1

Describing Graphs: a First-Order Approach to Graph Canonization

Neil Immerman[1]
Eric Lander[2]

ABSTRACT In this paper we ask the question, "What must be added to first-order logic plus least-fixed point to obtain exactly the polynomial-time properties of unordered graphs?" We consider the languages $\mathcal{L}_k$ consisting of first-order logic restricted to k variables and $\mathcal{C}_k$ consisting of $\mathcal{L}_k$ plus "counting quantifiers". We give efficient canonization algorithms for graphs characterized by $\mathcal{C}_k$ or $\mathcal{L}_k$. It follows from known results that all trees and almost all graphs are characterized by $\mathcal{C}_2$.
This paper appeared in *Complexity Theory Retrospective,* Alan Selman, ed., Springer-Verlag (1990), 59-81.

1.1 Introduction

In this paper we present a new and different approach to the graph canonization and isomorphism problems. Our approach involves a combination of complexity theory with mathematical logic. We consider first-order languages for describing graphs. We define what it means for a language to characterize a set of graphs (Definition 1.4.2). We next define the languages $\mathcal{L}_k$ (resp. $\mathcal{C}_k$) consisting of the formulas of first-order logic in which only k variables occur (resp. $\mathcal{L}_k$ plus 'counting quantifiers'). We then study which sets of graphs are characterized by certain $\mathcal{L}_k$'s and $\mathcal{C}_k$'s. It follows by a result of Babai and Kučera [4] that the language $\mathcal{C}_2$ characterizes almost all graphs. We also show that $\mathcal{C}_2$ characterizes all trees. In Section 1.9 we give a simple $O[n^k \log n]$ step algorithm to test if two graphs G and H on

---

[1]Computer Science Dept., University of Massachusetts, Amherst, MA 01003.
Research supported by NSF grants DCR-8603346 and CCR-8806308. This paper appeared in in *Complexity Theory Retrospective,* Alan Selman, ed., Springer-Verlag (1990), 59-81.
[2]Whitehead Institute, Cambridge, MA 02142.
Research supported by grants from the National Science Foundation (DCB-8611317) and from the System Development Foundation (G612).

n vertices agree on all sentences in $\mathcal{L}_k$, or $\mathcal{C}_k$. If G is characterized by $\mathcal{L}_k$ (or $\mathcal{C}_k$), a variant of this algorithm computes a canonical labeling for G in the same time bound.

This line of research has two main goals. First, finding a language appropriate for graph canonization is a basic problem, central to the first author's work on descriptive computational complexity. We will explain this setting in the next section.

A canonization algorithm for a set of graphs, S, gives a unique ordering (canonical labeling) to each isomorphism class from S. Thus two graphs from S are isomorphic if and only if they are identical in the canonical ordering. The second goal of this work is to describe a simple and general class of canonization algorithms. We hope that variants of these algorithms will be powerful enough to provide simple canonical forms for all graphs; and do so without resorting to the the high powered group theory needed in the present, best graph isomorphism algorithms [27, 3].

1.2 Descriptive Complexity

In this section we discuss an alternate view of complexity in which the complexity of the descriptions of problems is measured. This approach has provided new insights and techniques to help us understand the standard complexity notions: time, memory space, parallel time, number of processors. The motivations for the present paper come from Descriptive Complexity. We can only sketch this area here. The interested reader should consult [24] for a more extensive survey.

Given a property, S, one can discuss the computational complexity of checking whether or not an input satisfies S. One can also ask, "What is the complexity of *expressing* the property S?" It is natural that these two questions are related. However, it is startling how closely tied they are when the second question refers to expressing the property in first-order logic. We will now describe the first-order languages in detail. Next we will state some facts relating descriptive and computational complexity.

First-Order Logic

In this paper we will confine our attention to graphs and properties of graphs. Thus when we mention complexity classes P, NP, etc. we will really be refering to those problems of ordered graphs that are contained in P, NP, etc. (If you want to know why the word "ordered" was included in the previous sentence, please read on. One of the main concerns of this paper is how to remove the need to order the vertices of graphs.)

For our purposes, a *graph* will be defined as a finite logical structure, $G = \langle V, E\rangle$. V is the universe (the vertices); and E is a binary relation on V (the edges). As an example, the undirected graph, $G_1 = \langle V_1, E_1\rangle$,

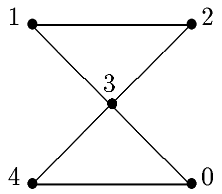

FIGURE 1.1. An Undirected Graph

pictured in Figure 1 has vertex set $V_1 = \{0, 1, 2, 3, 4\}$, and edge relation $E_1 = \{\langle 0,3\rangle, \langle 0,4\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \ldots, \langle 3,2\rangle, \langle 3,4\rangle\}$ 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.

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:

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

φ says that G is undirected and loop free. Unless we specifically say otherwise, we will assume that all graphs, G, in this paper satisfy φ, in symbols: $G \models \varphi$.

It is useful to consider an apparently[3] more general set of structures. The *first-order language of colored graphs* consists of the addition of a countable set of unary relations $\{C_1, C_2, \ldots\}$ to the first-order language of graphs. 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. (A vertex of a colored graph may satisfy zero, one, or several of the color relations. However, we will say that two vertices are the same color iff they satisfy the same *set* of color relations. Thus, by increasing the number of color relations we may assume that each vertex satisfies a unique color relation.)

[3]Colorings can be simulated in uncolored graphs by attaching gadgets. For example, a colored graph G with colors green and yellow can be modelled as a graph G' with some auxilliary vertices so that in G' each vertex v from G is now connected to either a triangle, or a square, or a pentagon, or a hexagon, according as v is green, yellow, green and yellow, neither green nor yellow. All mention of color predicates in this paper can be removed in this way.

RESULTS FROM DESCRIPTIVE COMPLEXITY

The ordering $0 < 1 < \cdots < n-1$ of the vertices of a graph is irrelevant to a "graph property". Unfortunately however, it is impossible to present an unordered graph to a computer. The vertices and edges must be presented in some order. Furthermore, many algorithms compute graph properties by visiting the vertices of the graph in some order which depends on this arbitrary ordering.

Let FO($\leq$) denote the first-order language of ordered graphs. This language includes the logical relation $\leq$ which must be interpreted as the usual ordering on the vertices, $V = \{0, 1, \ldots, n-1\}$. We will see in Fact 1.2.8 that FO($\leq$) is contained in CRAM[1] – the set of properties that can be checked by a concurrent, parallel, random access machine in constant time, using polynomially many processors.

In order to express a richer class of problems we consider uniform[4] sequences of first-order sentences, $\{\varphi_i\}_{i\in\mathbf{N}}$, where the sentence φ_n expresses the property in question for graphs with at most n vertices.

Let $\mathrm{FO}(\leq)[t(n)] - \mathrm{VAR}[v(n)]$ be the set of problems expressible by a uniform sequence of first-order sentences such that the n^{th} sentence has length $O[t(n)]$ and uses at most $v(n)$ distinct variables. The following fact says that the set of polynomial-time recognizable properties of ordered graphs is equal to the set of properties expressible by uniform sequences of first-order sentences of polynomial length using a bounded number of distinct variables.

