A comparison of compatible, finite, and inductive graph properties

Annegret Habel and Hans-Jörg Kreowski
Department of Computer Science, University of Bremen, W-2800 Bremen, Germany

Clemens Lautemann
Institute of Computer Science, University of Mainz, W-6500 Mainz, Germany

Communicated by A. Salomaa
Received September 1990
Revised July 1991

Abstract

Habel, A., H.-J. Kreowski and C. Lautemann, A comparison of compatible, finite, and inductive graph properties, Theoretical Computer Science 110 (1993) 145–168.

In the theory of hyperedge-replacement grammars and languages, one encounters three types of graph properties that play an important role in proving decidability and structural results. The three types are called compatible, finite, and inductive graph properties. All three of them cover graph properties that are well-behaved with respect to certain operations on hypergraphs. In this paper, we show that the three notions are essentially equivalent. Consequently, three lines of investigation in the theory of hyperedge replacement – so far separated – merge into one.

1. Introduction

Hyperedge-replacement grammars and their generated languages, originally introduced by Feder [6] as context-free plex grammars and languages, have been studied intensively in the last years in several variants and under different names. See, e.g., the algebraic study of context-free grammars by Bauderon and Courcelle [2], cellular graph grammars introduced and investigated by Lengauer and Wanke [12], and [8]. An interesting portion of the theory of hyperedge replacement is based on graph properties that behave "nicely" under certain operations on graphs and hypergraphs.

In [12], Lengauer and Wanke introduce the notion of a *finite* property Π. It means roughly that there is only a finite number of classes of graphs that behave differently with respect to Π when put into all possible environments. Courcelle investigates in

Correspondence to: A. Habel, Department of Computer Science, University of Bremen, W-2800 Bremen 33, Germany. Email: habel@informatik.uni-bremen.de.

[3, 4] the so-called *inductive* properties. Roughly speaking, a family of predicates is inductive relative to a set of graph operations if each predicate can be tested in the following way. If a graph is given as a result of applying one of the operations, then whether the predicate holds for it can be determined by checking the corresponding predicates for the arguments of the operation. Finally, in [10], graph properties are studied that are *compatible* with the derivation process of a class of hyperedge-replacement grammars in a certain way. In all three cases, more or less the same results have been shown. For example, given a compatible, or finite, or inductive property Π, the following questions are decidable for hyperedge-replacement grammars:

(1) Does the language generated by the input grammar have a member which satisfies Π?

(2) Do infinitely many members of the generated language satisfy Π?

(3) Do all members of the generated language satisfy Π?

Other examples are the linear-time theorem and the filter theorem: The linear-time theorem says that, given a compatible, or finite, or inductive property Π, for hyperedge-replacement grammars and generated hypergraphs represented by a derivation (or something equivalent), the question

(4) Does the hypergraph satisfy Π?

can be decided in linear time. The filter theorem states that the intersection of a generated language with the set of graphs satisfying the given property can be generated by a hyperedge-replacement grammar. Moreover, the known examples of compatible, finite, and inductive properties are the same: k-colorability, connectedness, planarity, Hamiltonicity, etc. No wonder, because in this paper all three notions are shown to be essentially equivalent for the class of all hyperedge-replacement grammars.

In more detail, the paper is organized in the following way. A common framework of hyperedge replacement is presented in Section 2. As objects of interest we have chosen unlabeled, undirected graphs, because they are well-known from graph theory and many areas in computer science. Our results can, however, be extended to richer structures, such as, e.g., directed node- and hyperedge-labeled hypergraphs. Graphs are decorated with hyperedges, so that they can be used to generate sets of graphs. In Section 3, the notion of a compatible graph property is recalled. Compatible properties are compared with finite graph properties in Section 4, and with inductive graph properties in Section 5. In both cases, we can show, essentially, equivalence. The equivalence proofs are based on the observation that the operations related to finiteness as well as the operations related to inductivity can be simulated by certain hyperedge-replacement grammars. The paper ends with a discussion.

The model of hyperedge replacement we have chosen in this paper is the one studied by Lengauer and Wanke. The models used in [2, 8, 10] are more general. In other words, we consider the intersection of the three models to get a basis for the comparisons. This also allows us to recall results from all three models as facts (sometimes just specialized to the chosen model).

2. Hyperedge replacement

In this section, we recall the basic notions and results on hyperedge replacement in order to establish a basis for the comparison of compatibility, finiteness, and inductivity. We choose ordinary unlabeled undirected graphs without loops and multiple edges as terminal structures of interest. To generate sets of graphs, they are decorated by hyperedges in intermediate steps. Each hyperedge has a label and an ordered finite set of tentacles each of which is attached to a node and is a placeholder for a graph or – recursively – for another decorated graph. Each decorated graph has a sequence of external nodes. If a hyperedge is replaced by a decorated graph, the external nodes of the latter are fused with the nodes adjacent to the hyperedge.

Definition 2.1 (*Graph*). A *graph* is a pair $G=(V,E)$, where V is a set of *nodes* (or *vertices*) and E is a set of 2-element subsets of V, called *edges*. The class of all graphs is denoted by $\mathcal{G}_0$.

Definition 2.2 (*Decorated graph*). Let N be a set of *nonterminal labels*. A (hyperedge-) *decorated graph* (over N) is a system $H=(V, E, Y, \text{lab}, \text{att}, \text{ext})$, where $(V,E)\in\mathcal{G}_0$, Y is a set of *hyperedges*, $\text{lab}\colon Y\to N$ is a mapping, called the *labeling*, $\text{att}\colon Y\to V^*$ is a mapping, called the *attachment*, and $\text{ext}\in V^*$, called the (sequence of) *external nodes*. The class of all decorated graphs over N is denoted by $\mathcal{G}(N)$.

Remarks. (1) The components V, E, Y, lab, att, and ext of H are also denoted by $V_H, E_H, Y_H, \text{lab}_H, \text{att}_H$, and ext_H, respectively.

(2) The length of ext_H is called the *type* of H and denoted by $\text{type}(H)$. If $\text{type}(H)=n$, H is called an *n-graph*. The class of all n-graphs over N is denoted by $\mathcal{G}_n(N)$.

(3) Accordingly, the length of $\text{att}_H(y)$ for $y\in Y_H$ is called the *type* of y, denoted by $\text{type}(y)$. If $\text{type}(y)=n$, y is called an n-edge. We use the word "type", because n-edges will play the role of placeholders for n-graphs.

(4) For an n-graph H, the 0-graph $U(H)=(V_H, E_H, Y_H, \text{lab}_H, \text{att}_H, \lambda)$ is said to be the 0-graph *underlying H*.

(5) A graph may be seen as a 0-graph without hyperedges. In this sense, $\mathcal{G}_0\subseteq\mathcal{G}_0(N)\subseteq\mathcal{G}(N)$. Moreover, the subclass of $\mathcal{G}_n(N)$, for $n\geqslant 0$, consisting of all n-graphs without hyperedges is denoted by $\mathcal{G}_n$, while $\mathcal{G}$ denotes the union of all $\mathcal{G}_n$, for $n\geqslant 0$. In the description of n-graphs without hyperedges we will drop the components Y, lab, and att.

(6) Another particular case is a decorated graph with a single hyperedge where external nodes and attachment coincide, i.e., $H=(V,\emptyset,\{y\},\text{lab},\text{att},\text{ext})$ with $\text{ext}=\text{att}(y)$. Such an H is called a *handle*.[1] If $V=[n]$,[2] $\text{lab}(y)=A$, and $\text{att}(y)=\text{ext}=1\ldots n$, H is denoted by $(A,n)^\bullet$.

[1] Note that our handles are different from classical handles in parsing.

[2] For $n\geqslant 1$, $[n]$ denotes the set $\{1,2,\ldots,n\}$.

