# An Optimal Lower Bound on the Number of Variables for Graph Identification

Jin-yi Cai*
*Computer Science Dept.*
*Princeton University*
*Princeton, NJ 08540*
jyc@princeton.edu

Martin Fürer†
*Dept. of Computer Science*
*Penn State University*
*University Park, PA 16802*
furer@cs.psu.edu

Neil Immerman‡
*Computer Science Dept.*
*University. of Massachusetts*
*Amherst, MA 01003*
immerman@cs.umass.edu

*Combinatorica,* (12:4) (1992), 389-410

**Abstract**

In this paper we show that $\Omega(n)$ variables are needed for first-order logic with counting to identify graphs on $n$ vertices. The $k$-variable language with counting is equivalent to the $(k-1)$-dimensional Weisfeiler-Lehman method. We thus settle a long-standing open problem. Previously it was an open question whether or not 4 variables suffice. Our lower bound remains true over a set of graphs of color class size 4. This contrasts sharply with the fact that 3 variables suffice to identify all graphs of color class size 3, and 2 variables suffice to identify almost all graphs. Our lower bound is optimal up to multiplication by a constant because $n$ variables obviously suffice to identify graphs on $n$ vertices.

## 1 Introduction

In this paper we show that $\Omega(n)$ variables are needed for first-order logic with counting to distinguish a sequence of pairs of graphs $G_n$ and $H_n$. These graphs have $O(n)$ vertices each, have color class size 4, and admit a linear time canonical labeling algorithm. This contrasts sharply with results in [10, 27] where it is shown that two variables suffice to identify all trees and almost all graphs, and that three variables suffice to identify all graphs of color class size 3.

*Research supported by NSF grant CCR-8709818.

†Research supported by NSF grant CCR-8805978 and Pennsylvania State University Research Initiation grant 428-45.

‡Research supported by NSF grants DCR-8603346 and CCR-8806308.

Another way to interpret our results is with stable colorings of $k$-tuples of vertices. The work of Weisfeiler and Lehman [40, 39] on combinatorial and group theoretic properties of colored graphs, has inspired the idea of separating the orbits of the automorphism group of a graph by coloring $k$-tuples of vertices. Sometimes, this approach is called, the $k$-dimensional Weisfeiler-Lehman method ($k$-dim W-L). In the late seventies and early eighties, this method was developed by many researchers, including Faradžev, Zemlyachenko, Babai, and Mathon. With $k = 1$, this method gives a linear-time graph isomorphism algorithm that works for almost all graphs [10]. Furthermore, the fastest known general graph isomorphism algorithms make use of this method with $k = O(\sqrt{n})$ [11]. It had been conjectured that this method would provide a polynomial time graph isomorphism test at least for graphs of bounded valence. (Valence is a synonym for degree.) Our result disposes of such conjectures.

Up until now, most lower bounds in this area were proved using random graphs. This method does not work when counting is included in the language because as mentioned above, almost all graphs can be identified using only two variables with counting. In our construction we choose graphs $T_n$, $(n = 1, 2, \ldots)$ with $O(n)$ vertices and separator size $n$ (Definition 6.3). Then we deterministically modify $T_n$ producing a pair of non-isomorphic graphs $G_n, H_n$, which agree on all properties expressible with $n$ variables. Our lower bound is linear in the separator size of the graphs $T_n$. This linear lower bound, combined with a straightforward upper bound (Proposition 7.3) allows us to precisely determine how many variables are needed to identify many classes of graphs in first-order logic, with or without counting.

This paper is organized as follows: In Section 2 we recount some of the history of the Weisfeiler-Lehman method. In Section 3 we give some background in descriptive complexity and explain the significance of this problem from the logical point of view. In Section 4 we introduce some combinatorial games and prove that they characterize logical equivalence in the languages we are considering. In Section 5 we prove the equivalence of the $(k-1)$-dimensional Weisfeiler-Lehman method and the $k$-variable language with counting. In Section 6 we use the above combinatorial game to prove the linear lower bound. Section 7 describes some corollaries and extensions of this work.

## 2 History of the Weisfeiler-Lehman Method

An old basic idea in graph isomorphism testing and canonical labeling is the naive vertex classification algorithm as described in Read and Corneil [37]. First, the vertices are labeled or colored with their valences. During the iteration, all labels are extended by the multiset ("set" with possibly multiple elements) of the labels of their neighbors. Between rounds, the labels are replaced by their order numbers in the lexicographic order of all the occurring labels. This always keeps the labels short. The algorithm stops when the set of labels stabilizes, meaning that no new differences between vertices are discovered. A labeling algorithm identifies a class of graphs, if all vertex properties which are invariant under isomorphisms are discovered. In other words, the sets of vertices with the same labels are the orbits of the automorphism group.

The naive vertex classification algorithm, which we want to call the one dimensional Weisfeiler-Lehman method (1-dim W-L), does not solve the worst cases of the graph isomorphism problem. Nevertheless, it is usually a good start, and in fact it succeeds most of the time.

Babai, Erdős and Selkow [8] have shown that the 1-dim W-L algorithm already produces normal forms for all but an $n^{-1/7}$ fraction of the $n$-vertex graphs. This has been improved to a $c^{-n \log n / \log \log n}$ fraction [10] producing an average linear time canonical labeling algorithm by handling the few exceptions with a slow algorithm.

Vertex classification is probably the basis of every practical implementation of a graph isomorphism test. For example, this is the case for the "nauty" package [35], which is said to be the fastest practical graph isomorphism package. It should be mentioned, however, that in addition to vertex refinement, "nauty" makes extensive use of partial automorphism information in its backtrack process; and it is not clear whether or not our examples may lead to graphs on which "nauty" requires excessive time. In general, it is quite difficult to construct "hard cases" for graph isomorphism.

There is a class of graphs for which the vertex classification algorithm alone is obviously useless, because the algorithm cannot even get started. These are the regular graphs, which have the same degree in each vertex. Here it seems quite natural to go beyond vertex classification to the 2-dim W-L or edge classification algorithm. Initially every ordered pair $(u, v)$ is labeled or colored with one of three possible colors, depending on whether $u = v$ and whether there is an edge $\{u, v\}$. Then information about the multiset of pairs of colors assigned to paths of length 2 from $u$ to $v$ is repeatedly added to the color of $(u, v)$. The algorithm stops when no color class is split any more. A modification of this algorithm has been shown to produce normal forms for all regular graphs in linear average time [29].

The $k$-tuple coloring algorithm (named $k$-dim W-L by Babai [13]) classifies $k$-tuples of vertices. It might color vertices and edges implicitly by using $k$-tuples with repetition of components. It could start with some encoding of the graph into the labels assigned to the $k$-tuples. For example, the initial label or color of every $k$-tuple could be the number of its distinct components except when this number is two. Then two colors could be used to encode the presence or absence of an edge between the two vertices. We prefer to get a quicker start by initially coloring each $k$-tuple with its isomorphism type. Repeatedly, the color of $(u_1, \ldots, u_k)$ is refined by the $n$ element multiset (containing one element for each vertex $v$) of $k$-tuples of colors previously assigned to

$$(v, u_2, \ldots, u_k), (u_1, v, u_3, \ldots, u_k), \ \ldots \ , (u_1, \ldots, u_{k-1}, v)$$

We only need to consider $k$-tuples of distinct elements, if we finally color the vertices by the multiset of colors of incident $k$-tuples. A more formal description of the $k$-dim W-L method is given in Section 5.

Possibly weaker algorithms have been considered. We might call them *special $k$-dim W-L algorithms*. In Weisfeiler's book [39] only such a method is mentioned and called *deep stabilization*. It consists of individualization followed by a low dimensional ($k = 1$ or 2) W-L algorithm. For every distinguished (i.e., initially colored with unique colors) $(k-1)$-tuple or $(k-2)$-tuple, a 1-dim W-L or a 2-dim W-L respectively is performed to detect invariant properties of vertices or edges. These methods seem to be weaker than the standard $k$-dim W-L method.

Two of the current authors have independently reinvented the $k$-dim W-L algorithm in the early eighties and conjectured its capability of identifying the graphs of bounded valence (with $k$ being a suitable function of the valence). Later, we have learned that such conjectures have been around before in the Soviet Union, where the $k$-dim W-L algorithm (or

maybe sometimes a special $k$-dim W-L algorithm) has been investigated for two decades. Significant results have been obtained by Faradžev's group, which contributed many papers to Weisfeiler's book [39]. The Russians have built a huge algebraic theory with extensive applications around the notion of stable colorings of pairs. The key notion is that of a *cellular algebra* (see [39, 30]), which has been discovered in another context and called *coherent configuration* by Higman [21].

Weisfeiler and Lehman have asked whether the special $k$-dim W-L method with a slowly growing value of $k$ would be sufficient to solve the graph isomorphism problem. There was actually good reason to conjecture $k = O(\log n)$ or even $O(1)$ to be sufficient.