Fact 1.2.1 ([18])

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

The Least Fixed Point (LFP) operator has long been used by logicians to formalize the power to define new relations by induction, cf. [28]. In [19] and in [31] it is shown that the uniform sequence of formulas in Fact 1.2.1 can be represented by a single use of LFP applied to a first-order formula. Thus,

Fact 1.2.2

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

---

[4]The uniformity in question can be purely syntactical, i.e. the n^{th} sentences of a FO$[t(n)]$ property consists of a fixed block of quantifiers repeated $t(n)$ times followed by a fixed quantifier-free formula. Uniformity is discussed extensively in [6]. In this paper the reader may think of uniform as meaning that the map from n to φ_n is easily computable, e.g. in logspace.

Example 1.2.3 The *monotone circuit value problem* is an example of a complete problem for P which we will use to illustrate Fact 1.2.2, [13]. Instances of this problem consist of boolean circuits with only "and" and "or" gates and a unique output gate whose value as determined by the inputs is one.

We can code the monotone circuit value problem as a colored, directed graph with colors T, A, R as follows: inputs are colored *True* if they are on; other vertices are colored *And* if they are "and" gates; otherwise they are "or" gates. The unique output gate is colored *Root.* The obvious inductive definition of circuit value can be written in $(\mathrm{FO} + \mathrm{LFP})$ as follows:

$$\begin{aligned} \psi(V, x) \quad \equiv \quad & T(x) \;\vee\; \Big[(\exists y)(E(x,y) \wedge V(y)) \\ & \wedge\; \big(A(x) \rightarrow (\forall y)(E(x,y) \rightarrow V(y))\big)\Big] \end{aligned}$$

The intuitive meaning of $V(x)$ is that the value of node x is one. Thus $V(x)$ is true if x is an input that is on, or if x is an "or" gate and there exists a gate y such that $V(y)$ and y is an input to x, or if x is an "and" gate having at least one input, and all of its inputs, y, satisfy $V(y)$.

Fix a graph, C. The formula $\psi(V, x)$ induces an operator ψ_C from unary relations on the vertices of C to unary relations on the vertices of C as follows:

$$\psi_C(R) \;=\; \{z \mid C \models \psi(R, z)\}$$

Furthermore, the correct Value relation on the circuit C is the least fixed point of ψ_C, i.e. the smallest unary relation V such that $\psi_C(V) = V$. Use the notation $(\mathrm{LFP}\,\psi(V, x))$ to denote this least fixed point.

Let $(\exists! x)\alpha$ (there exists a unique x such that α) be an abbreviation for the formula,

$$(\exists x)(\alpha(x) \wedge (\forall y)(\alpha(y) \rightarrow y = x))$$

The circuit value problem can now be expressed as follows:

$$(\exists! w)(R(w)) \;\wedge\; (\exists w)(R(w) \wedge (\mathrm{LFP}\,\psi(V, x))(w))$$

$\square$

A particular kind of inductive operator that is worth studying on its own is the transitive closure operator (TC) introduced in [20]. Let $\varphi(\bar{x}, \bar{y})$ be any binary relation on k-tuples. Then $(\mathrm{TC}\,\varphi(\bar{x}, \bar{y}))$ denotes the reflexive, transitive closure of φ. The following was proved in [20] with the finishing touch proved in [21].

Fact 1.2.4

$$\mathrm{FO}(\leq, \mathrm{TC}) \;=\; \mathrm{NSPACE}[\log n]$$

Example 1.2.5 Consider the following complete problem for NSPACE[$\log n$]. The GAP problem consists of the set of directed graphs having a unique vertex colored S and a unique vertex colored T and such that there is a path from the S vertex to the T vertex. It is obvious how to express GAP in (FO + TC):

$$(\exists! x)(S(x)) \wedge (\exists! x)(T(x)) \wedge (\exists st)(S(s) \wedge T(t) \wedge (\mathrm{TC}\, E(x,y))(x,t))$$

□

It is interesting to examine the relationship between the number of variables needed to describe a problem, and the computational complexity of the problem. Let FO[$t(n)$]$-$VAR[v] be the restriction of FO[$t(n)$] to sentences with at most v distinct variables. Then the following bounds can be derived from the proof of Fact 1.2.1 in [18]:

Fact 1.2.6 *[18]*

$$\mathit{DTIME}[n^k] \;\subseteq\; \mathrm{FO}(\leq)[n^k] - \mathrm{VAR}[k+3] \;\subseteq\; \mathit{DTIME}[n^{2k+4}]$$

Thus the DTIME[n^k] properties of ordered graphs are *roughly* the properties expressible by first-order sentences with k variables and length n^k. Obviously this is very rough. A closer relationship between machine complexity and first-order expressibility is obtained if one takes into account the built in parallelism of quantifiers.

Let CRAM[$t(n)$]-PROC[$p(n)$] be the set of problems accepted by a concurrent-read, concurrent-write, parallel random access machine (CRAM) in parallel time $O[t(n)]$ using $O[p(n)]$ processors. In order to get a precise relationship with the CRAM model when $t(n) < \log n$ it was necessary to add another logical relation to FO. Since variables range over the universe $\{0, 1, \ldots, n-1\}$ they may be thought of as $\log n$ bit numbers. Let the relations BIT(x, y) be true just if the x^{th} bit in the binary expansion of y is a one. In [22] it is shown that FO$(\leq, \mathrm{BIT})[t(n)] - \mathrm{VAR}[O[1]]$ is exactly the set of properties checkable by a CRAM in parallel time $O[t(n)]$ using polynomially many processors. (In fact, FO$(\leq, \mathrm{BIT})[t(n)] - \mathrm{VAR}[v]$ corresponds to CRAM-TIME[$t(n)$] using *roughly* n^v processors.)

Fact 1.2.7 *For all* $t(n)$,

$$\mathrm{FO}(\leq, \mathrm{BIT})[t(n)] - \mathrm{VAR}[O[1]] \;=\; \mathrm{CRAM}[t(n)]\text{-}\mathrm{PROC}[n^{O[1]}]$$

In particular, we have that the first-order properties are those checkable in constant time by a CRAM using polynomially many processors,

Fact 1.2.8

$$\mathrm{FO}(\leq, \mathrm{BIT}) \;=\; \mathrm{CRAM}[1]\text{-}\mathrm{PROC}[n^{O[1]}]$$

1.3 Properties of (Unordered) Graphs

Facts 1.2.2 and 1.2.4 give natural languages expressing respectively the polynomial-time and nondeterministic logspace properties of ordered graphs. When the ordering is not present, it is possible to prove nearly optimal upper and lower bounds on the number of quantifiers and variables needed to express various properties in first-order logic.