Definition 2.3 (*Isomorphic decorated graphs*). Let $H, H' \in \mathcal{G}(N)$. Then H and H' are *isomorphic*, denoted by $H \cong H'$, if there are bijections $f\colon V_H \to V_{H'}$ and $g\colon Y_H \to Y_{H'}$ such that $E_{H'} = \{\{f(x), f(y)\} \mid \{x, y\} \in E_H\}$, $\text{lab}_H(y) = \text{lab}_{H'}(g(y))$ and $f^*(\text{att}_H(y)) = \text{att}_{H'}(g(y))$[3] for all $y \in Y_H$, and $f^*(\text{ext}_H) = \text{ext}_{H'}$.

Hyperedge replacement is composed of three simple constructions on decorated graphs: Hyperedge removal, disjoint union, and node fusion.

Definition 2.4 (*Operations on decorated graphs*). (1) Let $H \in \mathcal{G}(N)$ and $B \subseteq Y_H$. Then the *removal* of B from H yields the decorated graph

$$H - B = (V_H, E_H, Y_H - B, \text{lab}, \text{att}, \text{ext}_H),$$

with $\text{lab}(y) = \text{lab}_H(y)$ and $\text{att}(y) = \text{att}_H(y)$ for all $y \in Y_H - B$.[4]

(2) Let $H, H' \in \mathcal{G}(N)$. Then the *disjoint union* of H and H' is the decorated graph

$$H + H' = (V_H + V_{H'}, E_H + E_{H'}, Y_H + Y_{H'}, \text{lab}, \text{att}, \text{ext}_H),$$

with $\text{lab}(y) = \text{lab}_H(y)$ and $\text{att}(y) = \text{att}_H(y)$ for all $y \in Y_H$ and $\text{lab}(y) = \text{lab}_{H'}(y)$ and $\text{att}(y) = \text{att}_{H'}(y)$ for all $y \in Y_{H'}$.[4]

(3) Let $H \in \mathcal{G}(N)$, δ an equivalence relation on V_H, and V the corresponding quotient set (of V_H through δ), where the equivalence class of each $v \in V_H$ is denoted by $[v]$. Then the *node fusion* in H according to δ yields the decorated graph

$$H/\delta = (V, E, Y_H, \text{lab}_H, \text{att}, \text{ext}),$$

with

- $E = \{\{[v], [v']\} \mid \{v, v'\} \in E_H, [v] \neq [v']\}$,
- $\text{att}(y) = [v_1]\dots[v_m]$ if $\text{att}_H(y) = v_1 \dots v_m$ $(y \in Y_H)$, and
- $\text{ext} = [v_1]\dots[v_n]$ if $\text{ext}_H = v_1 \dots v_n$.

If $u = x_1 \dots x_n$, $v = y_1 \dots y_n \in V_H^*$ and if δ is the equivalence relation on V_H generated by $x_i = y_i$, for $i = 1, \dots, n$, we also write $H/(u = v)$ instead of H/δ, in order to emphasize the generating relation.

Remarks. (1) Removal removes some hyperedges without changing anything else. In particular, we have $\text{type}(H - B) = \text{type}(H)$.

(2) The disjoint union is asymmetric with respect to the choice of the external nodes which are taken from the first component only. Hence, $\text{type}(H + H') = \text{type}(H)$.

(3) Node fusion may identify different nodes of a graph, according to an equivalence relation. The external nodes as well as the incidences of edges and hyperedges are adapted accordingly. Labels and types remain unchanged. In particular, $\text{type}(H/\delta) = \text{type}(H)$.

[3] For a mapping $f\colon A \to B$, the free symbolwise extension $f^*\colon A^* \to B^*$ is defined by $f^*(a_1 \dots a_k) = f(a_1) \dots f(a_k)$ for all $k \in \mathbb{N}$ and $a_1, \dots, a_k \in A$.

[4] For sets A and B, $A - B$ denotes the set-theoretic difference of A and B, i.e., $A - B = \{a \in A \mid a \notin B\}$; $A + B$ denotes the disjoint union of A and B.

Definition 2.5 (*Hyperedge replacement*). Let $H\in\mathcal{G}(N)$ and $B=\{y_1,\dots,y_n\}\subseteq Y_H$. Let $\text{repl}:B\to\mathcal{G}(N)$ be a *well-typed* mapping i.e., a mapping with $\text{type}(y_i)=\text{type}(\text{repl}(y_i))$ for $i=1,\dots,n$. Then the *replacement* of B in H through repl yields the decorated graph

$$\text{REPLACE}(H,\text{repl})=\left((H-B)+\sum_{i=1}^{n}\text{repl}(y_i)\right)\Big/(\text{att}=\text{ext})$$

with $\text{att}=\text{att}_H(y_1)\dots\text{att}_H(y_n)$ and $\text{ext}=\text{ext}_{\text{repl}(y_1)}\dots\text{ext}_{\text{repl}(y_n)}$.

Remarks. (1) Hyperedge replacement is a simple construction where some hyperedges are removed, the associated decorated graphs are added disjointly and their external nodes are fused with the corresponding nodes formerly attached to the replaced hyperedges.

(2) Note that the component graphs replacing hyperedges are fully embedded into the resulting graph where their external nodes lose this status and may additionally be fused with other nodes.

(3) If $H\in\mathcal{G}(N)$, $B=\{y_1,\dots,y_n\}\subseteq Y_H$, and $\text{repl}(y_i)=R_i$ for $i=1,\dots,n$, then we sometimes write $H[y_1/R_1,\dots,y_n/R_n]$ instead of $\text{REPLACE}(H,\text{repl})$.

A hyperedge replacement defines (up to isomorphism) a direct derivation step in a grammar if the label of each replaced hyperedge and the replacing decorated graph form a production of the grammar. The graphs generated by such a grammar are obtained by starting from the handle $(S,0)^\bullet$, where S is the start symbol of the grammar, and iterating derivation steps until all hyperedges are replaced.

Definition 2.6 (*Hyperedge-replacement grammar and language*). (1) A *production* (over N) is a pair $p=(A,R)$ with $A\in N$ and $R\in\mathcal{G}(N)$. A is called the *left-hand side* of p and denoted by $\text{lhs}(p)$. R is called the *right-hand side* and denoted by $\text{rhs}(p)$.

(2) Let $H\in\mathcal{G}(N)$ and $B\subseteq Y_H$. Let P be a set of productions. A mapping $b:B\to P$ is called a *base* in H if $\text{lab}_H(y)=\text{lhs}(b(y))$ and $\text{type}(y)=\text{type}(\text{rhs}(b(y)))$ for all $y\in B$.

(3) Let $H,H'\in\mathcal{G}(N)$ and $b:B\to P$ be a base in H. Then *H directly derives H' through b* if $H'\cong\text{REPLACE}(H,\text{repl})$ with $\text{repl}(y)=\text{rhs}(b(y))$ for all $y\in B$. A direct derivation is denoted by $H\underset{b}{\Rightarrow}H'$ or $H\underset{P}{\Rightarrow}H'$ if the underlying set P of productions is emphasized.

(4) A sequence of direct derivations of the form $H_0\underset{P}{\Rightarrow}H_1\underset{P}{\Rightarrow}\cdots\underset{P}{\Rightarrow}H_k$ is called a *derivation* (of length k) from H_0 to H_k and is denoted by $H_0\overset{k}{\underset{P}{\Rightarrow}}H_k$. If $H\cong H'$, we write $H\overset{0}{\underset{P}{\Rightarrow}}H'$ and call this a derivation (of length 0) from H to H'. If, in a derivation $H\overset{k}{\underset{P}{\Rightarrow}}H'$ for some $k\geqslant 0$, the length k does not matter, we write $H\overset{*}{\underset{P}{\Rightarrow}}H'$. We omit P if it is clear from the context.

(5) A *hyperedge-replacement grammar* is a system $\text{HRG}=(N,P,S)$, where N is a set of *nonterminals*, P is a finite set of *productions* (over N), and $S\in N$. The class of all hyperedge-replacement grammars is denoted by $\mathcal{HRG}$.