The second hope was partly based on the following result of Cameron [14], obtained independently by Gol'fand (cf. [19, 31]). Let us call a graph *$k$-regular*, if the number of common neighbors of a $k$ element subset of vertices only depends on the isomorphism type of the subgraph induced by the $k$ vertices. (1-regular and 2-regular graphs are well known as *regular* and *strongly regular* graphs respectively.) Cameron and Gol'fand have shown that apart from the pentagon and the line graph of K(3,3), only the trivial examples of 5-regular graphs exist, namely the disjoint unions of complete graphs of equal size, and their complements (complete multipartite graphs). These graphs are homogeneous, i.e., all isomorphisms of their subgraphs extend to automorphisms. Therefore, they are immune to $k$-dim W-L refinements for any $k$: No refinement beyond the isomorphism type of $k$-tuples will follow. However, for any other graph, the Cameron-Gol'fand result assures us that the 5-dim W-L method will give at least some nontrivial partitioning of the 5-tuples.

Lipton [32] has proved that a special $k$-dim W-L method with a fixed $k$ is sufficient for canonical labeling of trivalent (degree 3) graphs with arc-transitive automorphism groups. (An arc is an ordered pair of adjacent vertices.)

Support for the $k = O(\log n)$ conjecture has been provided by Gary Miller [36]. He has shown that for certain classes of strongly regular graphs and other combinatorial objects such as Latin squares $k = \log n$ is sufficient. Previously, such graphs have been considered to be difficult examples for isomorphism testing, because of their high degree of regularity and symmetry. The importance of symmetries for graph isomorphism testing has been pointed out by Babai and by Mathon [34] who showed that the graph isomorphism problem is equivalent to computing the order of automorphism groups of graphs.

Individualization followed by a low (1 or 2) dimensional refinement (i.e., the special W-L method) has produced pioneer results in the areas of bounded valence as well as general graph isomorphism and canonical labeling. Babai's technical report [3] started to use group theoretical algorithms to obtain provable upper bounds for isomorphism problems. Not only did he get his well known probabilistic polynomial time isomorphism test for graphs of bounded color class size, he also started the work on bounded valence graphs. Individualization of $k = \sqrt{n}(\log n)^c$ vertices splits a bounded valence graph into color classes of size at most $\sqrt{n}$ resulting in an $exp(\sqrt{n}(\log n)^c)$ isomorphism test. Subsequently Luks [33] proved, using group theory to greater depth, that isomorphism for graphs of bounded valence is in polynomial time. Finally the canonical labeling problem for graphs of bounded valence has been solved in polynomial time by [11] and [18] independently.

Individualization followed by naive refinement has also been the tool used by Babai to handle strongly regular graphs [4] and primitive coherent configurations [6]. He used individualization of $k = 2\sqrt{n}\log n$ vertices. Strongly regular graphs and more generally,

coherent configurations are stable under 2-dim W-L. While strongly regular graphs are just undirected graphs, coherent configurations are edge-colored complete directed graphs. A coherent configuration is primitive if the diagonal has one color and all other colors define connected graphs. If a transitive automorphism group is primitive, then 2-dim W-L produces necessarily a primitive coherent configuration. For tournaments, the isomorphism problems of primitive and arbitrary coherent configurations are polynomial time equivalent [11].

The general graph isomorphism problem has been attacked by Zemlyachenko. The method is described in [5] and [41]. By individualization of $O(\sqrt{n})$ vertices and canonical edge-switching, he has been able to reduce the valence to $O(\sqrt{n})$. Combining this with the method of Luks [33], Zemlyachenko obtained the first interesting upper bound for general graph isomorphism [41] (cf. [5]). His bound is $\exp(n^{1-c})$ for some positive constant $c$. This has subsequently been improved by Babai and Luks [11] to $\exp(n^{1/2+o(1)})$.

Instead of measuring the reduction in the valence, one could ask about the effect of these methods on the color class size. Babai [7] has investigated this splitting power of Zemlyachenko's method combined with 2-dim W-L. The result is that individualizing $k = O(n^{2/3} \log n)$ vertices, and applying Zemlyachenko's method *and* the 2-dim W-L method, he obtains color classes which have $\leq k$ vertices in each connected component of the resulting graph.

# 3 Logical Background

In [23, 24, 25] one of us has pursued an alternate view of complexity theory in which the complexity of a problem is characterized in terms of the complexity of the simplest first-order sentences expressing the problem. For example, it is shown in [23] that the polynomial-time properties are exactly the properties expressible by first-order sentences iterated[1] polynomially many times:

**Fact 3.1 ([23])**

$$P \;=\; \bigcup_{k=1}^{\infty} \mathrm{FO}(\leq)[n^k]$$

The notation $\mathrm{FO}(\leq)[n^k]$ denotes the set of properties describable by a very uniform sequence of sentences $\{\varphi_n\}$ such that each sentence $\varphi_n$ has length $O(n^k)$ and has a bounded number of variables independent of $n$.[2] The symbol $\leq$ is included to emphasize the presence of a total ordering on the universe of the input structures. In [24] and in [38] it is also shown that this uniform sequence of formulas can be represented by a least fixed point operator (LFP) applied to a single formula. Thus,

$$P \;=\; \mathrm{FO}(\leq) + \mathrm{LFP} \;=\; \bigcup_{k=1}^{\infty} \mathrm{FO}(\leq)[n^k]\,.$$

[1] More precisely, the sentence expressing the property for structures of size $n$ consists of a fixed block of restricted quantifiers written $p(n)$ times, followed by a fixed formula.

[2] In [23] the notation Var&Sz$[O(1), n^k]$ instead of FO$[n^k]$ was used.

Fact 3.1 gives a natural language expressing exactly the polynomial-time properties of ordered graphs. Let a *graph property* be an order independent property of ordered graphs. One can ask the question,

**Question 3.2** *Is there a natural language for the polynomial-time graph properties?*

Since the notion of "natural" is not well defined, some readers may prefer the more precise question:

**Question 3.3** *Is there a recursively enumerable listing of a set of Turing machines that accept exactly all the polynomial-time graph properties?*

These questions were first asked with respect to database query languages [15]. See [28] for a discussion of the role of ordering in the database context.

We remark that should it be the case that graph canonization (i.e. given a graph return a canonical form such that two graphs are isomorphic iff their canonical forms are equal) is in polynomial time, then the answer to Question 3.3 is, "Yes." Thus a negative answer would imply that P is not equal to NP.

Previous to this paper, the only polynomial-time graph properties known not to be expressible in FO + LFP (without ordering) were "counting problems". For example, that a graph has an even number of edges is not expressible in FO + LFP. In [24] a language which we now call "FO + LFP + COUNT" was proposed as an answer to Question 3.2. This language describes two sorted structures consisting of an unordered domain of vertices together with an edge predicate, plus an ordered domain of numbers. The domains are defined via counting quantifiers as in Section 3.2. We show in Corollary 7.1 that this language fails badly on certain linear time properties of graphs.

In [27] and [26] the exact number of variables needed to identify various classes of trees with and without counting, respectively, is determined. (Without counting this number increases linearly with the arity of the trees; with counting two variables suffice.) The question of how many variables are needed to identify various classes of graphs is interesting in its own right, and also has applications to temporal logic [26].

In the remainder of this section we explain the logical background we need. Some of this material is described in more detail in [27].

## 3.1 First-Order Logic

For our purposes, a *graph* will be defined as a finite first-order structure, $G = \langle V_G, E_G \rangle$. $V_G$ is the universe, (the vertices). $E_G$ is a binary relation on $V_G$, (the edges).

As an example, the undirected graph, $G_1 = \langle V_1, E_1 \rangle$, pictured in Figure 1 has vertex set $V_1 = \{0, 1, 2, 3, 4\}$, and edge relation

$$E_1 = \{\langle 0, 3\rangle, \langle 0, 4\rangle, \langle 1, 2\rangle, \langle 1, 3\rangle, \ldots, \langle 4, 0\rangle, \langle 4, 3\rangle\}$$

consisting of 12 pairs corresponding to the six undirected edges. By convention, we will assume that all structures referred to in this paper have universe $\{0, 1, \ldots, n-1\}$ for some natural number $n$.

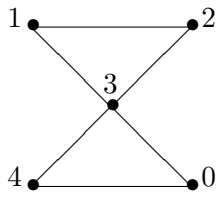


Figure 1: An Undirected Graph

The *first-order language* of graph theory is built up in the usual way from the variables $x_1, x_2, \ldots$, the relations symbols $E$ and $=$, the logical connectives $\wedge, \vee, \neg, \rightarrow$, and the quantifiers $\forall$ and $\exists$. The quantifiers range over the vertices of the graph in question. For example consider the following first-order sentence:

$$\varphi \equiv \forall x \forall y [E(x,y) \rightarrow \ E(y,x) \wedge x \neq y]$$

$\varphi$ says that $G$ is undirected and loop free. We will only consider graphs that satisfy $\varphi$, in symbols: $G \models \varphi$.