For example, in [18] the graphs Y_k and N_k are constructed. These graphs have the property that Y_k has a complete subgraph on k vertices, but N_k does not. However using Ehrenfeucht-Fraisse games (cf. Section 1.6) one can show that Y_k and N_k agree on all sentences with $k-1$ variables but without ordering. It thus follows that k variables are necessary and sufficient to express the existence of a complete subgraph of size k. If these bounds applied to the languages with ordering they would imply that $\mathrm{P} \neq \mathrm{NP}$.

In [17] there is a similar construction of a sequence of pairs of graphs which differ on a polynomial-time complete property, but agree on all sentences of poly-logarithmic length without ordering. If this result went through with the ordering it would follow that $\mathrm{NC} \neq \mathrm{P}$, and in particular that NSPACE[$\log n$] is not equal to P.

The reason these arguments do not go through with ordering is as follows. For any constant c, there is a very simple formula[5] with ordering, $\alpha_c(x)$, that holds just for x equal to the c^{th} vertex. It follows that whenever two graphs agree on all simple sentences with ordering, they are equal.

It is of great interest to understand the role of ordering and if possible to replace the ordering with a more benign construction. Furthermore, the most basic problem on which to study the role of ordering is graph isomorphism. If two graphs differ on any property they are certainly not isomorphic!.

Let a *graph property* be an order independent property of ordered graphs. One can ask the question,

Question 1.3.1 *Is there a natural language for the polynomial-time graph properties?*

Gurevich has conjectured that the answer to Question 1.3.1 is, “No,” [14]. An affirmative answer to this question would imply a similar answer to the more basic,

Question 1.3.2 *Is there a recursively enumerable listing of all polynomial-time graph properties?*

Questions 1.3.1 and 1.3.2 are important in various settings. It is well

[5] More precisely, the formula has 3 variables and length $O[\log n]$.

known that graphs are the most general logical structures.[6] Thus these questions are equivalent to the corresponding questions concerning relational databases: i.e. give a database query language for expressing exactly the polynomial-time queries that are independent of the arbitrary ordering of tuples, cf. [9]. We believe that the answers to Questions 1.3.1 and 1.3.2 are both, "Yes," and we ask the more practical,

Question 1.3.3 *What must we add to first-order logic after taking out the ordering so that Fact 1.2.2 remains true? Put another way, describe a language $\mathcal{L}$ that expresses exactly the polynomial-time graph properties.*

The ordering relation is crucial for simulating computation: a Turing machine will be given an input graph in some order. It will visit the vertices of the graph using this ordering; and it is difficult to see how to simulate an arbitrary computation without reference to this ordering. It is well known that first-order logic without ordering is *not* strong enough to express computation. Let EVEN be the set of graphs with an even number of vertices. We will show in Proposition 1.6.4 that the property EVEN requires n variables for graphs with n vertices. (For a property to be expressible in FO+LFP a necessary condition is that it is expressible in a constant number of variables independent of n.)

In view of Proposition 1.6.4, it is natural to add the ability to count to first-order logic without ordering. This is formalized in Section 1.7, where we define the languages $\mathcal{C}_k$ of first-order logic restricted to k distinct variables, plus "counting quantifiers". We show in Corollary 1.8.5 that the very simple language $\mathcal{C}_2$ suffices to give unique descriptions and thus efficient canonical forms for almost all graphs.

For a long time we suspected that first-order logic plus least fixed point and counting was enough to express all polynomial-time graph properties. It would have immediately followed that for each polynomial-time graph property Q there would be a fixed k such that for all n, the property Q restricted to graphs of size n is expressible in $\mathcal{C}_k$. In particular, if our suspicion were right, then for every set of graphs S admitting a polynomial-time graph isomorphism algorithm, there would exist a fixed k such that $\mathcal{C}_k$ characterizes S (to be defined later). This implies that for any two graphs G and H from S, if G and H are $\mathcal{C}_k$ equivalent (i.e. G and H agree on all sentences from $\mathcal{C}_k$) then they are isomorphic. For example, the sets of graphs of bounded color class size (defined below) admit polynomial-time graph isomorphism algorithms. We show in Proposition 1.5.3 that the language $\mathcal{C}_3$ characterizes graphs of color class size 3. However, the following recent result shows in a strong way that no $\mathcal{C}_k$ characterizes the graphs of

[6]More precisely, every first-order language may be interpreted in the first-order theory of graphs. We would like to know to whom this is due, and where it appears in print.

color class size 4. Thus our suspicion was wrong: first-order logic plus least fixed point and counting does not express all the polynomial-time graph properties.

Fact 1.3.4 ([8]) *There exists a sequence of pairs of non-isomorphic graphs $\{G_n, H_n\}$ such that G_n and H_n have $O[n]$ vertices, color class size 4, and admit linear time and logspace canonization algorithms. However, G_n and H_n are $\mathcal{C}_n$ equivalent.*

1.4 Characterization of Graphs

Throughout this paper we will be considering various languages for describing colored graphs. We are interested in knowing when a language suffices to characterize a particular graph, or class of graphs. Some of the following definitions and notation are adapted from [25].

Definition 1.4.1 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 $\varphi \in \mathcal{L}$,

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

A *partial valuation* over a graph $G = (V, E)$ is a partial function $u : \{x_1 \ldots\} \to V$. The domain of u is denoted ∂u. Call a $(k\text{-})$*configuration* over G, H a pair (u, v) where u is a partial valuation over G and v is a partial valuation over H such that $\partial u = \partial v (\subseteq \{x_1, \ldots x_k\})$. If (u, v) is a k-configuration over G and H, we say that G, u and H, v are *$\mathcal{L}$-equivalent* $(G, u \equiv_{\mathcal{L}} H, v)$ iff for all formula $\varphi \in \mathcal{L}$, with free variables from $x_1, \ldots, x_k$,

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

□

Using the concept of $\mathcal{L}$-equivalence, we can now define what it means for the language $\mathcal{L}$ to characterize a set of graphs.

Definition 1.4.2 We say that $\mathcal{L}$ *k-characterizes* G iff for all graphs H, and for all k-configurations (u, v) over G, H, if G, u and H, v are $\mathcal{L}$-equivalent then there is an isomorphism from G to H extending the correspondence given by (u, v). In symbols,

$$(G, u \equiv_{\mathcal{L}} H, v) \quad \Rightarrow \quad (\exists f \supset v \circ u^{-1})(f : G \xrightarrow{\cong} H) \, .$$

We say that $\mathcal{L}$ *characterizes* G iff $\mathcal{L}$ 1-characterizes $\hat{G}$, for all colorings $\hat{G}$ of G. For a set of graphs S, we say that $\mathcal{L}$ *characterizes* S iff for all $G \in S$, $\mathcal{L}$ characterizes G. □

Proposition 1.4.3 *Let GRAPHS be the set of all finite, colored graphs, and let FO be the first-order language of colored graphs. Then FO characterizes GRAPHS.*