(6) Given such a grammar HRG $=(N, P, S)$, the *generated graph language* consists of all graphs which can be derived from $(S, 0)^{\bullet}$ by applying productions of P:

$$L(\mathrm{HRG})=\{G\in\mathcal{G}_0 \mid (S,0)^{\bullet} \overset{*}{\underset{P}{\Rightarrow}} G\}.$$

3. Compatible graph properties

In this section, we recall the notion of a compatible graph property which is based on the graph decomposition provided by the Context-freeness Lemma for hyperedge-replacement grammars.

Replacements of different hyperedges cannot interfere with each other, except for some node fusions. Consequently, each derivation starting in a decorated graph H_0 can be restricted to any handle induced by a hyperedge of H_0 and, thus, leads to a so-called fibre of the original derivation. The most interesting fact about these fibres is that the original derivation can be reconstructed from all of them. This result on hyperedge replacement is called "Context-freeness Lemma". A recursive version of it yields decompositions of graphs derived from handles into smaller graphs derived from handles. More precisely, each graph derived from a handle coincides with the right-hand side of a production where all hyperedges are replaced by derived graphs.

Fact 3.1 (Context-freeness Lemma). *Let* HRG $=(N, P, S)\in\mathcal{HRG}$, $A\in N$, $G\in\mathcal{G}_k$, *for some* $k\geqslant 0$, *and* $n\geqslant 0$. *Then there is a derivation* $\mathrm{der}=((A,k)^{\bullet} \overset{n+1}{\underset{P}{\Rightarrow}} G)$ *if and only if there is some* $(A, R)\in P$ *and, for each* $y\in Y_R$, *there exists a derivation of the form* $\mathrm{der}(y)=((\mathrm{lab}_R(y), \mathrm{type}(y))^{\bullet} \overset{n}{\underset{P}{\Rightarrow}} G(y))$, *such that* $G\cong \mathrm{REPLACE}(R, \mathrm{repl})$ *with* $\mathrm{repl}(y)=G(y)$, *for all* $y\in Y_R$.

Remarks. (1) Because the derivation der starts in a handle, the derivation decomposes always into a production $(A, R)\in P$ and a derivation $R \overset{n}{\underset{P}{\Rightarrow}} G$. The latter derivation can be restricted to the handles of R, leading to the derivations $\mathrm{der}(y)$. Vice versa, such fibres can jointly be embedded into R, yielding the graph REPLACE(R, repl) with $\mathrm{repl}(y)=G(y)$, for $y\in Y_R$.

(2) In the situation of the Context-freeness Lemma the graphs $G(y)$ for $y\in Y_R$ are called *y-components* of G.

(3) The proof of the Context-freeness Lemma can be found in [7].

The decomposition (induced by the derivation of a graph) is the key to various decidability results on hyperedge-replacement languages. If a graph property Π can be checked for a derived graph by checking related properties for the involved right-hand side and the smaller derived component graphs, Π is called compatible. (In [10, 7] it is shown that k-colorability, connectedness, the existence of Eulerian and Hamiltonian paths and cycles, and other graph properties are compatible.) For a graph property Π which is compatible with the derivation process of all grammars out of a given class $\mathscr{C}$ of hyperedge-replacement grammars, many interesting questions are known to be decidable for $\mathscr{C}$ (cf. Section 1, see [10, Theorems 4.3 and 4.4]).

Definition 3.2 (*Compatibility*). (1) Let $\mathscr{C}$ be a class of hyperedge-replacement grammars, $I=(I_k)_{k\in\mathbb{N}}$ be an infinite family of finite sets I_k, called *index sets*, *PROP* a decidable predicate,[5] defined on pairs (G, i) with $G\in\mathscr{G}_k$ and $i\in I_k$, and *PROP'* a decidable predicate on triples (R, assign, i) with $R\in\mathscr{G}_k(N)$, a mapping assign on Y_R such that $\text{assign}(y)\in I_{\text{type}(y)}$ for $y\in Y_R$,[6] and $i\in I_k$. Then PROP is called ($\mathscr{C}$, PROP')-*compatible* if, for all $\text{HRG}=(N, P, S)\in\mathscr{C}$, all derivations $(A, k)^\bullet \Rightarrow R \overset{*}{\Rightarrow} G$ in HRG with $(A, R)\in P$, $\text{type}(R)=k$, and $G\in\mathscr{G}_k$, and all $i\in I_k$, the following holds:

PROP(G, i) is true if and only if there is a well-typed mapping assign on Y_R such that PROP'(R, assign, i) is true and PROP$(G(y), \text{assign}(y))$ is true for all $y\in Y_R$.

(2) A predicate Π defined on $G\in\mathscr{G}_k$ is called $\mathscr{C}$-*compatible* if there exist predicates PROP and PROP' and an index $i_k\in I_k$ such that PROP is ($\mathscr{C}$, PROP')-compatible and $\Pi=\text{PROP}(-, i_k)$.[7]

Example 3.3 (3-*Colorability*). Let $\Pi(G)$ be true if and only if G is 3-colorable. A mapping $c: V_G\to\{1, 2, 3\}$ such that $\{v, w\}\in E_G$ implies $c(v)\neq c(w)$ is said to be a 3-*coloring* of G. Moreover, we say that a 3-coloring c of $G\in\mathscr{G}_k$ *agrees with* $i=(i_1, \ldots, i_k)\in\{1, 2, 3\}^k$, if, for all $j\in[k]$, the jth external vertex is colored with i_j, i.e., $c((\text{ext}_G)_j)=i_j$.

In order to show that Π is $\mathscr{HRG}$-compatible, let $I_k=\{1, 2, 3\}^k$, and define PROP by PROP$(G, i)\Leftrightarrow G$ has a 3-coloring which agrees with i. Similarly, define PROP' by PROP'$(R, \text{assign}, i)\Leftrightarrow R-Y_R$ has a 3-coloring c which agrees with i and $\text{assign}(y)=c^*(\text{att}_R(y))$ for all $y\in Y_R$ (c^* is the natural extension of c to sequences, cf. Footnote 3).

Let $\text{HRG}=(N, P, S)\in\mathscr{HRG}$, let $(A, k)^\bullet\Rightarrow R \overset{*}{\Rightarrow} G$ be a derivation in HRG where $(A, R)\in P$, $R\in\mathscr{G}_k(N)$, and $G\in\mathscr{G}_k$, and let, for every $y\in Y$, $G(y)$ be the y-component of G (cf. the remark following the Context-freeness Lemma 3.1). Let $c: V_G\to\{1, 2, 3\}$ be a 3-coloring of G which agrees with $i\in I_k$. Then c', the restriction of c to V_R, is a 3-coloring of (V_R, E_R, ext_R) which agrees with i, and for every $y\in Y_R$, c_y, the restriction of c to $G(y)$, is a 3-coloring of $G(y)$ which agrees with $c^*(\text{att}_R(y))$. Choosing assign such that $\text{assign}(y)=c^*(\text{att}_R(y))$ for $y\in Y_R$,

(*) PROP'(R, assign, i) and PROP$(G(y), \text{assign}(y))$, for all $y\in Y_R$

[5] We assume that all considered predicates are *closed under isomorphisms*, i.e., if a predicate Φ holds for $H\in\mathscr{G}$ ($H\in\mathscr{G}(N)$) and $H\cong H'$, then Φ holds for H', too.

[6] Such mappings will be called *well-typed*, cf. Definition 2.5.

[7] For $i_k\in I_k$, PROP$(-, i_k)$ denotes the unary predicate defined by PROP$(-, i_k)(G)=\text{PROP}(G, i_k)$, for all $G\in\mathscr{G}_k$.

becomes satisfied. On the other hand, for every mapping assign which satisfies (*), we obtain a 3-coloring of G which agrees with i: color all attachment nodes of hyperedges of R as prescribed by assign; by (*), all remaining nodes of G can then be colored consistently, since they belong either to R or to one of the $G(y)$. Altogether, we conclude that PROP(G, i) holds if and only if there exists a mapping assign such that PROP$'(R$, assign, $i)$ holds and PROP$(G(y)$, assign$(y))$ holds for all $y \in Y_R$, i.e., PROP is $(\mathcal{HRG}$, PROP$')$-compatible. Moreover, for $G \in \mathcal{G}_0$, $\Pi(G)$ holds if and only if PROP$(G, (\ ))$ holds, where $(\ )$ denotes the only member of I_0. Hence, the predicate Π defined on $G \in \mathcal{G}_0$ is $\mathcal{HRG}$-compatible.