It is useful to consider a slightly more general set of structures. The *first-order language of colored graphs* results from the addition of a countable set of unary relations $\{C_1, C_2, \ldots\}$ to the first-order language of graphs.[3] Define a *colored graph* to be a graph that interprets these new unary relations so that all but finitely many of the predicates are false at each vertex. These unary relations may be thought of as colorings of the vertices.

**Definition 3.4** For a given language $\mathcal{L}$ we say that the graphs $G$ and $H$ are *$\mathcal{L}$-equivalent* ($G \equiv_{\mathcal{L}} H$) iff for all sentences $\varphi \in \mathcal{L}$,

$$G \models \varphi \ \ \Leftrightarrow \ \ H \models \varphi \, .$$

We say that $\mathcal{L}$ *identifies* the graph $G$ iff for all graphs $H$, if $G \equiv_{\mathcal{L}} H$ then $G$ and $H$ are isomorphic. $\mathcal{L}$ *identifies* a set of graphs $S$ if it identifies every element of $S$.

**Note:** For the languages $\mathcal{L}_k, \mathcal{C}_k$ which we consider in this paper, and any graph $G$, the set of sentences in the language that are true about $G$ has a polynomial size description which may be computed in polynomial-time [27]. Thus any set of graphs identified by $\mathcal{L}_k$ or $\mathcal{C}_k$ has a polynomial-time canonization algorithm.

Of course the First-Order Language of Colored Graphs identifies all colored graphs. From a computational viewpoint it is interesting to consider weaker languages admitting much faster equivalence testing algorithms.

### 3.2 The Languages $\mathcal{L}_k$ and $\mathcal{C}_k$

Define $\mathcal{L}_k$ to be the set of first-order formulas $\varphi$, such that the variables in $\varphi$ are a subset of $x_1, x_2, \ldots, x_k$. Note that variables in first-order formulas are similar to variables in programs: they can be reused (i.e. requantified).

[3]Coloring relations are a clean tool for restricting the automorphisms of graphs. However, all the coloring relations in this paper could be replaced by simple gadgets in the graphs, without changing any of the results.

For example, consider the following sentence in $\mathcal{L}_2$.

$$\psi \quad \equiv \quad \forall x_1 \exists x_2 \Big( E(x_1, x_2) \wedge \exists x_1 [\neg E(x_1, x_2)] \Big)$$

The sentence, $\psi$, says that every vertex is adjacent to some vertex which is itself not adjacent to every vertex. As an example, the graph from Figure 1 satisfies $\psi$. Note that the outermost quantifier, $\forall x_1$, refers only to the free occurrence of $x_1$ within its scope.

Define a *color class* to be the set of vertices which satisfy a particular set of color relations. The *color class size* of a graph is the cardinality of its largest color class. In [27] it is shown that $\mathcal{L}_3$ identifies the set of graphs of color class size 3.

As noted above, the languages $\mathcal{L}_k$ are too weak to count, or even to express the parity of the number of edges. It is thus natural to strengthen these languages by adding *counting quantifiers* to the languages $\mathcal{L}_k$, thus obtaining the new languages $\mathcal{C}_k$. For each positive integer $i$, we include the quantifier, $(\exists i\, x)$. The meaning of "$(\exists 17\, x_1)\varphi(x_1)$", for example, is that there exist at least 17 vertices such that $\varphi$. It is sometimes convenient to use the following abbreviation $(\exists! i\, x)$, meaning that there exists exactly $i$ $x$'s:

$$(\exists! i\, x)\varphi(x) \quad \equiv \quad (\exists i\, x)\varphi(x) \ \wedge \ \neg(\exists i+1\, x)\varphi(x)$$

As an example, the following sentence in $\mathcal{C}_2$ says that there exist exactly 17 vertices of degree 5,

$$(\exists! 17\, x_1)(\exists! 5\, x_2) E(x_1, x_2)$$

As an even worse example, the following sentence in $\mathcal{C}_2$ identifies the graph in Figure 1. It says that the whole graph contains exactly 5 vertices and that one vertex is adjacent to four vertices each of which has degree 2.

$$[(\exists! 5\, x_1)(x_1 = x_1)] \ \wedge \ [(\exists! 1\, x_1)(\exists 4\, x_2)(E(x_1, x_2) \wedge (\exists! 2\, x_1) E(x_2, x_1))]$$

Note that every sentence in $\mathcal{C}_k$ is equivalent to an ordinary first-order sentence with perhaps many more variables and quantifiers. In Section 5 it is shown that testing $\mathcal{C}_k$ equivalence corresponds to the $(k-1)$-dimensional Weisfeiler-Lehman Method. It thus follows that the language $\mathcal{C}_2$ identifies all trees and almost all graphs. In [27], TIME$(n^k \log n)$ algorithms for testing $\mathcal{L}_k$ or $\mathcal{C}_k$ equivalence of graphs on $n$ vertices are presented.

# 4 Pebbling Games

We next describe two pebbling games that are equivalent to testing $\mathcal{L}_k$ and $\mathcal{C}_k$ equivalence, respectively. These games are variants of the games of Ehrenfeucht and Fraïssé, [16, 17]. The results in this section concerning the $\mathcal{L}_k$ game and the $\mathcal{C}_k$ game originally appeared in [23] and [27], respectively.

Let $G$ and $H$ be two graphs, and let $m$ and $k$ be natural numbers. Define the $m$-move $\mathcal{L}_k$ *game* on $G$ and $H$ as follows. There are two players, and for each variable $x_i$, $i = 1, \ldots, k$ there is a pair of $x_i$ pebbles.

On each move, Player I picks up the pair of $x_i$ pebbles, for some $i \in \{1, \ldots, k\}$, and he places one of them on a vertex in one of the graphs.[4] Player II must then place the other $x_i$ pebble on a vertex of the other graph.

Define a *k-configuration* on a pair of graphs $G, H$ to be a pair $(u, v)$ of partial functions,

$$u : \{x_1, \ldots, x_k\} \to V_G; \quad v : \{x_1, \ldots, x_k\} \to V_H$$

such that the domains of $u$ and $v$ are equal. We will use the notation $D_u$ to denote the domain of the partial function $u$. Thus a $k$-configuration on $G, H$ is a valid position of the $\mathcal{L}_k$ game on $G, H$. Here $u(x_i) = g$ means that an $x_i$ pebble is on $g \in V_G$. If $x_i \notin D_u = D_v$ this means that the $x_i$ pebbles are not currently placed on the board.

Let $(u_r, v_r)$ be the configuration of the game after move number $r$. Then we say *Player I wins the game after move r* if the map that takes $u_r(x_i)$ to $v_r(x_i)$, $i \in D_{u_r}$, is not an isomorphism of the subgraphs induced by these vertices. (Note that if the graphs are colored then an isomorphism must preserve colors as well as edges.) We say that Player I wins the $m$-move game if for some $r \in \{0, 1, 2, \ldots, m\}$, Player I wins the game after move $r$. Player II wins iff Player I does not win. Finally, we say that Player II has a winning strategy for the $\mathcal{L}_k$ game on $G$ and $H$, iff for all $m$, Player II has a winning strategy for the $m$-move game on $G$ and $H$.

Thus Player II has a winning strategy for the $\mathcal{L}_k$ game just if she can always find matching vertices to preserve the isomorphism.[5] Player I is trying to point out a difference between the two graphs and Player II is trying to keep them looking the same.

The number of moves in the $\mathcal{L}_k$ game corresponds to the depth of nesting of quantifiers of the sentences in $\mathcal{L}_k$ needed to distinguish the graphs $G$ and $H$. Define the language $\mathcal{L}_{k,m}$ to be the restriction of $\mathcal{L}_k$ to formulas of quantifier depth $m$. The relationship between the $\mathcal{L}_k$ game and the language $\mathcal{L}_k$ is given in Theorem 4.2. Before we state it, we need the following definition.

**Definition 4.1** Let $G, H$ be a pair of graphs and let $(u, v)$ be a $k$-configuration on $G, H$. We will say that $G$ is $\mathcal{L}_{k,m}$*-equivalent* to $H$, in symbols, $G \equiv_{\mathcal{L}_{k,m}} H$ iff for all formulas $\varphi \in \mathcal{L}_{k,m}$ whose free variables are a subset of $D_u$,

$$G, u \models \varphi \quad \Leftrightarrow \quad H, v \models \varphi$$

Similarly we will say that $G$ is $\mathcal{L}_k$*-equivalent* to $H$, in symbols, $G \equiv_{\mathcal{L}_k} H$ iff for all $m$, $G \equiv_{\mathcal{L}_{k,m}} H$.

**Theorem 4.2 ([23])** *Player II has a winning strategy for the m-move $\mathcal{L}_k$ game on $G, H$ if and only if $G \equiv_{\mathcal{L}_{k,m}} H$. Thus, Player II has a winning strategy for the $\mathcal{L}_k$ game on $G, H$ iff $G \equiv_{\mathcal{L}_k} H$.*

Before we prove Theorem 4.2, we will give a few examples of the game.