Proof Let $G \in$ GRAPHS have n vertices, and let u be a partial valuation over G. For simplicity, suppose that $\partial u = \{x_1, \ldots, x_k\}$, and $u(x_1), \ldots, u(x_k)$ are all distinct. Let $g_1, \ldots, g_n$ be a numbering of G's vertices so that $g_i = u(x_i)$, for $1 \le i \le k$. Let r be a subscript greater than that of any color relation holding in G. It is simple to write a first-order formula, χ_r, with $n+1-k$ quantifiers that says (a) there exist $x_{k+1} \ldots x_n$ such that the x_i's are all distinct; (b) any other vertex is equal to one of the x_i's; (c) each pair (x_i, x_j) has an edge or not exactly as the edge (g_i, g_j) occurs or not in G; and finally (d) for each x_i, $i \le n$, and each C_j, $j \le r$, $C_j(x_i)$ holds exactly if $C_j(g_i)$ holds in G. Let H be any graph, and let r be greater than the index of any color relation holding in H. Let v be any valuation over H such that H, v satisfies χ_r. Let v' be an extension of v to a valuation over H with $\partial v' = \{x_1 \ldots x_n\}$, making the quantifier-free part of χ_r true. Then clearly $f : g_i \mapsto v'(x_i)$ is the desired isomorphism. □

Proposition 1.4.3 leads to an inefficient graph canonization algorithm. In the next section, we consider languages weaker than full first-order logic, in order to obtain efficient algorithms.

1.5 The Language $\mathcal{L}_k$

Define $\mathcal{L}_k$ to be the set of first-order formulas, φ, such that the quantified variables in φ 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). For example consider the following sentence in $\mathcal{L}_2$.

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

The sentence, ψ, 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 ψ. Note that the outermost quantifier, $\forall x_1$, refers only to the free occurrence of x_1 within its scope.

In this section we will consider the question, "Which graphs are characterized by $\mathcal{L}_k$?" Define a *color class* to be the set of vertices which satisfy a particular set of color relations and no others. The *color class size* of a graph is defined to be the cardinality of the largest color class.

Proposition 1.5.1 *$\mathcal{L}_2$ characterizes the colored graphs with color class size one.*

Proof This is clear. In $\mathcal{L}_2$ we can assert that each color class is of size at most one, e.g. $\forall x_1 \forall x_2 \big(B(x_1) \wedge B(x_2) \ \rightarrow x_1 = x_2\big)$. We can also say which edges exist, e.g. the blue vertex is connected to the red vertex. Thus if graph G has color class size one, and if $G, g \equiv_{\mathcal{L}_2} H, h$ then there is an isomorphism $f : G \rightarrow H$. Since f preserves colors, $f(g) = h$. □

Next we consider the much more powerful language $\mathcal{L}_3$. In this language we can express the existence of paths.

Proposition 1.5.2 *For any natural number r, the formula $P_r(x_1, x_2)$, meaning that there is a path of length at most r from x_1 to x_2, can be written in $\mathcal{L}_3$.*

Proof By induction. $P_1(x_1, x_2)$ is $E(x_1, x_2) \vee x_1 = x_2$. Inductively,

$$P_{s+t}(x_1, x_2) \quad \equiv \quad \exists x_3 \big(P_s(x_1, x_3) \wedge P_t(x_3, x_2)\big)$$

Note that a maximum of 3 distinct variables is used. □

We will see in Section 1.6 that there are graphs with color class size 2 that cannot be distinguished by a sentence in $\mathcal{L}_2$. The ability of $\mathcal{L}_3$ to talk about path lengths makes it slightly less trivial:

Proposition 1.5.3 *$\mathcal{L}_3$ characterizes graphs of color class size at most three.*

Proof Let G and H be colored graphs, let g and h be vertices of G and H, and suppose that $G, g \equiv_{\mathcal{L}_3} H, h$. We will build an isomorphism $f : G \rightarrow H$, such that $f(g) = h$.

We first refine the colorings of the vertices of G and H to correspond to $\mathcal{L}_3$ types. For $A, B \in \{G, H\}$, vertices $a \in A$ and $b \in B$ will have the same $\mathcal{L}_k$-refined color iff they satisfy the same $\mathcal{L}_k$ formulas, i.e.

$$\big\{\varphi \in \mathcal{L}_k \;\big|\; A \models \varphi_a^{x_1}\big\} = \big\{\varphi \in \mathcal{L}_k \;\big|\; B \models \varphi_b^{x_1}\big\}^7 .$$

The following lemma says that we may assume that the color types of G and H are already refined.

Lemma 1.5.4 *Let the finite, colored graphs G and H be $\mathcal{L}_k$ equivalent and let G' and H' be the $\mathcal{L}_k$ color refinements of G and H. Then G' and H' are $\mathcal{L}_k$ equivalent.*

---

[7] The notation α_t^x denotes the formula α with the term t substituted for the variable x.

Proof Since G and H are finite, each refined color class C'_i is determined by the conjunction $\psi_i \in \mathcal{L}_k$ of a finite set of formulas. That is for all i, G' and H' both satisfy

$$\forall x_1 (C'_i(x_1) \leftrightarrow \psi_i) \ .$$

Note that ψ_i has x_1 as its free variable. Thus any occurrence of $C'_i(x_1)$ may be replaced by the equivalent ψ_i. Similarly any occurrence of $C'_i(x_j), j = 2, \ldots, k$ may be replaced by $\psi_i^{\pi_j}$ where π_j is a permutation of $\{x_1, \ldots, x_k\}$ sending x_1 to x_j. Now for any formula $\alpha \in \mathcal{L}_3(C'_1, C'_2, \ldots)$ we may replace each occurrence of $C'_i(x_j)$ by $\psi_i^{\pi_j}$ to obtain an equivalent formula $\alpha' \in \mathcal{L}_3(C_1, \ldots C_r)$. □

By the above lemma we may assume that the color classes of G and H correspond exactly to the $\mathcal{L}_3$ types of the vertices. Let R and B be two colors and consider the edges between red and blue vertices in G or H. Note that this is a regular bipartite graph because we can express in $\mathcal{L}_3$ that a red vertex has $0, 1, 2$, or all blue vertices as neighbors. Note also that for color classes of size at most 3, the only regular bipartite graphs representing nontrivial relationships between vertices are the 1:1 correspondence graphs and their complements. Let us then change such bipartite graphs as follows: replace the complete bipartite graph by its complement, and replace the graphs of degree two whose complements are 1:1 correspondence graphs by these complements. Note that when we perform these changes on G and H the new graphs are still $\mathcal{L}_3$ equivalent, and they are isomorphic now iff they were before.

Let the *color valence* of a graph be the maximum number of edges from any vertex to vertices of a fixed color. We have reduced the problem to constructing an isomorphism between $\mathcal{L}_3$-equivalent graphs G and H when these graphs have color valence one. We construct the isomorphism f as follows: Begin by letting $f(g) = h$. Next, while there is a vertex g_1 in the domain of f with a (unique) neighbor g_2 of color C_i not yet in the domain of f, do the following. Let h_2 be the neighbor of $f(g_1)$ of color C_i, and let $f(g_2) = h_2$.