$\mathcal{C}$-compatible predicates are closed under Boolean operations (cf. [7]).

Fact 3.4 (Closure under Boolean operations). *$\mathcal{C}$-compatible predicates are closed under Boolean operations, i.e. if Π_1 and Π_2 are $\mathcal{C}$-compatible, then the predicates $(\Pi_1 \wedge \Pi_2)$, $(\Pi_1 \vee \Pi_2)$, and $(\neg \Pi_1)$, with*
- $(\Pi_1 \wedge \Pi_2)(G) \Leftrightarrow \Pi_1(G) \wedge \Pi_2(G)$
- $(\Pi_1 \vee \Pi_2)(G) \Leftrightarrow \Pi_1(G) \vee \Pi_2(G)$
- $(\neg \Pi_1)(G) \Leftrightarrow \neg \Pi_1(G)$

are $\mathcal{C}$-compatible.

In the following, so-called proper compatible predicates will be of interest. Roughly speaking, a predicate PROP is proper if for each graph $G \in \mathcal{G}_k$, there is only one index $i \in I_k$ for which PROP(G, i) holds. In this case, the set $\{i \in I_k \mid \text{PROP}(G, i) \text{ for some } G \in \mathcal{G}_k\}$ determines a decomposition of the set $\mathcal{G}_k$ into a finite number of nonempty classes PROP$(i) = \{G \in \mathcal{G}_k \mid \text{PROP}(G, i)\}$.

Definition 3.5 (*Proper compatible predicates*). A $(\mathcal{C}$, PROP$')$-compatible predicate PROP is said to be *proper*, if for each $G \in \mathcal{G}_k$, there exists uniquely one $i \in I_k$ such that PROP(G, i) is true, and for each $R \in \mathcal{G}_k(N)$ and each well-typed mapping assign, there exists uniquely one $i \in I_k$ such that PROP$'(R$, assign, $i)$ is true.

In [7], it is shown that for each compatible predicate Π, a proper $(\mathcal{C}$, PROP$')$-compatible predicate PROP can be found such that for $G \in \mathcal{G}_k$, $\Pi(G)$ may be expressed by the disjunction of some PROP(G, i) $(i \in I_k)$.

Fact 3.6 (Existence of proper compatible predicates). *For each $\mathcal{C}$-compatible predicate Π on $\mathcal{G}_n$, there is a family $I = (I_k)_{k \in \mathbb{N}}$ of finite, nonempty sets I_k, decidable predicates* PROP *and* PROP$'$ *and a subset $I'_n \subseteq I_n$ such that* PROP *is a proper $(\mathcal{C}$,* PROP$')$*-compatible predicate and $\Pi = \bigvee_{i \in I'_n}$* PROP$(-, i)$.

4. Finite graph properties

In this section, we compare finite graph properties in the sense of Lengauer and Wanke [12, 13] with graph properties compatible with the derivation process of hyperedge-replacement grammars.

The main notion in [12] is the notion of replaceability of a graph by a graph. Informally, graphs G and G' are replaceable with respect to a given graph property if G and G' behave equally with respect to this property whenever they are inserted into the same context.

Notation 4.1. Let $G, G' \in \mathcal{G}_k$. Then $G \circ G'$ denotes the 0-graph

$$G \circ G' = \hat{G}[y/G'],$$

where $\hat{G} = (V_G, E_G, \{y\}, \text{lab}, \text{att}, \lambda)$ with $\text{lab}(y) = A$, for some $A \in N$, and $\text{att}(y) = \text{ext}_G$.

The definition of REPLACE ensures the commutativity of $\circ$.

Fact 4.2 (Commutativity). *The operation* $\circ$ *is commutative*: *For* $G, G' \in \mathcal{G}_k$, $G \circ G' \cong G' \circ G$.

Definition 4.3 (Replaceability). Let Π be a property defined on 0-graphs. Then $G, G' \in \mathcal{G}_k$ are *replaceable* with respect to Π, denoted by $G \sim_\Pi G'$, if, for all $C \in \mathcal{G}_k$, $\Pi(C \circ G) = \Pi(C \circ G')$.

Remark. $\sim_\Pi$ is an equivalence relation. For $G \in \mathcal{G}_k$, the equivalence class of G induced by $\sim_\Pi$, $\{G' \in \mathcal{G}_k \mid G' \sim_\Pi G\}$, is denoted by $[G]_\Pi$. Then the set of all such equivalence classes is denoted by $M_\Pi(k)$; its cardinality is denoted by $|M_\Pi(k)|$.

Lemma 4.4 (Replaceable graphs possess the same property). *Let* $G, G' \in \mathcal{G}_k$. *Then* $G \sim_\Pi G'$ *implies* $\Pi(U(G)) = \Pi(U(G'))$.

Proof. Let $G, G' \in \mathcal{G}_k$ and $G \sim_\Pi G'$. Then for all $C \in \mathcal{G}_k$, $\Pi(C \circ G) = \Pi(C \circ G')$. In particular, $\Pi(\text{EMPTY}_k \circ G) = \Pi(\text{EMPTY}_k \circ G')$ for $\text{EMPTY}_k = (\{1, \dots, k\}, \emptyset, 1 \dots k) \in \mathcal{G}_k$. On the other hand, $\text{EMPTY}_k \circ G \cong U(G)$ and $\text{EMPTY}_k \circ G' \cong U(G')$. Since Π is closed under isomorphisms, we get $\Pi(U(G)) = \Pi(\text{EMPTY}_k \circ G) = \Pi(\text{EMPTY}_k \circ G') = \Pi(U(G'))$. □

Lemma 4.5 (Closure under replacement). *Let* $R \in \mathcal{G}(N)$, *and* repl, repl′: $Y_R \to \mathcal{G}$ *be well-typed mappings with* $\text{repl}(y) \sim_\Pi \text{repl}'(y)$ *for* $y \in Y_R$. *Then we have* $\text{REPLACE}(R, \text{repl}) \sim_\Pi \text{REPLACE}(R, \text{repl}')$.

Proof. Let $R \in \mathcal{G}_k(N)$, $Y_R = \{y_1, \dots, y_n\}$, and repl, repl′: $Y_R \to \mathcal{G}$ be two well-typed mappings with $\text{repl}(y_i) \sim_\Pi \text{repl}'(y_i)$ for $i = 1, \dots, n$. Then we define auxiliary mappings repl_j: $Y_R \to \mathcal{G}$ with

$$\text{repl}_j(y_i) = \begin{cases} \text{repl}'(y_i) & \text{if } i < j \\ \text{repl}(y_i) & \text{if } i \geq j \end{cases}$$

for $j = 1, \dots, n+1$. In particular, $\text{repl}_1 = \text{repl}$ and $\text{repl}_{n+1} = \text{repl}'$. Let $\text{repl}_j|$ denote the restriction of repl_j to $Y_R - \{y_j\}$ and let $R_j = \text{REPLACE}(R, \text{repl}_j|)$. Then R_j is the k-graph R in which all hyperedges with index $i < j$ are replaced according to repl′ and all hyperedges with index $i>j$ are replaced according to repl. Moreover, let $G_j=\text{repl}(y_j)$ and $G'_j=\text{repl}'(y_j)$. Since simultaneous replacement of hyperedges may be sequentialized and sequential replacements of different hyperedges may be done simultaneously, we get $\text{REPLACE}(R, \text{repl}_j)=\text{REPLACE}(R, \text{repl}_j|)[y_j/G_j]=R_j[y_j/G_j]$ and $\text{REPLACE}(R, \text{repl}_{j+1})=\text{REPLACE}(R, \text{repl}_j|)[y_j/G'_j]=R_j[y_j/G'_j]$. Then for all $C\in\mathcal{G}_k$ and all $j\in[n]$,

$$\begin{aligned}
&\Pi(C\circ\text{REPLACE}(R, \text{repl}_j))\\
&\quad=\Pi(C\circ(R_j[y_j/G_j]))\\
&\quad=\Pi((C\circ R_j)[y_j/G_j]) && \text{(by the associativity of REPLACE)}\\
&\quad=\Pi((C\circ R_j)[y_j/G'_j]) && \text{(since } G_j\sim_\Pi G'_j)\\
&\quad=\Pi(C\circ(R_j[y_j/G'_j])) && \text{(by the associativity of REPLACE)}\\
&\quad=\Pi(C\circ\text{REPLACE}(R, \text{repl}_{j+1})).
\end{aligned}$$