[4] To make the play of the games easier to follow we will use masculine pronouns for Player I and feminine pronouns for Player II.

[5] By definition, the strategy can depend on the given number $m$ of moves, but as $G$ and $H$ are finite, there is actually one strategy winning for all $m$.

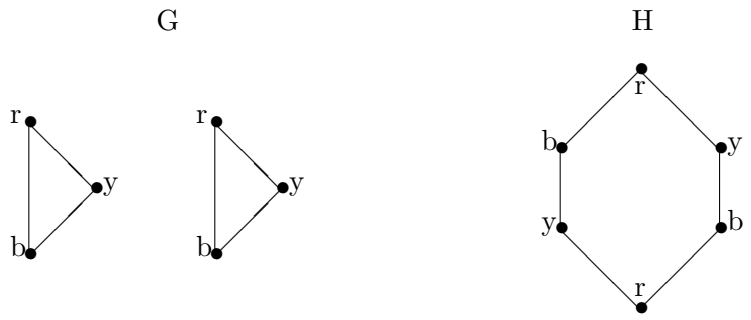


Figure 2: The $\mathcal{L}_2$ Game

**Example 4.3** Consider the $\mathcal{L}_2$ game on the graphs $G$ and $H$ shown in Figure 2.

Suppose that Player I's first move is to place an $x_1$ pebble on a red vertex in $G$. Player II may answer by putting the other $x_1$ pebble on either of the red vertices in $H$. Now suppose Player I puts $x_2$ on an adjacent yellow vertex in $H$. Player II has a response because in $G$, every red vertex has an adjacent yellow vertex. The reader should convince himself or herself that in fact Player II has a winning strategy for the $\mathcal{L}_2$ game on the given $G$ and $H$. It follows from Theorem 4.2 that $G$ and $H$ agree on all sentences from $\mathcal{L}_2$.

On the other hand, clearly Player I has a win in the 3-move, $\mathcal{L}_3$ game on $G$ and $H$. He can simply put his pebbles on three points in one of the triangles in $G$. Since $H$ has no triangle, Player II will lose. Notice that in this case Player I is playing the following sentence from $\mathcal{L}_{3,3}$ which is true of $G$ and false of $H$:

$$(\exists x_1)(\exists x_2)(\exists x_3)[E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)]$$

Finally, a more interesting example of an $\mathcal{L}_3$ game would be with $H'$ consisting of a hexagon like $H$, but without the colors and $G'$ consisting of a disjoint union of two copies of the hexagon $H'$. Here Player II has a winning strategy for the 3-move $\mathcal{L}_3$ game, but Player I has a winning strategy for the 4-move $\mathcal{L}_3$ game.[6] His strategy is to play the following sentence true of $H'$ but not of $G'$ saying that every pair of vertices is joined by a path of length at most three:

$$(\forall x)(\forall y)(\exists z)[(E(x, z) \vee x = z) \wedge (\exists x)(E(z, x) \wedge E(x, y))]$$

In order to prove Theorem 4.2 we need the following

**Lemma 4.4** *For any relational language with finitely many relation symbols, and any $k$ and $m$ there are only finitely many formulas up to equivalence in $\mathcal{L}_{k,m}$. Furthermore if we include a finite set of additional quantifiers, e.g. counting quantifiers, then the expanded language still only has finitely many formulas up to equivalence.*

[6]The reason we removed the colors is that with the colors there is a sentence from $\mathcal{L}_{3,3}$ distinguishing the two graphs, namely in $H$ every pair of vertices of different color is joined by a path of length at most two.

**Proof** This is easy to see by induction on $m$. When $m = 0$, there are only finitely many variables and only finitely many relation symbols, so only finitely many sets of possible facts about these variables. Assume that there are a total of $f$ different kinds of quantifiers. Inductively, assume there are $s_m$ inequivalent formulas of quantifier depth $m$. Then there are no more than $2^{2^{fks_m}}$ inequivalent formulas of quantifier depth $m+1$. ∎

**Proof of Theorem 4.2:** We prove by induction on $m$, that for all $k$-configurations $(u, v)$ on $G, H$ the following statements are equivalent:

1. Player II has a winning strategy for the $m$-move $\mathcal{L}_k$ game on $G, H$ starting from the initial configuration $(u, v)$.
2. $G, u \equiv_{\mathcal{L}_{k,m}} H, v$

The base of the induction is immediate because the map from $u(D_u)$ to $v(D_u)$ is an isomorphism of the induced subgraphs iff $G, u$ and $H, v$ agree on all quantifier-free formulas.[7]

Assume the equivalence of (1) and (2) for all $m$-move games, and let $(u, v)$ be the initial configuration of an $(m+1)$-move game. Assume condition (2) is false and let $\varphi \in \mathcal{L}_{k,m+1}$ be a formula on which $G, u$ and $H, v$ disagree. If $\varphi$ is a disjunction, conjunction, or negation of smaller formulas then $G, u$ and $H, v$ must disagree on one of these smaller formulas, so we may assume that $\varphi$ begins with a quantifier. We may assume by symmetry that $\varphi = (\exists x_i)\psi$ and $G, u \models \varphi$, but $H, v \models \neg\varphi$. Player I should then place one of the $x_i$ pebbles on a vertex $g$ such that $\psi$ holds in $G, u(x_i/g)$. No matter what vertex $h$ Player II answers with, we know that $\neg\psi$ will hold in $H, v(x_i/h)$. Letting $(u_1, v_1) = (u(x_i/g), v(x_i/h))$ be the configuration after this move we have that $G, u_1 \not\equiv_{k,m} H, v_1$. Thus by induction Player I has a winning strategy for the remaining $m$-move game and thus for the original $m+1$-move game.

Conversely, assume that condition (2) is true and let Player I's first move be to place one of the $x_i$ pebbles on some vertex $g$ from $G$. Let $u_1 = u(x_i/g)$ be the result of this move. Note that there are only finitely many color predicates that any vertex in $G$ or $H$ satisfies. Thus, Lemma 4.4 applies and there is only a finite set $F_{k,m}$ of inequivalent formulas of interest in $\mathcal{L}_{k,m}$. Define $S$ to be the set of formulas in $F_{k,m}$ that are satisfied by $G, u_1$ and let $\sigma$ be the conjunction of the finitely many formulas in $S$. Thus we have that

$$G, u \models (\exists x_i)\sigma$$

It follows that $H, v \models (\exists x_i)\sigma$. Let Player II place the other $x_i$ pebble on a witness, $h$, for $\sigma$ in $H$, and let $v_1 = v(x_i/h)$ be the result of this move. By the definition of $\sigma$ it follows that

$$G, u_1 \equiv_{\mathcal{L}_{k,m}} H, v_1$$

Thus it follows by induction that Player II has a winning strategy for the remaining $m$-move game and thus also for the original $m+1$-move game. ∎

[7] We are presenting the proof here for the language with no constant or function symbols. The proof goes through when function symbols are present, under the additional assumption that the cardinality of any finitely generated set is finite [26].

### 4.1 The $\mathcal{C}_k$ Game

A modification of the $\mathcal{L}_k$ game provides a combinatorial tool for analyzing the expressive power of $\mathcal{C}_k$. Given a pair of graphs define the $\mathcal{C}_k$ game on $G$ and $H$ as follows: Just as in the $\mathcal{L}_k$ game, we have two players and $k$ pairs of pebbles. The difference is that each move now has two parts.

1. Player I picks up the $x_i$ pebble pair for some $i$. He then chooses a set $A$ of vertices from one of the graphs. Now Player II answers with a set $B$ of vertices from the other graph. $B$ must have the same cardinality as $A$.
2. Player I places one of the $x_i$ pebbles on some vertex $b \in B$. Player II answers by placing the other $x_i$ pebble on some $a \in A$.

The definition for winning is as before. What is going on in the two part move is that Player I asserts that there exist $|A|$ vertices in $G$ with a certain property. Player II answers with the same number of such vertices in $H$. Player I challenges one of the vertices in $B$ and Player II replies with an equivalent vertex from $A$. This game captures expressibility in $\mathcal{C}_k$:

**Theorem 4.5 ([27])** *Player II has a winning strategy for the $\mathcal{C}_k$ game on $G, H$ if and only if $G \equiv_{\mathcal{C}_k} H$.*

Theorem 4.5 follows from Theorem 5.2, which we prove in the next section.

## 5 $\mathcal{C}_k$-Equivalence Equals $(k-1)$-dim W-L

In this section we describe the $k$-dimensional Weisfeiler-Lehman method ($k$-dim W-L). We then prove that a pair of $k$-tuples of vertices from a graph agree on all formulas in $\mathcal{C}_{k+1}$ iff they are in the same equivalence class arising from the $k$-dim W-L.

