-4612-4478-3_5.
- 27 Sandra Kiefer. The Weisfeiler-Leman algorithm: an exploration of its power. *ACM SIGLOG News*, 7(3):5–27, 2020. doi:10.1145/3436980.3436982.
- 28 Sandra Kiefer, Iliia Ponomarenko, and Pascal Schweitzer. The Weisfeiler-Leman dimension of planar graphs is at most 3. *J. ACM*, 66(6):44:1–44:31, 2019. doi:10.1145/3333003.
- 29 Haiyan Li, Iliia Ponomarenko, and Peter Zeman. On the Weisfeiler-Leman dimension of some polyhedral graphs. *arXiv:2305.17302 [math.CO]*, May 2023. arXiv. doi:10.48550/arXiv.2305.17302.

- 30 Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 400–404. ACM, 1992. doi:10.1145/129712.129750.
- 31 Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/f81dee42585b3814de199b2e88757f5c-Abstract.html>.
- 32 Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/J.EJC.2005.01.010.
- 33 Martin Otto. *Bounded Variable Logics and Counting: A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Cambridge University Press, 2017. doi:10.1017/9781316716878.
- 34 Oleg Pikhurko and Oleg Verbitsky. Logical complexity of graphs: A survey. In Martin Grohe and Johann A. Makowsky, editors, *Model Theoretic Methods in Finite Combinatorics - AMS-ASL Joint Special Session, Washington, DC, USA, January 5-8, 2009*, volume 558 of *Contemporary Mathematics*, pages 129–180. American Mathematical Society, 2009.
- 35 Jamie Tucker-Foltz. Inapproximability of unique games in fixed-point logic with counting. 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/LICS52264.2021.9470706.
- 36 Steffen van Bergerem. Learning concepts definable in first-order logic with counting. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785811.
- 37 Steffen van Bergerem. *Descriptive complexity of learning*. PhD thesis, RWTH Aachen University, Germany, 2023. URL: <https://publications.rwth-aachen.de/record/953243>, doi:10.18154/RWTH-2023-02554.
- 38 E. R. van Dam, J. H. Koolen, and H. Tanaka. Distance-regular graphs. *Electronic Journal of Combinatorics*, 1(DynamicSurveys), 2018. doi:10.37236/4925.
- 39 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.
- 40 Qing Wang, Dillon Ze Chen, Asiri Wijesinghe, Shouheng Li, and Muhammad Farhan. N-WL: A new hierarchy of expressivity for graph neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: <https://openreview.net/pdf?id=5cAI0qXyv>.