Hence, we have $\Pi(C\circ\text{REPLACE}(R, \text{repl}))=\Pi(C\circ\text{REPLACE}(R, \text{repl}_1))=\cdots=\Pi(C\circ\text{REPLACE}(R, \text{repl}_{n+1}))=\Pi(C\circ\text{REPLACE}(R, \text{repl}'))$ for all $C\in\mathcal{G}_k$, i.e., $\text{REPLACE}(R, \text{repl})\sim_\Pi\text{REPLACE}(R, \text{repl}')$. □

For all $k\in\mathbb{N}$, $M_\Pi(k)$ may be finite or infinite, as the following examples show.

Example 4.6 (*3-Colorability*). Let $\Pi(G)\Leftrightarrow G$ is 3-colorable. Then for every $k\in\mathbb{N}$, $M_\Pi(k)$ is finite. This may be seen as follows: Define for every k-graph G the set

$$C(G)=\{i\in\{1,2,3\}^k\mid G \text{ has a 3-coloring which agrees with } i\}$$

(cf. Example 3.3). Clearly, $C(G)=C(G')$ implies $G\sim_\Pi G'$, and as there are at most 2^{3^k} possibilities for $C(G)$, $M_\Pi(k)$ must be finite.

Example 4.7 (*Regularity*). Let $\Pi(G)\Leftrightarrow G$ be regular, i.e., all vertices of G have the same degree. Then for every $k\in\mathbb{N}$, $M_\Pi(k)$ is infinite: For every $l\in\mathbb{N}$, let G_l be the union of k isolated external nodes and the complete bipartite graph $K_{l,l}$. Then $C\circ G_l$ is regular if and only if C is l-regular and, therefore, $G_l\not\sim_\Pi G_{l'}$, if $l\neq l'$.

In the following, we focus on those properties Π for which the set of induced classes is finite. In [12, 13], such properties are called *finite* graph properties.

Definition 4.8 (*Finite graph property*). Let Π be a decidable graph property. Then Π is said to be *k-finite* if the set $M_\Pi(k)$ is finite. Π is called *finite* if $M_\Pi(k)$ is finite for every $k\in\mathbb{N}$.

Then the following fact holds (see [12]).

Fact 4.9 (Lengauer and Wanke [12, Theorems 3, 4, and 6]). (1) *For each* $k \in \mathbb{N}$, $(k+1)$*-finiteness of a graph property* Π *implies k-finiteness of* Π.

(2) *For each* $k \in \mathbb{N}$, *there exist graph properties that are k-finite but not* $(k+1)$*-finite.*

(3) *If* $|M_\Pi(k)|$ *is computable (as a function in k) then, for each k, one can compute a set of k-graphs that contains exactly one representative for each class in* $M_\Pi(k)$. *Given such a set* $\mathrm{REP}_\Pi(k)$ *of representatives, the replaceability w.r.t.* Π *can be decided, i.e., for each two graphs* $G, G' \in \mathcal{G}_k$, $G \sim_\Pi G'$ *can be tested.*

Now we are able to formulate the main results of this section.

Theorem 4.10 (Finiteness implies compatibility). *If* Π *is a finite graph property and the mapping* $k \mapsto |M_\Pi(k)|$ *is computable, then* Π *is* $\mathcal{HRG}$*-compatible.*

Proof. Let Π be a finite graph property and let $n: k \mapsto |M_\Pi(k)|$ be computable. Then, Π is k-finite for all $k \in \mathbb{N}$ and, for each $k \in \mathbb{N}$, there exist k-graphs $G_{k,1}, \ldots, G_{k,n(k)} \in \mathcal{G}_k$ such that $M_\Pi(k) = \{[G_{k,1}]_\Pi, \ldots, [G_{k,n(k)}]_\Pi\}$. For proving the $\mathcal{HRG}$-compatibility of Π, we proceed as follows. First, we define predicates PROP and PROP′ and show the $(\mathcal{HRG}$, PROP′)-compatibility of PROP. In a second step, we show how Π and PROP are related and derive the $\mathcal{HRG}$-compatibility of Π from the $(\mathcal{HRG}$, PROP′)-compatibility of PROP.

(1) Define

- a family $I = (I_k)_{k \in \mathbb{N}}$ of finite index sets I_k by $I_k = \{1, \ldots, n(k)\}$,
- a predicate PROP on pairs (G, i) with $G \in \mathcal{G}_k$ and $i \in I_k$ by

$$\mathrm{PROP}(G, i) \Leftrightarrow G \sim_\Pi G_{k,i},$$

- an auxiliary predicate PROP′ on triples (R, assign, i) with $R \in \mathcal{G}_k(N)$, a well-typed mapping $\mathrm{assign}: Y_R \to I$, and $i \in I_k$ by

$$\mathrm{PROP}'(R, \mathrm{assign}, i) \Leftrightarrow \mathrm{REPLACE}(R, \mathrm{repl}) \sim_\Pi G_{k,i},$$

where $\mathrm{repl}(y) = G_{\mathrm{type}(y), \mathrm{assign}(y)}$ for $y \in Y_R$.

Note that, for each $k \in \mathbb{N}$, I_k is finite and nonempty. Moreover, PROP and PROP′ are decidable because, for each $k \in \mathbb{N}$, a realization of $M_\Pi(k)$ can be computed and the replaceability w.r.t Π can be decided.

Then PROP is $(\mathcal{HRG}$, PROP′)-compatible. This may be seen as follows. Let $\mathrm{HRG} = (N, P, S)$ be a hyperedge-replacement grammar, $(A, k)^\bullet \Rightarrow R \overset{*}{\Rightarrow} G$ be a derivation in HRG with $A \in N$, $k \in \mathbb{N}$, and $G \in \mathcal{G}_k$, $(\mathrm{lab}_R(y), \mathrm{type}(y))^\bullet \overset{*}{\Rightarrow} G(y)$ be the fibre induced by $y \in Y_R$, and $i \in I_k$.

First, let $\mathrm{PROP}(G, i)$ be satisfied. Then, by definition, $G \sim_\Pi G_{k,i}$. For each $G(y) \in \mathcal{G}_{\mathrm{type}(y)}$, there is exactly one index $i(y) \in I_{\mathrm{type}(y)}$ such that $G(y) \sim_\Pi G_{\mathrm{type}(y), i(y)}$. Choosing assign such the $\mathrm{assign}(y) = i(y)$ for $y \in Y_R$. $\mathrm{PROP}(G(y), \mathrm{assign}(y))$ becomes satisfied for all $y \in Y_R$. By Lemma 4.5, $\mathrm{REPLACE}(R, \mathrm{repl}) \sim_\Pi \mathrm{REPLACE}(R, \mathrm{repl}')$, where $\mathrm{repl}(y) = G(y)$ and $\mathrm{repl}'(y) = G_{\mathrm{type}(y), \mathrm{assign}(y)}$ for $y \in Y_R$. Since $\sim_\Pi$ is symmetric and REPLACE(R, repl)$=G$, we obtain REPLACE(R, repl$'$) $\sim_\Pi$ REPLACE(R, repl)$=$ $G \sim_\Pi G_{k,i}$. Hence, PROP$'(R$, assign, $i)$ is satisfied, too.