The 1-dim W-L is also called vertex refinement. Let $G = \langle V, E, C_1, \ldots, C_r \rangle$ be a colored graph in which every vertex satisfies exactly one color relation. Let $W^0 : V \to \{1 \ldots n\}$ be given by $W^0(v) = i$ iff $v \in C_i$. We then define $W^{r+1}$, the refinement of $W^r$ as follows: The new color of each vertex g is defined to be the following tuple:

$$\langle W^r(g), y_1, n_1, \ldots, y_r, n_r \rangle$$

where $y_i$ is the number of vertices of color $i$ that $g$ is adjacent to, and $n_i$ is the number of vertices of color $i$ that $g$ is not adjacent to. In practice, we sort these new colors lexicographically and assign $W^{r+1}(g)$ to be the number of the new color class that $g$ inhabits. However, we retain a table decoding the "meaning" of each of the colors. Thus two vertices are in the same new color class precisely if they were in the same old color class, and they were adjacent to the same number of vertices of each color. We keep refining the coloring until at some level $W^r = W^{r+1}$. We let $\overline{W} = W^r$ and call $\overline{W}$ the *stable refinement* of $W^0$.

We will see in Theorem 5.2 that stable coloring provides exactly the same information as $\mathcal{C}_2$ equivalence.

Next define the $k$-dim W-L for $k > 1$ as follows. Let $G$ be a colored graph and let $u$ be a (total) map from $\{x_1, \ldots, x_k\}$ to $V_G$. Define the initial color $W^0(u)$ according to the isomorphism type of $u$. That is, $W^0(u) = W^0(v)$ iff the map from $(u(x_1), \ldots, u(x_k))$ to $(v(x_1), \ldots, v(x_k))$ is an isomorphism.

For each $g \in V_G$, define the operation

$$\text{sift}(f, u, g) \;=\; \langle f(u(x_1/g)), f(u(x_2/g)), \ldots, f(u(x_k/g))\rangle$$

Thus $\text{sift}(W^r, u, g)$ is the $k$-tuple of $W^r$-colors arising from substituting $g$ in turn for each of the $k$ positions in $u$.

We define the $r+1^{\text{st}}$ color of $u$ from the $r^{\text{th}}$ color by considering the $r^{\text{th}}$ color of $u$ together with the number of vertices $g$ such that $\text{sift}(W^r, u, g) = \bar{t}$ for each possible $k$-tuple of colors $\bar{t}$. More explicitly, form the new color of $u$ as the tuple:

$$\langle W^r(u), \text{SORT}\{\text{sift}(W^r, u, g) \mid g \in G\}\rangle$$

As in the one dimensional case, we sort these new colors lexicographically, and assign $W^{r+1}$ according to the ordering that ensues. However, we do retain a table decoding the meaning of each color. Thus for a pair of configurations $u, v$ from different graphs, $W^{r+1}(u) = W^{r+1}(v)$ iff the numbers of the colors assigned are the same, and the decoding tables for the two graphs are identical. Thus $W^{r+1}(u) = W^{r+1}(v)$ just if $W^r(u) = W^r(v)$ and for each $k$-tuple of colors $\bar{t}$,

$$|\{g \,|\, \text{sift}(W^r, u, g) = \bar{t}\}| \quad = \quad |\{g \,|\, \text{sift}(W^r, v, g) = \bar{t}\}| \tag{5.1}$$

(Note that the difference between the case $k = 1$ and the case $k > 1$ is that in the former case we have to explicitly consider which of the $g$'s are adjacent to $u(x_1)$ in the above definition of new color; whereas, for $k > 1$, this adjacency is part of the information in the initial color of the tuples $u(x_j/g)$ for $j \neq 1$.)

Let $\overline{W}(u)$ denote the stable color of $u$. Note that there can be at most $n^k$ color classes for a graph with $n$ vertices and thus the algorithm stops after at most $n^k$ iterations.

**Theorem 5.2** *Let $G, H$ be a pair of colored graphs and let $(u, v)$ be a $k$-configuration on $G, H$, where $k \geq 1$. Then the following are equivalent:*

1. $\overline{W}(u) = \overline{W}(v)$
2. $G, u \equiv_{\mathcal{C}_{k+1}} H, v$
3. *Player II has a winning strategy for the $\mathcal{C}_{k+1}$ game on $(G, H)$, whose initial configuration is $(u, v)$.*

**Proof** By induction on $r$ we show that the following are equivalent:

1. $W^r(u) = W^r(v)$
2. $G, u \equiv_{\mathcal{C}_{k+1,r}} H, v$

3. Player II has a winning strategy for the $r$-move $\mathcal{C}_{k+1}$ game on $(G, H)$ whose initial configuration is $(u, v)$.

The base case is by definition. $W^0(u) = W^0(v)$ iff the map from $u(x_1), \ldots, u(x_k)$ to $v(x_1), \ldots, v(x_k)$ is an isomorphism. This is true iff $G, u$ and $H, v$ satisfy all the same quantifier-free formulas; and it is also the definition of Player II winning the zero move game.

Assume that the equivalence holds for all $(u, v)$ and for all $r < m$.

$(\neg 1 \Rightarrow \neg 2)$ : Suppose that $W^m(u) \neq W^m(v)$. There are two cases. If $W^{m-1}(u) \neq W^{m-1}(v)$ then by the inductive assumption there is a formula $\varphi \in \mathcal{C}_{k+1,m-1}$ on which $G, u$ and $H, v$ differ. Otherwise it must be that for some $k$-tuple of colors, $\bar{t} = (t_1, \ldots, t_k)$, Equation 5.1 fails. Let $N$ be the cardinality of the larger set in Equation 5.1.

By induction, two $k$-tuples of vertices are in the same $f^{(m-1)}$ color class iff they agree on all formulas from $\mathcal{C}_{k+1,m-1}$. By Lemma 4.4 there are only finitely many inequivalent $\mathcal{C}_{k+1,m-1}$ formulas, when we restrict our attention to graphs with the same finite number of vertices as $G$. (If $G$ and $H$ have different numbers of vertices, then for $r \geq 1$, all the above conditions are false.) Let $\psi_i$ be the conjunction of the finitely many $\mathcal{C}_{k+1,m-1}$ formulas characterizing the $m-1$ color class $i$. Thus, for $w$ a $k$-configuration on $F \in \{G, H\}$

$$W^{m-1}(w) = i \quad \Leftrightarrow \quad F, w \models (\psi_i)$$

It follows that $G, u$ and $H, v$ differ on the following formula from $\mathcal{C}_{k+1,m}$.[8]

$$(\exists N x_{k+1})(\psi_{t_1}(x_1/x_{k+1}) \wedge \cdots \wedge \psi_{t_k}(x_k/x_{k+1}))$$

$(\neg 2 \Rightarrow \neg 3)$ : Suppose that $G, u \models \varphi$ but $H, v \models \neg\varphi$, for some $\varphi \in \mathcal{C}_{k+1,m}$. If $\varphi$ is a conjunction then $G, u$ and $H, v$ must differ on at least one of the conjuncts, so we may assume that $\varphi$ is of the form $(\exists N x_i)\psi$. We may assume that $x_i$ is the currently unassigned variable $x_{k+1}$. On the first move of the game Player I picks up the pair of $x_{k+1}$ pebbles and chooses a set of $N$ vertices $g$, such that $G, u(x_{k+1}/g) \models \psi$. Whatever Player II chooses as $B$ there will be at least one vertex $h \in B$ such that $H, v(x_{k+1}/h) \models \neg\psi$. Player I puts his pebble number on this $h$. Player II must respond with some $g \in A$. Now $G, u(x_{k+1}/g)$ and $H, v(x_{k+1}/h)$ differ on $\psi \in \mathcal{C}_{k+1,m-1}$. Thus by induction Player II loses the remaining $m-1$ move game.

$(1 \Rightarrow 3)$ : Suppose that $W^m(u) = W^m(v)$. It follows that Equation 5.1 holds for each $k$-tuple of colors $\bar{t}$. Clearly Player I's strongest move involves the presently unused pair of $x_{k+1}$ pebbles. Suppose he picks them up and chooses a set $A$ of $N$ vertices from $G$. For each $\bar{t}$, let $N_{\bar{t}}$ be the number of vertices $g \in A$ such that $\bar{t} = \text{sift}(W^{m-1}, u, g)$. It follows from Equation 5.1 that Player II can put $N_{\bar{t}}$ vertices $h$ into $B$ such $\bar{t} = \text{sift}(W^{m-1}, v, h)$.[9]

In the second part of the move Player I will put $x_{k+1}$ on some $h \in B$. Player II should then answer with a $g \in A$ such that

$$\text{sift}(W^{m-1}, u, g) = \text{sift}(W^{m-1}, v, h) = \bar{t}$$

[8] For the case $k = 1$ we must explicitly consider adjacency and so the formula is $(\exists N\, x_2)(E(x_1, x_2) \wedge \psi_{t_1}(x_1/x_2))$.