We claim that the function f constructed above is an isomorphism from G to H. If not, then it must be the case that there is a loop of a certain color sequence in one of the graphs but not the other. For example, suppose that we chose $g_1, g_2, \ldots, g_j$ and $h_1, h_2, \ldots, h_j$ so that g_1 and h_1 are color C_1, and for $i < j$, g_{i+1} and h_{i+1} are the unique neighbors of g_i and h_i, respectively, of color C_{i+1}. However, suppose now that the neighbor of h_j of color C_1 is h_1, but that g_1 is not a neighbor of g_j. In this case there is a certain easily describable loop in H but not in G. That means that G and H disagree on the following $\mathcal{L}_3$ formula:

$$\begin{aligned}\Big(C_1(x_1) \quad &\wedge \quad \exists x_2 \big(C_2(x_2) \wedge E(x_1, x_2) \wedge \exists x_3 (C_3(x_3) \wedge E(x_2, x_3) \wedge \\ &\wedge \quad \exists x_2 (C_4(x_2) \wedge E(x_3, x_2) \wedge \ \ldots \ \wedge \exists x_i (C_j(x_i) \wedge E(x_i, x_1)) \ldots \big)\Big)\end{aligned}$$

Since $G \equiv_{\mathcal{L}_3} H$ they must agree on the above formula. Therefore f is an isomorphism as claimed. □

In the next section we describe some games that may be used to prove lower bounds on the expressibility of the $\mathcal{L}_k$'s. We will show as an example using these games that $\mathcal{L}_2$ does not suffice to characterize graphs of color class size 2. Recently it has been shown (cf. Fact 1.3.4) that no fixed $\mathcal{L}_k$ suffices to characterize the graphs of color class size 4.

1.6 Lower Bounds

In this section we will show that $\mathcal{L}_k$ is not expressive enough to characterize graphs efficiently. We will use the combinatorial games of Ehrenfeucht and Fraisse [10, 12] as modified for $\mathcal{L}_k$ (see [18, 7, 29]). All of the results in this section could be proved by induction on the complexity of the sentences in question; but, we find that the games offer more intuitive arguments.

Let G and H be two graphs, and let k be a natural number. Define the *$\mathcal{L}_k$ game* on G and H as follows. There are two players, and there are k pairs of pebbles, $g_1, h_1, \ldots, g_k, h_k$. On each move, Player I picks up any of the pebbles and he places it on a vertex of one of the graphs. (Say he picks up g_i. He must then place it on a vertex from G.) Player II then picks up the corresponding pebble, (If Player I chose g_i then she must choose h_i), and places it on a vertex of the appropriate graph (H in this case).

Let $p_i(r)$ be the vertex on which pebble p_i is sitting just after move r. Then we say *Player I wins the game at move r* if the map that takes $g_i(r)$ to $h_i(r)$, $i = 1, \ldots, k$, is not an isomorphism of the induced k vertex subgraphs. Note that if the graphs are colored then an isomorphism must preserve color as well as edges. Thus Player II has a winning strategy for the $\mathcal{L}_k$ game just if she can always find matching points to preserve the isomorphism. Player I is trying to point out a difference between the two graphs and Player II is trying to keep them looking the same.

As an example 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 g_1 on a red vertex in G. Player II may answer by putting h_1 on either of the red vertices in H. Now suppose Player I puts h_2 on an adjacent yellow vertex in H. Player II has a response because in G, $g_1(1)$ also 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. The relevant theorem concerning the relationship between this game and the matter at hand is:

Fact 1.6.1 *[18, Theorem C.1] Let (u, v) be a k-configuration over G, H. Player II has a winning strategy for the $\mathcal{L}_k$ game on (u, v) if and only if $G, u \equiv_{\mathcal{L}_k} H, v$.*

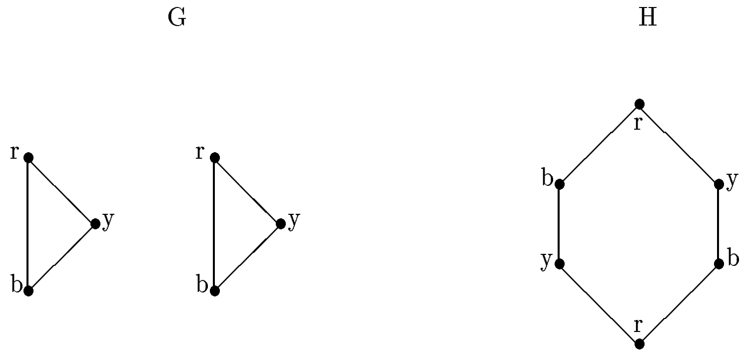

FIGURE 1.2. The $\mathcal{L}_2$ Game

Note that we have the following

Corollary 1.6.2 *$\mathcal{L}_2$ does not characterize graphs of color class size 2.*

We will prove in Section 1.9 that testing whether $G \equiv_{\mathcal{L}_k} H$ can be done in time $O[n^k \log n]$. Furthermore, if $\mathcal{L}_k$ characterizes a set S of graphs, then canonical forms for the graphs in S may be computed in this same time bound.

It is interesting to note that not only does no $\mathcal{L}_k$ characterize all graphs, but almost all graphs are indistinguishable in $\mathcal{L}_k$. Thus if two graphs of size $n \gg k$ are chosen at random they will almost certainly be $\mathcal{L}_k$ equivalent, but not isomorphic.

Fact 1.6.3 *[18],cf [11] Fix k and let $Pr_n(G \equiv_{\mathcal{L}_k} H)$ be the probability that two randomly chosen graphs of size n are $\mathcal{L}_k$ equivalent. Then*

$$\lim_{n\to\infty}\Big[Pr_n(G \equiv_{\mathcal{L}_k} H)\Big] = 1$$

Not only does $\mathcal{L}_k$ not characterize most graphs, it is not strong enough to express counting:

Proposition 1.6.4 *Let EVEN be the set of graphs with an even number of vertices. This property is not expressible in $\mathcal{L}_n$ for graphs with n or more vertices. Furthermore, $\mathcal{L}_n$ does not characterize the set of totally disconnected graph on n vertices.*

Proof Let D_n be the uncolored graph with n vertices and no edges. We claim that $D_n \equiv_{\mathcal{L}_n} D_{n+1}$. The following is a winning strategy for Player II in the n-pebble game on D_n and D_{n+1}. Player I's moves are answered preserving distinctness. That is, if Player I places pebble i on a vertex already occupied by pebble j, then Player II does the same. If Player I places pebble i on a vertex not occupied by any other pebbles, then Player II does the same. This is possible, because there are n vertices, and only

$n-1$ other pebbles. Since there are no edges, the resulting maps are always isomorphisms. □

In the next section we increase the expressive power of the $\mathcal{L}_k$'s by adding the ability to count.