Conversely, assume that there is a well-typed mapping assign on Y_R such that PROP$'(R$, assign, $i)$ as well as PROP($G(y)$, assign(y)) hold for all $y \in Y_R$. Then we have $G(y) \sim_\Pi G_{\text{type}(y),\,\text{assign}(y)}$ for $y \in Y_R$, and, consequently, REPLACE(R, repl) $\sim_\Pi$ REPLACE(R, repl$'$), where repl$(y)=G(y)$ and repl$'(y)=$ $G_{\text{type}(y),\,\text{assign}(y)}$ for $y \in Y_R$. Moreover, we have REPLACE(R, repl$'$) $\sim_\Pi G_{k,i}$. Since $G=$ REPLACE(R, repl) and $\sim_\Pi$ is transitive, we get $G \sim_\Pi G_{k,i}$, i.e., PROP(G, i) is satisfied.

(2) Based on the ($\mathscr{HRG}$, PROP$'$)-compatibility of PROP, we will show the $\mathscr{HRG}$-compatibility of Π. Let $G \in \mathscr{G}_0$ and $i \in I_0$ such that $G \sim_\Pi G_{0,i}$. (Note that there exists exactly one index with this property.) Then, by Lemma 4.4, $\Pi(G)=\Pi(G_{0,i})$. Now, let $\text{SAT}_0=\{i \in I_0 \mid \Pi(G_{0,i}) \text{ is satisfied}\}$. Then, for each $G \in \mathscr{G}_0$,

$$\Pi(G) \Leftrightarrow \bigvee_{i \in \text{SAT}_0} \text{PROP}(G, i).$$

By definition of compatibility, for each $i \in I_0$, PROP$(-, i)$ is $\mathscr{HRG}$-compatible. From Fact 3.4 it follows that the predicate

$$\Pi = \bigvee_{i \in \text{SAT}_0} \text{PROP}(-, i)$$

is $\mathscr{HRG}$-compatible. □

We cannot show a direct converse of Theorem 4.10, but a slightly weaker assertion holds.

Theorem 4.11 (Compatibility implies finiteness). *Let Π be $\mathscr{HRG}$-compatible. Then Π is a finite graph property and there is a computable function f such that $|M_\Pi(k)| \leq f(k)$ for all $k \in \mathbb{N}$.*

Proof. Let Π be a $\mathscr{HRG}$-compatible predicate Π on $\mathscr{G}_0$. Then, by Fact 3.6, there is a family $I=(I_k)_{k \in \mathbb{N}}$ of finite index sets I_k, decidable predicates PROP and PROP$'$, and a subset $I'_0 \subseteq I_0$ such that PROP is a proper ($\mathscr{C}$, PROP$'$)-compatible predicate and $\Pi = \bigvee_{i \in I'_0}$ PROP$(-, i)$. In the following, we will characterize replaceability by sets of indices. It can be shown that for all $k \in \mathbb{N}$ and all $G, G' \in \mathscr{G}_k$,

(∗) $G \sim_\Pi G'$ if and only if $I_k(G)=I_k(G')$,

where for $G \in \mathscr{G}_k$, $I_k(G)$ denotes the set of indices

$$I_k(G)=\{j \in I_{k,ex} \mid \exists \text{assign}: \{y\} \rightarrow I_k: \bigvee_{i \in I'_0} \text{PROP}'(\hat{G}, \text{assign}, i) \text{ and assign}(y)=j\}$$

and $I_{k,ex}=\{i \in I_k \mid \exists G \in \mathscr{G}_k: \text{PROP}(G, i)\}$. By (∗), every equivalence class on $\mathscr{G}_k$ is of the form $\{G' \in \mathscr{G}_k \mid I_k(G')=\text{IND}\}$ for some index set IND $\subseteq I_k$. Moreover, for each $k \in \mathbb{N}$, the index set I_k is finite. Therefore, the number of different equivalence classes is equal to the number of index sets IND with $\{G' \in \mathcal{G}_k \mid I_k(G') = \text{IND}\} \neq \emptyset$, which is bounded by $|\mathcal{P}(I_k)|$, the cardinality of the powerset of I_k.

It remains to prove (∗).

$$G \sim_\Pi G'$$

$$\Leftrightarrow \forall H \in \mathcal{G}_k : \Pi(H \circ G) = \Pi(H \circ G') \quad \text{(by Definition 4.3)}$$

$$\Leftrightarrow \forall H \in \mathcal{G}_k : \Pi(G \circ H) = \Pi(G' \circ H) \quad \text{(by Fact 4.2)}$$

$$\Leftrightarrow \forall H \in \mathcal{G}_k : \bigvee_{i \in I_0'} \text{PROP}(G \circ H, i) = \bigvee_{i' \in I_0'} \text{PROP}(G' \circ H, i')$$

(by the relationship of Π and PROP)

$$\Leftrightarrow (A) \begin{cases} \forall H \in \mathcal{G}_k : \\ \Big[\bigvee_{i \in I_0'} \bigvee_{\text{assign}:\{y\} \to I_k} [\text{PROP}'(\hat{G}, \text{assign}, i) \wedge \text{PROP}(H, \text{assign}(y))] \\ \Leftrightarrow \bigvee_{i' \in I_0'} \bigvee_{\text{assign}':\{y\} \to I_k} [\text{PROP}'(\hat{G}'. \text{assign}', i') \wedge \text{PROP}(H, \text{assign}'(y))] \Big] \end{cases}$$

(by the $\mathcal{HRG}$-compatibility of PROP)

$$\Leftrightarrow (B) \; I_k(G) = I_k(G').$$

The last equivalence can be shown as follows:

"⇐" Let (B) be satisfied.

If $I_k(G) = I_k(G') = \emptyset$, then PROP$'(\hat{G}$, assign, $i)$ and PROP$'(\hat{G}'$, assign, $i)$ are false for all $i \in I'_0$ and all assign with assign$(y) \in I_{k,ex}$. Moreover, for a mapping assign with assign$(y) \in I_k - I_{k,ex}$, PROP$(H$, assign$(y))$ is false for all $H \in \mathcal{G}_k$.

Otherwise, let $H \in \mathcal{G}_k$ be an arbitrary k-graph. Then, by the properness of PROP, there exists exactly one index j such that PROP(H, j) is satisfied.

Case 1. $j \in I_k(G)$ $(= I_k(G'))$. Choose assign and assign$'$ such that assign$(y) = j$ and assign$'(y) = j$. Then,

$$\bigvee_{i \in I_0'} \bigvee_{\text{assign}:\{y\} \to I_k} [\text{PROP}'(\hat{G}, \text{assign}, i) \wedge \text{PROP}(H, \text{assign}(y))]$$

as well as

$$\bigvee_{i' \in I_0'} \bigvee_{\text{assign}':\{y\} \to I_k} [\text{PROP}'(\hat{G}', \text{assign}', i') \wedge \text{PROP}(H, \text{assign}'(y))]$$

become true.

Case 2. $j \notin I_k(G)$ $(= I_k(G'))$. Then PROP$(H$, assign$(y))$ is false for all assign with assign$(y) \in I_k(G)$, (since we have assign$(y) \neq j$) and, for assign with assign$(y) \notin I_k(G)$,

- assign$(y) \notin I_{k,ex}$ and PROP$(H$, assign$(y))$ is false or
- assign$(y) \in I_{k,ex}$ and $\bigvee_{i \in I_0'}$ PROP$'(\hat{G}$, assign, $i)$ is false.

Analogously, for all assign$'$, $\bigvee_{i' \in I_0'}$ PROP$'(\hat{G}'$, assign$'$, $i') \wedge$ PROP$(H$, assign$'(y))$ is false.