[9] In the case $k = 1$, Player II must choose $h$'s that are adjacent to $v(x_1)$ iff the corresponding $g$'s are adjacent to $u(x_1)$.

Consider the remaining game on configuration $(u(x_{k+1}/g), v(x_{k+1}/h))$. Note that Player II has not yet lost. At the beginning of the next move, Player I will choose some pair of pebbles $x_i$ and pick them up. Now we know that the remaining configurations have the same $W^{m-1}$ color. It follows by induction that Player II wins the remaining $(m-1)$-move game. ∎

The following observation will be useful in the proof of our main theorem.

**Observation 5.3** *If Player I has a winning strategy for the $m$-move $\mathcal{C}_k$ game on $G, H$, then he has a winning strategy in which throughout the game he only chooses monochromatic sets $A$.*

**Proof** We saw in the above proof that whenever Player I chooses a set $A$, this set may be partitioned according to the $k$-tuple of color classes induced. Player II then answers separately for each $k$-tuple of colors. If Player II does not have the right number of elements in one of these classes then she will lose, and Player I need only have selected his elements from that class. Each of these classes is monochromatic. ∎

It is not hard to see using standard coloring algorithms, cf. [1, §4.13], that

**Fact 5.4 ([27])** *The stable colorings of $k$-tuples may be computed in $O(k^2 n^{k+1} \log n)$ steps on a RAM.*

It then follows from Theorem 5.2 that graphs that are identified by the language $\mathcal{C}_{k+1}$ have a canonization algorithm that runs in time $O(k^2 n^{k+1} \log n)$.

**Remark 5.5** *It is interesting to note that the language $\mathcal{L}_{k+1}$ enjoys a relationship similar to that of Theorem 5.2 with a variant of the $k$-dim W-L algorithm with the same time bound. The only difference is that the computation of the new color treats the following as a set instead of a multiset:*

$$\{\text{sift}(W^{m-1}, u, g) \mid g \in V_G\}$$

*That is, after sorting the collection of $k$-tuples, we eliminate duplicates.*

# 6 Construction

We construct our counterexample graphs by starting with low degree graphs having only linear size separators. We replace each vertex $v$ of degree $k$ in such a graph by the graph $X_k$, defined as follows: $X_k = (V_k, E_k)$, where

$$\begin{aligned} V_k &= A_k \cup B_k \cup M_k \text{ where } A_k = \{a_i \mid 1 \le i \le k\}, \\ &\quad B_k = \{b_i \mid 1 \le i \le k\}, \text{and} \\ &\quad M_k = \{m_S \mid S \subseteq \{1, \ldots, k\}, |S| \text{ is even}\} \\ E_k &= \{(m_S, a_i) \mid i \in S\} \cup \{(m_S, b_i) \mid i \notin S\} \end{aligned}$$

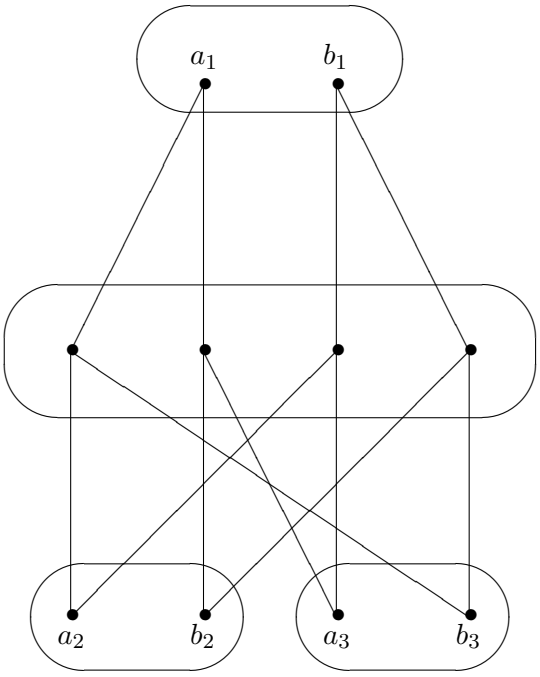


Figure 3: The Graph $X_3$

Thus $X_k$ consists of a set of $2^{k-1}$ vertices in the middle each connected to one vertex from each of the pairs $\{a_i, b_i\}$, $1 \leq i \leq k$. Furthermore, each of the middle vertices is connected to an even number of $a_i$'s. (We will assume that the middle vertices ($M_k$) of $X_k$ have a different color, say magenta, from the others ($A_k \cup B_k$). Furthermore, the pairs $a_i$ and $b_i$ should be able to recognize their mates. If necessary, add vertices $c_i$ colored chartreuse, with edges to $a_i$ and $b_i$.) See Figure 3 for a diagram of $X_3$.

The following lemma describes the relevant property of the graph $X_k$. The proof is immediate.

**Lemma 6.1** *Suppose that we color the vertices $a_i$ and $b_i$ of graph $X_k$ with the color $i$. (Thus all automorphisms of $X_k$ must fix the sets $\{a_1, b_1\}, \ldots, \{a_k, b_k\}$.) Then there are exactly $2^{k-1}$ automorphisms of $X_k$. Each is determined by interchanging $a_i$ and $b_i$ for each $i$ in some subset $S$ of $\{1, \ldots, n\}$ of even cardinality.*

Let $G$ be a finite, connected, undirected graph such that every vertex of $G$ has degree at least two. Define the graph $X(G)$ ("$X$ of $G$") as follows. For each vertex $v$ of $G$, we replace $v$ by a copy of $X_k$, call it $X(v)$, where $k$ is the degree of $v$. To each edge $(v, w)$ of $v$ we associate one of the pairs $\{a_i, b_i\}$ from $X(v)$, call this pair $a(v, w)$ and $b(v, w)$. Finally, we connect the $a$ vertices and the $b$ vertices at each end of each edge, that is we draw the edges $(a(u, v), a(v, u))$ and $(b(u, v), b(v, u))$. If $G$ is a colored graph, then each vertex in $X(v)$ should inherit the color of $v$. Next, define the graph $\tilde{X}(G)$ ("$X$ twist of $G$") as follows: In the above construction of $X(G)$ arbitrarily choose one edge $(v, w)$ and twist it, that is reverse the connections, drawing edges $(a(u, v), b(v, u))$ and $(b(u, v), a(v, u))$. In the next lemma we show some relevant properties of $X(G)$ and $\tilde{X}(G)$, including the fact that $\tilde{X}(G)$ is well defined.

**Lemma 6.2** *Let $G$ be any finite, connected graph such that every vertex of $G$ has degree at least two. Let $X(G)$ and $\tilde{X}(G)$ be as above. Let $\hat{X}(G)$ be constructed like $X(G)$, but with exactly $t$ of its edges twisted. Then $\hat{X}(G)$ is isomorphic to $X(G)$ iff $t$ is even, and $\hat{X}(G)$ is isomorphic to $\tilde{X}(G)$ iff $t$ is odd.*

**Proof** First observe the following fact about $\hat{X}(G)$. Let $v$ be any vertex of $G$, and let $(x,v),(y,v)$ be any two edges incident at $v$. If in $\hat{X}(G)$ we twist both $(x,v)$ and $(y,v)$, then the resulting graph is isomorphic to $\hat{X}(G)$. (This is immediate from Lemma 6.1.)

Now suppose that the number of twists in $t$ is greater than or equal to two. The above observation lets us move the twists towards each other until they overlap and cancel each other out. Thus if $t$ is even then $\hat{X}(G)$ is isomorphic to $X(G)$, otherwise it is isomorphic to $\tilde{X}(G)$.

It remains to show that $X(G)$ is not isomorphic to $\tilde{X}(G)$. Assume for the sake of a contradiction that $\varphi$ is an isomorphism from $X(G)$ to $\tilde{X}(G)$. Consider the action of $\varphi$ on any pair $\{a(v,w),b(v,w)\} \subset X(v)$, for $(v,w)$ an edge of $G$. Because of the colorings in the definition of $X_k$, $\varphi$ must map the pair $\{a(v,w),b(v,w)\}$ to some $\{a(v',w'),b(v',w')\}$ in $\tilde{X}(G)$, and thus $\varphi$ also maps $\{a(w,v),b(w,v)\}$ to $\{a(w',v'),b(w',v')\}$. Define $\oplus\varphi$ to be the sum mod 2 over all such pairs in $X(G)$ of the number of times $\varphi$ maps an $a$ to a $b$. Clearly if we consider the two pairs corresponding to every edge $(x,y)$ in $G$, the number of such switches is either zero or two, except for the unique edge chosen in the construction of $\tilde{X}(G)$, where the number is one. Hence $\oplus\varphi$ is one mod 2. Now let's consider the mod 2 sum in another way, namely in terms of each copy of $X_k$ in $X(G)$. By Lemma 6.1, it is immediate that $\oplus\varphi$ is zero mod 2. This contradiction proves the lemma. ∎

**Definition 6.3** A *separator* of a graph $G=(V,E)$ is a subset $S \subset V$ such that the induced subgraph on $V-S$ has no connected component with more than $|V|/2$ vertices.