1.7 Counting Quantifiers

In this section we add *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 φ. We will sometimes also use the quantifiers, $(\exists ! i\, x)$, meaning that there exists exactly i x's:

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

Example 1.7.1 As our first example, note that the following sentence in $\mathcal{C}_2$ characterizes the graph D_n of Proposition 1.6.4:

$$(\exists ! n\, x)(x = x) \;\wedge\; (\forall x)(\forall y)(\neg E(x, y))\,.$$

□

Note that every sentence in $\mathcal{C}_k$ is equivalent to an ordinary first-order sentence with perhaps many more variables and quantifiers. We will see that testing $\mathcal{C}_k$ equivalence is no harder than testing $\mathcal{L}_k$ equivalence – the idea is that to test the truth of $\forall x$ or $\exists x$ we have to consider all possible x's anyway, and it doesn't cost more to count them. In Corollary 1.9.7 we show that $\mathcal{C}_k$ equivalence can be tested in time $O[n^k \log n]$. Similarly, graphs characterized by $\mathcal{C}_k$ can be given canonical labelings in the same time.

The following notation is useful.

Definition 1.7.2 Let Σ be a set of finite graphs. Define $\mathrm{var}(\Sigma, n)$ (resp. $\mathrm{vc}(\Sigma, n)$) to be the minimum k such that $\mathcal{L}_k$ (resp. $\mathcal{C}_k$) characterizes the graphs in Σ with at most n vertices. Let $\mathrm{var}(n) = \mathrm{var}(GRAPHS, n)$ and $\mathrm{vc}(n) = \mathrm{vc}(GRAPHS, n)$. When $\mathrm{var}(\Sigma, n)$ or $\mathrm{vc}(\Sigma, n)$ is bounded, we write $\mathrm{var}(\Sigma) = \max_n \mathrm{var}(\Sigma, n)$, and $\mathrm{vc}(\Sigma) = \max_n \mathrm{vc}(\Sigma, n)$. □

For example, by combining various results obtained so far we know that $var(\mathrm{GRAPHS}, n) = n+1$, $var(CC1) = 2$, and $var(CC2) = var(CC3) = 3$. Here we are letting CCk be the set of color class k graphs.

We will now examine $\mathcal{C}_k$, attempting to compute $\mathrm{vc}(S)$ for various sets of graphs, S. 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 like the $\mathcal{L}_k$ game we have two players and k pairs of pebbles. Now however each move has two steps.

1. Player I picks up a pebble (say g_i). He then chooses a set, A, of vertices from one of the graphs (in this case G). Now Player II must answer with a set, B, of vertices from the other graph. B must have the same cardinality as A.

2. Player I places h_i on some vertex $b \in B$. Player II answers by placing g_i on some $a \in A$.

The definition for winning is as before. Note that what is going on in the two step move is that Player I is asserting that there exist $|A|$ vertices in G with a certain property. Player II answers that there are the same number of such vertices in H. A straight forward extension of the proof of Fact 3.1 shows that this game does indeed capture expressibility in $\mathcal{C}_k$.

Theorem 1.7.3 *Let (u, v) be a k-configuration over G, H. Player II has a winning strategy for the $\mathcal{C}_k$ game on (u, v) if and only if $G, u \equiv_{\mathcal{C}_k} H, v$.*

Consider the following example of the $\mathcal{C}_k$ game.

Proposition 1.7.4 *Player II has a win for the $\mathcal{C}_2$ game on the graphs pictured in Figure 2. Thus $vc(CC2) > 2$.*

Proof Player II's winning strategy is as follows: She matches the first vertex chosen by Player I with any vertex of the same color. Now suppose that at any point in the game, the first pair of pebbles are placed on vertices g_1 and h_1, both vertices of the same color, say red. Suppose that Player I's next move involves the other pair of pebbles. There is a 1:1 correspondence between the vertices in G and H as follows:

g_1	$\mapsto h_1$
blue vertex adjacent to g_1	$\mapsto$ blue vertex adjacent to h_1
yellow vertex adjacent to g_1	$\mapsto$ yellow vertex adjacent to h_1
red vertex not adjacent to g_1	$\mapsto$ red vertex not adjacent to h_1
yellow vertex not adjacent to g_1	$\mapsto$ yellow vertex not adjacent to h_1
blue vertex not adjacent to g_1	$\mapsto$ blue vertex not adjacent to h_1

If Player I chooses a set A, then Player II chooses the set B to be the corresponding set of vertices under the above map. Whichever vertex Player I then picks from B, Player II will choose the corresponding vertex in A. Thus the chosen pair of vertices will be the same color and either both adjacent, or both not adjacent to the other chosen pair. Thus Player II can always preserve the partial isomorphism. $\square$

1.8 Vertex Refinement Corresponds to $\mathcal{C}_2$

It turns out that the expressive power of $\mathcal{C}_2$ is characterized by the well known method of vertex refinement (see [2, 16]). Let $G = \langle V, E, C_1, \ldots, C_r \rangle$

be a colored graph in which every vertex statisfies exactly one color relation. Let $f : V \to \{1 \ldots n\}$ be given by $f(v) = i$ iff $v \in C_i$. We then define f', the refinement of f as follows: The new color of each vertex, v, is defined to be the following tuple:

$$\langle f(v), n_1, \ldots, n_r \rangle$$

where n_i is the number of vertices of color i that v is adjacent to. We sort these new colors lexicographically and assign $f'(v)$ to be the number of the new color class which v inhabits. Thus two vertices are in the same new color class just 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 $f^{(k)} = f^{(k+1)}$. We let $\bar{f} = f^{(k)}$ and call $\bar{f}$ the *stable refinement* of f.

The equivalence of stable colorings and $\mathcal{C}_2$ equivalence is summed up by the following

Theorem 1.8.1 *Given a colored graph, $G = \langle V, E, C_1, \ldots, C_r \rangle$, with two vertices, g_1 and g_2, the following are equivalent:*

1. $\bar{f}(g_1) = \bar{f}(g_2)$
2. *For all $\varphi(x_1) \in \mathcal{C}_2$, $G \models \varphi(g_1)$ iff $G \models \varphi(g_2)$.*
3. *Player II wins the $\mathcal{C}_2$ game on two copies of G, with pebble pair number 1 initially placed on g_1 and g_2 respectively.*

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

1. $f^{(r)}(g_1) = f^{(r)}(g_2)$
2. For all $\varphi(x_1) \in \mathcal{C}_2$ of quantifier depth r, $G \models \varphi(g_1)$ iff $G \models \varphi(g_2)$.
3. Player II wins the r move $\mathcal{C}_2$ game on two copies of G, with pebble pair number 1 initially placed on g_1 and g_2 respectively.

The base case is by definition. $f^{(0)}(g_1) = f(g_1) = f(g_2)$ iff g_1 and g_2 satisfy the same initial color predicate. This is true if and only if g_1 and g_2 satisfy all the same quantifier free formulas. This in turn is true if and only if the map sending g_1 to g_2 is a partial isomorphism. This last is the definition of Player II winning the 0 move game.

Assume that the equivalence holds for all g_1 and g_2 and for all $r < m$.