"$\Rightarrow$" Assume that (B) does not hold, i.e., $I_k(G) \neq I_k(G')$. Without loss of generality, there is an index $\text{assign}(y) \in I_k(G) - I_k(G')$. By definition of $I_k(G)$, there exists $H \in \mathscr{G}_k$ for which $\text{PROP}(H, \text{assign}(y))$ is satisfied. Then, $\bigvee_{i \in I_0'} \text{PROP}'(\hat{G}, \text{assign}, i)$ as well as $\text{PROP}(H, \text{assign}(y))$ are true. On the other hand, $\text{PROP}(H, \text{assign}'(y))$ is false for all those assign′ for which $\text{assign}'(y) \in I_k(G')$, (because then $\text{assign}'(y) \neq \text{assign}(y)$); and for all assign′ with $\text{assign}'(y) \notin I_k(G')$, $\text{assign}'(y) \notin I_{k,ex}$ and $\text{PROP}(H, \text{assign}'(y))$ is false or $\bigvee_{i' \in I_0'} \text{PROP}'(\hat{G}', \text{assign}', i)$ is false. Therefore, (A) is not satisfied. □

Remark. In the proof of the second theorem, compatibility is needed for a certain type of grammar only. Thus, a stronger formulation is possible, assuming only $\mathscr{C}$-compatibility, where for each k and for each pair (R, G) of k-graphs, $\mathscr{C}$ contains a grammar with the two rules $(S, \hat{R})$, and (A, G) (cf. Notation 4.1).

The decidability results in [12, 13] only require the computability of an upper bound on the index of $\sim_\Pi$ on every $\mathscr{G}_k$. Thus, Theorem 4.11, although not a proper converse of Theorem 4.10, is strong enough to transfer these results to $\mathscr{HRG}$-compatible properties (corresponding results for (hyper)edge-replacement grammars are proved directly in [11].

5. Inductive graph properties

In this section, we compare compatibility with Courcelle's notion of an effectively locally finite and inductive family of predicates, which is based on his algebra of graphs. First, we will briefly recall the operations of this algebra, as given in [4], and will then show how they can be simulated by hyeredge replacement, and vice versa.

Definition 5.1 (*Operations on graphs*). The operation set $\mathscr{E}$ on $(\mathscr{G}_k)_{k \in \mathbb{N}}$ consists of the following operations:

- $\mathbf{0}$: $\to \mathscr{G}_0$, where $\mathbf{0}$ is the empty graph.
- $\mathbf{2}$: $\to \mathscr{G}_2$, where $\mathbf{2}$ is the 2-graph with a single edge joining two external nodes.
- $\text{id}_n : \mathscr{G}_n \to \mathscr{G}_n$, for every $n \in \mathbb{N}$, where $\text{id}_n(G) = G$.[8]
- $\oplus_{m,n} : \mathscr{G}_n \times \mathscr{G}_m \to \mathscr{G}_{n+m}$, for every $(n, m) \in \mathbb{N}^2$, where $\oplus_{m,n}(G, H) = (V_G + V_H, E_G + E_H, \text{ext}_G \cdot \text{ext}_H)$.[9]
- $\theta_{n,\delta} : \mathscr{G}_n \to \mathscr{G}_n$, for every $n \in \mathbb{N}$ and every equivalence relation δ on $[n]$, where $\theta_{n,\delta}(G) = G/\delta'$ and δ' is the extension of the equivalence relation $\{(v_i, v_j) | (i, j) \in \delta\}$ on EXT_G to V_G for $\text{ext}_G = v_1 \ldots v_n$.
- $\sigma_{n,p,a} : \mathscr{G}_n \to \mathscr{G}_p$, for every $(n, p) \in \mathbb{N}^2$, and every $a : [p] \to [n]$, where $\sigma_{n,p,a}(G) = (V_G, E_G, \text{ext})$ and $\text{ext} = v_{a(1)} \ldots v_{a(p)}$ for $\text{ext}_G = v_1 \ldots v_n$.

Notation 5.2. Let $\mathscr{F}'$ be a set of operations. Then closure $(\mathscr{F}')$ denotes the closure of

[8] The identity functions are not explicitly given in [4], they are added here for convenience only.
[9] $\text{ext}_G \cdot \text{ext}_H$ denotes the concatenation of ext_G and ext_H.

$\mathscr{F}'$ under composition, i.e., the set of those operations that can be expressed by linear terms over $\mathscr{F}'$.[10] closure($\mathscr{E}$) will henceforth be denoted by $\mathscr{F}$.

($\mathscr{G}$, $\mathscr{F}$) is a many-sorted algebra. In the following lemma, we will show how the operations in $\mathscr{F}$ can be simulated by hyperedge replacement, and vice versa.

Lemma 5.3. (1) *For every operation $f\colon \mathscr{G}_{k_1} \times \cdots \times \mathscr{G}_{k_r} \to \mathscr{G}_k$ in $\mathscr{F}$, there is a decorated graph H_f in $\mathscr{G}_k(N)$ with $Y_H = \{y_1, \ldots, y_r\}$ such that for all tuples $(G_1, \ldots, G_r) \in \mathscr{G}_{k_1} \times \cdots \times \mathscr{G}_{k_r}$,*

$$f(G_1, \ldots, G_r) \cong H_f[y_1/G_1, \ldots, y_r/G_r].$$

(2) *For every decorated graph H in $\mathscr{G}_k(N)$ with $Y_H = \{y_1, \ldots, y_r\}$, there is some operation $f_H\colon \mathscr{G}_{\text{type}(y_1)} \times \cdots \times \mathscr{G}_{\text{type}(y_r)} \to \mathscr{G}_k$ in $\mathscr{F}$ such that for all $(G_1, \ldots, G_r) \in \mathscr{G}_{\text{type}(y_1)} \times \cdots \times \mathscr{G}_{\text{type}(y_r)}$,*

$$H[y_1/G_1, \ldots, y_r/G_r] \cong f_H(G_1, \ldots, G_r).$$

Proof. (1) The proof is by induction on the structure of f.

For the operations $f = \mathbf{0}$ and $f = \mathbf{2}$, the empty graph and the 2-graph with a single edge are the corresponding graphs. For the operation $f = \text{id}_n$, $H_f = (A, n)^\bullet$, with $A \in N$ is the corresponding decorated graph: For every $G \in \mathscr{G}_n$, $H_f[y/G] \cong G = f(G)$.

Let now $f = \oplus_{m,n}(f_1, f_2)$, where $f_1 : \mathscr{G}_{k_1} \times \cdots \times \mathscr{G}_{k_s} \to \mathscr{G}_m$ and $f_2 : \mathscr{G}_{k_{s+1}} \times \cdots \times \mathscr{G}_{k_r} \to \mathscr{G}_n$. By the induction hypothesis, there are decorated graphs H_1, H_2 satisfying the assertion of the lemma for f_1 and f_2, respectively. To construct H_f, take the decorated graph $H_{m,n} = ([m+n], \emptyset, \{y_1, y_2\}, \text{lab}, \text{att}, \text{ext})$, where $\text{lab}(y_i) = A_i$, for $i = 1, 2$, and some $A_1, A_2 \in N$, $\text{att}(y_1) = 1 \ldots m$, $\text{att}(y_2) = m+1 \ldots m+n$, and $\text{ext} = 1 \ldots m+n$ (i.e., $H_{m,n}$ is the disjoint union of an m-handle and an n-handle with their external nodes concatenated).

Define H_f as $H_f = H_{m,n}[y_1/H_1, y_2/H_2]$. If $Y_{H_1} = \{y_1, \ldots, y_s\}$ and $Y_{H_2} = \{y_{s+1}, \ldots, y_r\}$, then $Y_{H_f} = \{y_1, \ldots, y_r\}$. Let $(G_1, \ldots, G_r) \in \mathscr{G}_{k_1} \times \cdots \times \mathscr{G}_{k_r}$, and let $\text{repl}_j(y_i) = G_i$, for $j = 1, 2$. Then we have

$$\begin{aligned}
&f(G_1, \ldots, G_r)\\
&\quad = \oplus_{m,n}(f_1(G_1, \ldots, G_s), f_2(G_{s+1}, \ldots, G_r)),\\
&\quad\quad \text{since } f = \oplus_{m,n}(f_1, f_2)\\
&\quad \cong \oplus_{m,n}(\text{REPLACE}(H_1, \text{repl}_1), \text{REPLACE}(H_2, \text{repl}_2)),\\
&\quad\quad \text{(by induction hypothesis)}\\
&\quad \cong H_{m,n}[y_1/\text{REPLACE}(H_1, \text{repl}_1), y_2/\text{REPLACE}(H_2, \text{repl}_2)],\\
&\quad\quad \text{(by definition of } H_{m,n})
\end{aligned}$$

[10] A linear term is a term in which no variable occurs more than once.

$$\cong H_f[y_1/G_1,\dots,y_r/G_r]$$

(by definition of H_f and elementary properties of REPLACE).