We now prove our main theorem:

**Theorem 6.4** *Let $T$ be a graph such that every separator of $T$ has at least $s+1$ vertices. Then*

$$X(T) \equiv_{\mathcal{C}_s} \tilde{X}(T) \ .$$

**Proof** By Theorem 5.2, it suffices to give a winning strategy for Player II in the $\mathcal{C}_s$ game on $X(T)$ and $\tilde{X}(T)$. We will assume that the original graph $T$ has color class size one. The graphs $X(T)$ and $\tilde{X}(T)$ inherit these colors and so have color class size $2^{k-1}$, where $k$ is the maximum degree of any vertex in $T$. This only makes life more difficult for Player II.

We know by Lemma 6.2 that if we add a twist to any edge of $X(T)$, then the resulting graph is isomorphic to $\tilde{X}(T)$. After the $r^{\text{th}}$ move of the game, let $Q_r$ be the largest connected component in $T-P_r$ where $P_r$ is the set of vertices $g \in T$ such that just after the $r^{\text{th}}$ move there is a pebble on a vertex of $X(g)$ in $X(T)$. Since $T$ has no $s$ separator, we know that $Q_r$ contains over half the vertices of $T$. Player II's winning strategy will be to maintain the following property:

(∗) For each vertex $g \in Q_r$, let $X^g(T)$ be $X(T)$ with an edge adjacent to $g$ twisted. Then there exists an isomorphism $\alpha_{r,g}$ from $X^g(T)$ to $\tilde{X}(T)$, such that for all $i \leq s$, $\alpha_{r,g}$ maps the vertex under pebble $x_i$ in $X(T)$ to the vertex under pebble $x_i$ in $\tilde{X}(T)$.

The difference between $X(T)$ and $\tilde{X}(T)$ is that the latter graph has one twisted edge. An intuitive explanation of Player II's winning strategy is that she keeps this twisted edge inside of $Q_r$. With only $s$ pebbles, Player I cannot break apart $Q_r$ to expose the twist.

Clearly if Player II can maintain (∗), then the map from the pebbled points in $X(T)$ to the corresponding pebbled points in $\tilde{X}(T)$ is an isomorphism, and she wins. We show by induction on $r$, that Player II can maintain (∗). First let us make a remark about Player I's moves. By Observation 5.3, it always suffices for Player I to restrict himself to choosing a set of monochromatic points at each move. Notice that, if Player I chooses a vertex in $M(h)$, the middle of an $X(h)$, then all the other vertices in that $X(h)$ are determined. Furthermore, since one point in $M(h)$ determines all of $M(h)$, it suffices for Player I to choose only a single point at a time. (Thus counting does not help at all in distinguishing $X(T)$ from $\tilde{X}(T)$!)

Player II's inductive strategy can now be stated. Assume (∗) holds, and suppose that on move $r+1$ Player I picks up pebble $x_i$ and puts it down on a vertex in $M(w)$. Note that a new largest component $Q_{r+1}$ is determined. Let $g$ be a vertex in $Q_r \cap Q_{r+1}$. Player II's response is to answer Player I's move according to the isomorphism $\alpha_{r,g}$. To maintain (∗), let $\alpha_{r+1,g} = \alpha_{r,g}$. Since there is a pebble-free path from $g$ to every other vertex in $Q_{r+1}$, the proof of Lemma 6.2 shows us how to define all the other isomorphisms, $\alpha_{r+1,h}$, $h \in Q_{r+1}$. ∎

**Corollary 6.5** *There exists a sequence of pairs of graphs $\{G_n, H_n\}$, $n \in \mathbf{N}$ admitting a linear time canonical labeling algorithm and having the following additional properties:*

1. *$G_n$ and $H_n$ have $O(n)$ vertices.*
2. *$G_n$ and $H_n$ have degree three and color class size four.*
3. *$G_n \equiv_{\mathcal{C}_n} H_n$.*
4. *$G_n$ is not isomorphic to $H_n$.*

**Proof** This follows immediately from Theorem 6.4 when we let $G_n = X(T_n)$ and $H_n = \tilde{X}(T_n)$ where the $T_n$'s are a sequence of degree three graphs of separator size $n$, with each vertex of $T_n$ colored a unique color. Such graphs are well known to exist, see for example [2]. ∎

# 7 Corollaries

A long time ago, one of us showed that there is a polynomial-time property of graphs that requires $\Omega(2^{\sqrt{\log n}})$ quantifiers to be expressed in first-order logic without ordering. That proof also used the graphs $X(D_n)$ and $\tilde{X}(D_n)$, for a certain sequence of degree three graphs

$\{D_n\}$ [22, Theorem 7]. Now, Corollary 6.5 improves that lower bound to $\Omega(n)$ variables.[10] It also shows graphically that if we exclude the ordering relation from inductive first-order logic, then the addition of counting does not suffice to express all polynomial-time graph properties. In particular, we have the following:

**Corollary 7.1** *Let $\Gamma$ be the set of all graphs of the form $X(G)$, or $\tilde{X}(G)$, for all graphs $G$ of degree at most three and color class size one. Then the isomorphism problems for graphs in $\Gamma$ is expressible in first-order logic with ordering and sum mod 2, but it is not expressible by any sequence of first-order sentences from $\mathcal{C}_{r(n)}$ (without ordering), where $r(n) = o(n)$.*

**Remark 7.2** *In particular, inductive logic with counting, but without ordering does not contain all the polynomial-time computable graph properties. In fact, it does not even contain all such properties computable by a uniform sequence of bounded-depth, polynomial-size Boolean circuits that include parity gates, cf. [12].*

**Proof** We have seen in Corollary 6.5 that the graphs $X(T_n)$ and $\tilde{X}(T_n)$ are indistinguishable in $\mathcal{C}_{\epsilon n}$ for some constant $\epsilon > 0$. Suppose for the sake of a contradiction that there were a sentence $\sigma \in (\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT})$ that expresses the isomorphism property for graphs from $\Gamma$. That is for graphs $G, H \in \Gamma$,

$$(G, H) \models \sigma \quad \Leftrightarrow \quad G \cong H$$

Let $k$ be the number of distinct variables occurring in $\sigma$. For graphs of size $n$, let $\sigma_n$ be the unwinding of $\sigma$ as follows. Rewrite any least fixed points of arity $a$, $(\mathrm{LFP}\varphi)$ as $\varphi^{(n^a)}(\emptyset)$. Next replace any quantified number variable $\exists i$ (respectively, $\forall i$) by a disjunction $\bigvee_{i=0}^{n-1}$ (respectively, by a conjunction $\bigwedge_{i=0}^{n-1}$). Note that $\sigma_n \in \mathcal{C}_k$ and is equivalent to $\sigma$ for structures of size at most $n$.

Thus we have that $\sigma_n$ distinguishes the pair $P = (X(T_n), \tilde{X}(T_n))$ from the pair $Q = (X(T_n), X(T_n))$. It follows that Player I wins the $\mathcal{C}_k$ game on these two pairs. Note that Player II can match any vertex in the first $X(T_n)$ from $P$ with the same vertex in the first $X(T_n)$ from $Q$. Thus, Player I must have a winning strategy for the $\mathcal{C}_k$ game on $X(T_n)$ and $\tilde{X}(T_n)$. This contradiction shows that isomorphism for graphs from $\Gamma$ is not expressible in $(\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT})$.

We next show that we can distinguish $X(G)$ from $\tilde{X}(G)$ in first-order logic with ordering and sum mod 2. This is easy. The ordering gives us a way to mark each of the pairs $a(g, h)$ and $b(g, h)$ in the graphs. Let $a(g, h)$ be the first of the pair, and $b(g, h)$ the second. (Note that since the vertices in $M(g)$ and $M(h)$ inherit unique colors from $g$ and $h$, we are given as part of the input which pair of vertices is $a(g, h), b(g, h)$.) Now, given this assignment of $a$'s and $b$'s, a simple first-order sentence asserts that $X(g)$ is straight (i.e. isomorphic to $X_3$) or twisted (i.e. each vertex in $M(g)$ is adjacent to an odd number of $a$'s). Now, the graph is isomorphic to $X(G)$ iff the sum mod 2 of the number of twisted vertices and edges is 0, and it's isomorphic to $\tilde{X}(G)$ iff the sum mod 2 is 1.

[10] This is a major improvement because $n$ is much bigger than $2^{\sqrt{\log n}}$, and because a sentence with $q$ quantifiers can make use of at most $q$ variables, but a sentence with $v$ variables can make use of $2^{n^v}$ quantifiers.

Of course, if $G \neq H$, then since these graphs have color class size one, $X(G)$ and $X(H)$ can be distinguished by a sentence in $\mathcal{L}_2$. Thus isomorphism for graphs from $\Gamma$ is expressible in $\mathrm{AC}^0$ plus parity gates, as claimed. ∎

The next result proves a straightforward upper bound that nearly matches our lower bound on the number of variables needed to identify a class $\Delta$ of graphs as a function of the separator size of members of $\Delta$.