$(\neg 1 \Rightarrow \neg 2)$: Suppose that $f^{(m)}(g_1) \neq f^{(m)}(g_2)$. There are two cases. If $f^{(m-1)}(g_1) \neq f^{(m-1)}(g_2)$ then by the inductive assumption there is a quantifier depth $m-1$ formula $\varphi \in \mathcal{C}_2$ on which g_1 and g_2 differ. Otherwise it must be that g_1 and g_2 have a different number of neighbors of some $f^{(m-1)}$ color class i. Let N be the maximum of these two numbers. By induction two vertices are in the same $f^{(m-1)}$ color class iff they agree on all quantifier depth $m-1$ $\mathcal{C}_2$ formulas. Since quantifier depth $m-1$ formulas are closed under conjunction and the graphs in question are finite there is a depth $m-1$ $\psi_i \in \mathcal{C}_2$ such that for all $g \in G$,

$$f^{(m-1)}(g) = i \quad \Leftrightarrow \quad G \models (\psi_i)^{x_1}_g$$

It follows that g_1 and g_2 differ on the formula:

$$(\exists N x_2)(E(x_1, x_2) \wedge {\psi_i}^{x_1}_{x_2}) \ .$$

$(\neg 2 \Rightarrow \neg 3)$: Suppose that $G \models \varphi^{x_1}_{g_1}$ but $G \models \neg\varphi^{x_1}_{g_1}$, for some $\varphi \in \mathcal{C}_2$ of quantifier depth m. If φ is a conjunction then g_1 and g_2 must differ on at least one of the conjuncts, so we may assume that φ is of the form $(\exists N x_2)\psi(x_2)$. On the first move of the game Player I chooses the N vertices v such that $\psi(v)^{x_1}_{g_1}$. Whatever Player II chooses as B there will be at least one vertex v_2 such that $G \models \neg\psi(v_2)^{x_1}_{g_1}$. Player I puts his pebble number 2 on this v_2. Player II must respond with some $v_1 \in A$. The vertices v_1, v_2 now differ on a quantifier depth $m-1$ formula. Thus by induction Player II loses the remaining $m-1$ move game.

$(1 \Rightarrow 3)$: Suppose that $f^{(m)}(g_1) = f^{(m)}(g_2)$. It follows that g_1 and g_2 have the same number of neighbors of each $f^{(m-1)}$ color. Thus a 1:1 correspondence exists between the vertices in the first copy of G and those in the second preserving both the property of being adjacent to g_i and the $f^{(m-1)}$ color. (Note that since we are considering two copies of the same graph, if both copies have the same number of red neighbors of g_i then they also both have the same number of red non-neighbors of g_i.) It follows that Player II can assure that after the first move the pair of vertices chosen will be in the same $f^{(m-1)}$ color class. Thus by the induction hypothesis Player II has a win for the remaining $m-1$ move game. □

All Trees and Almost All Graphs

Theorem 1.8.1 combined with some facts about stable colorings provide us with several corollaries concerning graphs characterized by $\mathcal{C}_2$. First, it is well known that the set of finite trees is characterized by stable coloring [1]. Thus:

Corollary 1.8.2 *Let TREES be the set of finite trees. Then vc(TREES) = 2.*

It is interesting to compare Corollary 1.8.2 with the more complicated situation in which counting is not present:

Fact 1.8.3 *[25] Let T_k be the set of finite trees such that each node has at most k children, and let S_k be the subset of T_k in which each non-leaf has exactly k children. Then,* *1.*

$$var(T_k) = \begin{cases} 2 & \textit{if } k = 1 \\ 3 & \textit{if } 2 \leq k \leq 3 \\ k & \textit{if } k > 3 \end{cases}$$

2.

$$var(S_k) = \begin{cases} 2 & \textit{if } 1 \leq k \leq 2 \\ 3 & \textit{if } 3 \leq k \leq 6 \\ \lceil k/2 \rceil & \textit{if } k > 6 \end{cases}$$

Babai and Kučera have proved the following result about stable colorings of random graphs:

Fact 1.8.4 *[4] There exists a constant $\alpha < 1$ such that if G is chosen randomly from the set of all labeled graphs on n vertices then*

$$Prob\{G \textit{ has two vertices of the same stable color}\} < \alpha^n \, .$$

Corollary 1.8.5 *Almost all finite graphs are characterized by $\mathcal{C}_2$.*

It is easy to see that Fact 1.8.4 fails for regular graphs: all regular graphs of degree d on n vertices are $\mathcal{C}_2$ equivalent. More recently, Kučera has given a linear algorithm for canonization of regular graphs of a given, fixed degree [26]. It follows from his results that:

Corollary 1.8.6 *For all d, and sufficiently large n, $\mathcal{C}_3$ characterizes more than $1 - O[1/n]$ of the regular graphs of degree d on n vertices.*

1.9 Equivalence and Canonization Algorithms

The stable coloring of a graph is computable in $O[|E| \log n]$ steps [16]. We present the algorithm for completeness.

Algorithm 1.9.1 *1. Place indices $1, \ldots, r$ of initial color classes on list L.*
2. **While** $L \neq \emptyset$ **do begin**
3. **For** *each vertex v adjacent to some color classes in L,*
record how many neighbors of each such color class v has.
4. *Sort these records to form new color classes.*
5. *Replace L with indices of all but the largest piece of each old class.*

Theorem 1.9.2 *Algorithm 1.9.1 computes the vertex refinement of a graph G. It can be implemented to run in $O[|E| \log n]$ time on a RAM.*

Proof If we implement line 4 as a bucket sort then the amount of work in performing an iteration of the while loop is proportional to the number of edges traversed. Note that each time an edge is traversed, the color class of it's head is at most half of its previous size. Thus $O[|E|\log n]$ steps suffice. □

Corollary 1.9.3 *We can test if $G \equiv_{\mathcal{C}_2} H$ in $O[|E|\log n]$ steps, where $|E|$ is the number of edges in G.*

Proof We compute the stable coloring of $G \cup H$. G and H are $\mathcal{C}_2$ equivalent iff each color class has the same number of vertices from each graph. □

As promised, we show how to modify the above algorithm to compute canonical labelings of graphs characterized by $\mathcal{C}_2$.

Theorem 1.9.4 *Let S be a set of finite graphs characterized by $\mathcal{C}_2$. Then canonical labelings for S are computable in $O[|E|\log n]$ steps.*

Proof We modify Algorithm 1.9.1 as follows: When a stable coloring is reached, if each vertex has a unique stable color, then a canonical labeling is determined. Otherwise, let C_i be the first color class of size greater than one, and let g be a vertex of color C_i. Make g a new color, C_{new}, add C_{new} to L and continue the refinement.

Suppose that $G, g \equiv_{\mathcal{C}_2} H, h$. Let G' and H' be the result of coloring g and h 'new'. Since $\mathcal{C}_2$ characterizes S, G' and H' are isomorphic. It follows that $\mathcal{C}_2$ equivalent graphs will result in the same canonical labeling. Furthermore, the analysis of the revised algorithm is unchanged. □

We will next present the algorithm to test $\mathcal{C}_{k+1}$ equivalence for $k \geq 2$. Define *stable colorings of k tuples* as follows: Initially we give each k tuple of vertices from G a color according to its isomorphism type. That is $\langle g_1 \ldots g_k \rangle$ has the same initial color as $\langle h_1 \ldots h_k \rangle$, just if the map $\alpha : g_i \mapsto h_i, \quad i = 1 \ldots k$ is an isomorphism.