**Proposition 7.3** *Let $\Delta$ be a set of graphs closed under induced subgraphs, such that every graph $G \in \Delta$ has a separator of size at most $s(n)$, where $n$ is the number of vertices of $G$. Then $\Delta$ is identified by $\mathcal{C}_{V(n)}$ where*

$$V(n) = 3 + \sum_{i=0}^{\lfloor \log n \rfloor} s(\lfloor n 2^{-i} \rfloor) \; .$$

*(In particular, $V(n) \leq s(n) \log n$, and if $s(n) = n^\alpha$, then $V(n) = O(s(n))$.)*

**Proof** We use induction on $n$, the number of vertices of $G$. Given $G$, we can first say that there exist vertices $x_1, \ldots, x_{s(n)}$ such that every connected component of $G - \{x_i | 1 \leq i \leq s(n)\}$ has size at most $\lfloor n/2 \rfloor$. This is expressible in $s(n) + 3$ variables. Next we assert how many connected components of each isomorphism type there are. This requires $V(\lfloor n/2 \rfloor)$ variables, in addition to the $s(n)$ that we leave on $x_1, \ldots, x_{s(n)}$. ∎

# 8 Conclusions and Open Questions

1. We redirect the reader's attention to Questions 3.2 and 3.3. We have shown in Corollary 6.5 that first-order logic plus counting and least fixed point, but without ordering, fails badly. The question,"What besides counting must be added to FO + LFP to get all polynomial-time graph problems?" is worthy of much study, cf. [27, 20].

2. Planar graphs have separators of size $O(\sqrt{n})$, and thus by Proposition 7.3 they can be identified in $\mathcal{C}_{\sqrt{n}}$. However, Theorem 6.4 does not give a matching lower bound because even if $G$ is planar, the graph $X(G)$ need not be. We would like to know if $\Omega(\sqrt{n})$ variables are necessary to identify planar graphs.

**Acknowledgements:** Thanks to Sandeep Bhatt who improved our results by pointing out that the essential property of the counterexample graphs we were using was that their separators are large. Thanks to Laci Babai for informing us about the status of the research on the W-L method in the Soviet Union.

# References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley (1974).

[2] M. Ajtai, "Recursive Construction for 3-Regular Expanders," *28th IEEE Symp. on Foundations of Computer Science* (1987), 295-304.

[3] László Babai, "Monte Carlo Algorithms in Graph Isomorphism Testing," Tech. Rep. DMS 79-10, Université de Montréal, 1979.

[4] László Babai, "On the Complexity of Canonical Labeling of Strongly Regular Graphs," *SIAM J. Computing* **9** (1980), 212-216.

[5] László Babai, "Moderately Exponential Bound for Graph Isomorphism," *Proc. Conf. on Fundamentals of Computation Theory,* Lecture Notes in Computer Science, Springer, 1981, 34-50.

[6] László Babai, "On the Order of Uniprimitive Permutation Groups," *Annals of Math.* **113** (1981), 553-568.

[7] László Babai, *"Permutation Groups, Coherent Configurations, and Graph Isomorphism,"* D.Sc. Thesis, Hungarian Acad. Sci., 1984 (in Hungarian).

[8] László Babai, Paul Erdős, and Stanley M. Selkow, "Random Graph Isomorphism," *SIAM J. on Comput.* **9** (1980), 628-635.

[9] L. Babai, W.M. Kantor, and E.M. Luks, "Computational Complexity and the Classification of Finite Simple Groups," *24th IEEE Symp. on Foundations of Computer Science* (1983), 162-171.

[10] László Babai and Luděk Kučera, "Canonical Labelling of Graphs in Linear Average Time," *20th IEEE Symp. on Foundations of Computer Science* (1979), 39-46.

[11] László Babai and Eugene M. Luks, "Canonical Labeling of Graphs," *15th ACM Symposium on Theory of Computing* (1983), 171-183.

[12] David Mix Barrington, Neil Immerman, and Howard Straubing, "On Uniformity Within $NC^1$," *J. Comput. System Sci.* **41**, No. 3 (1990), 274-306.

[13] László Babai and Rudi Mathon, Talk at the South-East Conference on Combinatorics and Graph Theory, 1980.

[14] P.J. Cameron, "6-Transitive Graphs," *J. Combinat. Theory B* **28**, (1980), 168-179.

[15] Ashok Chandra and David Harel, "Structure and Complexity of Relational Queries," *J. Comput. System Sci.* **25** (1982), 99-128.

[16] A. Ehrenfeucht, "An Application of Games to the Completeness Problem for Formalized Theories," *Fund. Math.* **49** (1961), 129-141.

[17] R. Fraïssé, "Sur quelques classifications des systèms de relations," *Publ. Sci. Univ. Alger* **1** (1954), 35-182.

[18] Martin Fürer, Walter Schnyder, and Ernst Specker, "Normal Forms for Trivalent Graphs and Graphs of Bounded Valence," *15th ACM Symposium on Theory of Computing* (1983), 161-170.

[19] Ya. Yu. Gol'fand and M.H. Klin, "On $k$-Regular Graphs," in *Algorithmic Research in Combinatorics,* Nauka Publ., Moscow, 1978, 76-85.

[20] Yuri Gurevich, "Logic and the Challenge of Computer Science," in *Current Trends in Theoretical Computer Science,* ed. Egon Börger, Computer Science Press, 1988, 1-57.

[21] D.G. Higman, "Coherent Configurations I.: Ordinary Representation Theory," *Geometriae Dedicata* **4** (1975), 1-32.

[22] Neil Immerman, "Number of Quantifiers is Better than Number of Tape Cells," *J. Comput. System Sci.* **22**, No. 3 (1981), 384-406.

[23] Neil Immerman, "Upper and Lower Bounds for First Order Expressibility," *J. Comput. System Sci.* **25**, No. 1 (1982), 76-98.

[24] Neil Immerman, "Relational Queries Computable in Polynomial Time," *Information and Control* **68** (1986), 86-104.

[25] Neil Immerman, "Languages That Capture Complexity Classes," *SIAM J. Computing* **16**, No. 4 (1987), 760-778.

[26] Neil Immerman and Dexter Kozen, "Definability with Bounded Number of Bound Variables," *Information and Computation* **83** (1989), 121-139.

[27] Neil Immerman and Eric S. Lander, "Describing Graphs: A First-Order Approach to Graph Canonization," in *Complexity Theory Retrospective,* Alan Selman, ed., Springer-Verlag, 1990, 59-81.

[28] N. Immerman, S. Patnaik, and D. Stemple, "The Expressiveness of a Family of Finite Set Languages," *Tenth ACM Symposium on Principles of Database Systems* (1991), 37-52.

[29] Luděk Kučera, "Canonical Labeling of Regular Graphs in Linear Average Time," *28th IEEE Symp. on Foundations of Computer Science* (1987), 271-279.

[30] M.H. Klin, M.E. Muzichuk, and I.A. Faradžev, "Cellular Rings and Groups of Automorphisms of Graphs," Introductory Article to a Book to be Published by D. Reidel Publ. Co.

[31] M.Ch. Klin, R. Pöschel, and K. Rosenbaum, "Angewandte Algebra," Vieweg & Sohn Publ., Braunschweig 1988.

[32] R. Lipton, "The Beacon Set Approach to Graph Isomorphism," Yale Dept. Comp. Sci. preprint No. 135, 1978.

[33] Eugene M. Luks, "Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time," *J. Comput. System Sci.* **25** (1982), 42-65.

[34] R. Mathon, "A Note On the Graph Isomorphism Counting Problem," *Inform. Proc. Let.* **8** (1979), 131-132.

[35] B.D. McKay, "Nauty User's Guide (Version 1.2)," Tech. Rep. TR-CS-87-03, Dept. Computer Science, Austral. National Univ., Melbourne, 1987.

[36] G.L. Miller, "On the $n^{\log n}$ Isomorphism Technique," *10th ACM Symposium on Theory of Computing* (1978), 51-58.

[37] R.C. Read and D.G. Corneil, "The Graph Isomorphism Disease," *J. Graph Theory* **1** (1977), 339-363.

[38] M. Vardi, "Complexity of Relational Query Languages," *14th ACM Symposium on Theory of Computing* (1982), 137-146.

[39] Boris Weisfeiler, ed., *On Construction and Identification of Graphs,* Lecture Notes in Mathematics 558, Springer, 1976.

[40] B. Weisfeiler and A.A. Lehman, "A Reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction," (in Russian), *Nauchno-Technicheskaya Informatsia, Seriya 2,* **9** (1968), 12-16.

[41] V.N. Zemlyachenko, N. Kornienko, and R.I. Tyshkevich, *"Graph Isomorphism Problem,"* (in Russian), The Theory fo Computation I, Notes Sci. Sem. LOMI 118, 1982.