We next form the new color of $\langle g_1 \ldots g_k \rangle$ as the tuple:

$$\left\langle f(g_1 \ldots g_k), \mathrm{SORT}\left\{ \langle f(g, g_2 \ldots, g_k), f(g_1, g, g_3, \ldots, g_k), \ldots, f(g_1, \ldots, g_{k-1}, g) \rangle \;\middle|\; g \in G \right\} \right\rangle$$

That is the new color of a k-tuple is formed from the old color, as well as from considering, for each vertex g, the old color of the k k-tuples resulting from the substitution of g into each possible place.

Theorem 1.9.5 *A stable coloring of k tuples in an n vertex graph may be computed in $O[k^2 n^{k+1} \log n]$ steps.*

Proof This is a generalization of Algorithm 1.9.1. We must refine the coloring for each color class, B_i, of k-tuples. Each such refinement takes $O[kn]$ steps for each k-tuple in B_i. Each of the n^k k-tuples will have its color class treated at most $\log(n^k)$ times. □

Theorem 1.9.6 *Let G be a graph whose $k-1$ tuples of vertices are colored. Let $\vec{g}, \vec{h} \in G^{k-1}$. The following are equivalent.*

1. $\bar{f}(\vec{g}) = \bar{f}(\vec{h})$
2. *For all $\varphi(x_1 \ldots x_{k-1}) \in \mathcal{C}_k$, $G \models \varphi(\vec{g})$ iff $G \models \varphi(\vec{h})$*
3. *Player II wins the $\mathcal{C}_k$ game on two copies of G with pebbles $1 \ldots k-1$ initially placed on $g_1 \ldots g_{k-1}$ and $h_1 \ldots h_{k-1}$ respectively.*

Proof The proof is similar to that of Theorem 1.8.1. □

Corollary 1.9.7 *$\mathcal{C}_k$ equivalence may be tested in $O[n^k \log n]$ steps. (If k is allowed to vary with n this becomes $O[k^2 n^k \log n]$.) Similarly, if S is characterized by $\mathcal{C}_k$, then canonical labelings for S may be computed in the same time bound.*

1.10 Conclusions

We have begun a study of which sets of graphs are characterized by the languages $\mathcal{L}_k$ and $\mathcal{C}_k$. For such sets of graphs we have given simple and efficient canonization algorithms. General directions for further study include the following:

1. There are many interesting questions concerning the values $var(S)$ and $vc(S)$ for various classes of graphs S. In particular it would be very interesting to determine vc(Planar Graphs) and vc(Genus k) graphs.
2. Question 1.3.3 in its new form, "What must we add to first-order logic with fixed point and counting in order to obtain all polynomial-time graph properties" deserves considerable further study, cf. [8].
3. Fact 1.3.4 implies that (FO + LFP + counting) does not even include all of DSPACE[$\log n$]. It would be very interesting, and perhaps more tractable to answer question 2 for other classes such as NSPACE[$\log n$].

Acknowledgements: Thanks to Steven Lindell for suggesting some improvements to this paper.

1.11 References

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

[2] Laszlo Babai, "Moderately Exponential Bound for Graph Isomorphism," Proc. Conf. on Fundamentals of Computation Theory, Szeged, August 1981.

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

[4] Laszlo Babai and Luděk Kučera (1980), Canonical Labelling of Graphs in Linear Average Time," *20th IEEE Symp. on Foundations of Computer Science,* 39-46.

[5] Laszlo Babai and Eugene M. Luks, "Canonical Labeling of Graphs," *15th ACM STOC Symp.*, (1983), 171-183.

[6] D. Mix Barrington, N. Immerman, and H. Straubing, "On Uniformity Within NC^1," *Third Annual Structure in Complexity Theory Symp.* (1988), 47-59.

[7] Jon Barwise, "On Moschovakis Closure Ordinals," J. Symb. Logic 42 (1977), 292-296.

[8] J. Cai, M. Fürer, N. Immerman, "An Optimal Lower Bound on the Number of Variables for Graph Identification," *30th IEEE FOCS Symp.* (1989), 612-617.

[9] Ashok Chandra and David Harel, "Structure and Complexity of Relational Queries," *JCSS* **25** (1982), 99-128.

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

[11] Ron Fagin, "Probabilities on Finite Models," *J. Symbolic Logic* **41**, No. 1 (1976), 50-58.

[12] R. Fraissé, "Sur les Classifications des Systems de Relations," *Publ. Sci. Univ. Alger* **I** (1954).

[13] Leslie Goldschlager, "The Monotone and Planar Circuit Value Problems are Log Space Complete for P," *SIGACT News* **9**, No. 2 (1977).

[14] Yuri Gurevich, "Logic and the Challenge of Computer Science," in *Current Trends in Theoretical Computer Science,* ed. Egon Börger, Computer Science Press.

[15] Christoph M. Hoffmann, *Group-Theoretic Algorithms and Graph Isomorphism,* Springer-Verlag Lecture Notes in Computer Science 136 (1982).

[16] John E. Hopcroft and Robert Tarjan, "Isomorphism of Planar Graphs," in *Complexity of Computer Computations,* R. Miller and J.W Thatcher, eds., (1972), Plenum Press, 131-152.

[17] Neil Immerman, "Number of Quantifiers is Better than Number of Tape Cells," *JCSS* **22**, No. 3, June 1981, 65-72.

[18] Neil Immerman, "Upper and Lower Bounds for First Order Expressibility," *JCSS* **25**, No. 1 (1982), 76-98.

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

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

[21] Neil Immerman, "Nondeterministic Space is Closed Under Complementation," *SIAM J. Comput.* **17**, No. 5 (1988), 935-938.

[22] Neil Immerman, "Expressibility and Parallel Complexity," *SIAM J. of Comput* **18** (1989), 625-638.

[23] Neil Immerman, "Expressibility as a Complexity Measure: Results and Directions," *Second Structure in Complexity Theory Conf.* (1987), 194-202.

[24] Neil Immerman, "Descriptive and Computational Complexity," in *Computational Complexity Theory,* ed. J. Hartmanis, *Proc. Symp. in Applied Math.,* 38, American Mathematical Society (1989), 75-91.

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

[26] Luděk Kučera, "Canonical Labeling of Regular Graphs in Linear Average Time," *28th IEEE FOCS Symp.* (1987), 271-279.

[27] Eugene M. Luks, "Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time," JCSS 25 (1982), pp. 42-65.

[28] Yiannis N. Moschovakis, *Elementary Induction on Abstract Structures,* North Holland, 1974.

[29] Bruno Poizat, "Deux ou trois chose que je sais de Ln," *J. Symbolic Logic,* 47 (1982), 641-658.

[30] Simon Thomas, "Theories With Finitely Many Models," *J. Symbolic Logic,* **51**, No. 2 (1986), 374-376.

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