Similarly, the induction step is shown for $f=\theta_{n,\delta}(g)$ and $f=\sigma_{n,p,a}(g)$, using $H_{n,\delta}=(A,n)^\bullet/\delta'$, and $H_{n,p,a}=([n],\emptyset,\{y\},\text{lab},\text{att},\text{ext})$ with $\text{lab}(y)=A, \text{att}(y)=1\dots n$, and $\text{ext}=a(1)\dots a(p)$, respectively.

(2) Here, the proof is by induction on the number r of hyperedges in H.

For $r=0$, H is a k-graph without hyperedges, and by Proposition 2.8 in [4], H is the value of a graph expression, i.e., there is an operation f: $\to\mathcal{G}_k$ in $\mathcal{F}$ such that $f\cong H$. Let $r\geqslant 1$, and let $\text{type}(y_i)=k_i$, for $i=1,\dots,r$. Define H' to be the $k+k_r$-graph obtained from H by removing y_r and extending the sequence of external nodes by those attached to y_r. By the induction hypothesis, there is an operation f' in $\mathcal{F}$ satisfying the assertion of the lemma for H'. Define f by

$$f=\sigma_{n,k,a}(\theta_{n,\delta}(\oplus_{k+k_r,k_r}(f',\text{id}_{k_r}))),$$

where $n=k+2k_r$, δ is defined on $[k+2k_r]$ by $k+i\equiv_\delta k+k_r+i$, $i=1,\dots,k_r$, and $a:[k]\to[k+2k_r]$ is given by $a(i)=i$, for $i=1,\dots,k$. Then, for any $(G_1,\dots,G_r)\in\mathcal{G}_{k_1}\times\cdots\times\mathcal{G}_{k_r}$, we have

$$\begin{aligned}&f(G_1,\dots,G_r)\\ &\quad=\sigma_{n,k,a}(\theta_{n,\delta}(\oplus_{k+k_r,k_r}(f'(G_1,\dots,G_{r-1}),G_r)))\\ &\quad=\sigma_{n,k,a}(\theta_{n,\delta}(\oplus_{k+k_r,k_r}(H'[y_1/G_1,\dots,y_{r-1}/G_{r-1}],G_r)))\\ &\quad=H[y_1/G_1,\dots,y_r/G_r]. \qquad \square\end{aligned}$$

Remarks. (1) The constructions given in the proof of Lemma 5.3 are inverse to each other in the sense that, for all $f:\mathcal{G}_{k_1}\times\cdots\times\mathcal{G}_{k_r}\to\mathcal{G}_k$, and $(G_1,\dots,G_r)\in\mathcal{G}_{k_1}\times\cdots\times\mathcal{G}_{k_r}$, we have $f_{H_f}(G_1,\dots,G_r)\cong f(G_1,\dots,G_r)$, and, similarly, for every decorated k-graph H, letting $H'=H_{f_H}$, there is a bijection $b: Y_{H'}\to Y_H$ such that for every well-typed mapping repl: $Y_{H'}\to\mathcal{G}$ we have $\text{REPLACE}(H',\text{repl})\cong\text{REPLACE}(H,\text{repl}\circ b^{-1})$.

(2) If we extend the operations in $\mathcal{F}$ to $\mathcal{G}(N)$, i.e., to decorated graphs with hyperedges, then for any $f\in\mathcal{F}$, $f:\mathcal{G}_{k_1}(N)\times\cdots\times\mathcal{G}_{k_r}(N)\to\mathcal{G}_k(N)$, the decorated graph H_f of Lemma 5.3(1) can be described as

$$H_f=f((A_1,k_1)^\bullet,\dots,(A_r,k_r)^\bullet).$$

Next, we briefly recall from [4] the notion of an effectively locally finite and inductive family of predicates on $\mathcal{G}$, and then show that this notion is equivalent to compatibility as defined in Section 3.

Definition 5.4 (*$\mathcal{F}'$-inductivity*). Let $\mathcal{F}'\subseteq\mathcal{F}$ and let, for every $k\in\mathbb{N}$, P_k be a set of predicates on $\mathcal{G}_k$. The family $\mathcal{P}=(P_k)_{k\in\mathbb{N}}$ is called

- *locally finite*, if every P_k is finite,

- *$\mathcal{F}'$-inductive*, if for every operation $f: \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r} \to \mathcal{G}_k$ in $\mathcal{F}'$ and for every $p \in P_k$, there is a Boolean expression $B_{f,p}$ with variables $(x_q^i)_{q \in P_{k_i}}^{i=1,\dots,r}$ such that for all $(G_1, \dots, G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$,

$$p(f(G_1, \dots, G_r)) \Leftrightarrow B_{f,p}[(x_q^i | q(G_i))_{q \in P_{k_i}}^{i=1,\dots,r}],$$

  where $B[x|v]$ denotes the formula B with variable x substituted by v.
- *effectively locally finite and $\mathcal{F}'$-inductive*, if $\mathcal{P}$ is both, locally finite and $\mathcal{F}'$-inductive, and, additionally, the following mappings are all computable
  - $k \mapsto P_k$,
  - $(p, G) \mapsto p(G)$,
  - $(f, p) \mapsto B_{f,p}$, where $B_{f,p}$ is as above.

Remarks. (1) The definition of inductivity given above differs from the one in [4] in one aspect: In [4], $B_{f,p}$ contains only some of the variables x_q^i, and is given together with a corresponding choice of predicates. Since we are dealing exclusively with locally finite families, this complication is unnecessary here.

(2) If $\mathcal{P}$ is $\mathcal{F}'$-inductive and $\mathcal{F}'' \subseteq \mathcal{F}'$, then $\mathcal{P}$ is $\mathcal{F}''$-inductive.

Example 5.5 (*3-Colorability*). We define a family of predicates related to 3-colorability. For every $k \in \mathbb{N}$ and every $i \in \{1,2,3\}^k$ we define the predicate p_i on $\mathcal{G}_k$ by

$$p_i(G) \Leftrightarrow G \text{ has a 3-coloring which agrees with } i$$

(cf. Example 3.3). Clearly, with $P_k = \{p_i | i \in \{1,2,3\}^k\}$, the family $\mathcal{P} = (P_k)_{k \in \mathbb{N}}$ is locally finite. To see that $\mathcal{P}$ is also $\mathcal{E}$-inductive, we first observe that $P_0 = \{p_0\}$ and $P_2 = \{p_{lm} \,|\, l, m \in \{1,2,3\}\}$ where $p_0(G) \Leftrightarrow G$ is 3-colorable and $p_{lm}(G) \Leftrightarrow G$ is 3-colorable and $l \neq m$ for $l, m \in \{1,2,3\}$.

We now proceed with considering the operations of $\mathcal{E}$.

$f = \mathbf{0}$. The empty graph is 3-colorable; thus, we take $B_{f,p_0} = \text{true}$.

$f = \mathbf{2}$. The graph with one single edge is 3-colorable; thus, we take $B_{f,p_{lm}} = \text{true}$ iff $l \neq m$.

$f = \text{id}_n$. Obviously, $p(f(G)) \Leftrightarrow p(G)$; thus, $B_{f,p} = x_p^1$ suffices.

$f = \oplus_{m,n}$. Let $i \in \{1,2,3\}^{m+n}$, and let $i = i_1 \cdot i_2$, where $i_1 \in \{1,2,3\}^m$ and $i_2 \in \{1,2,3\}^n$. A $(m+n)$-graph $G_1 \oplus_{m,n} G_2$ has a 3-coloring which agrees with i if and only if G_j has a 3-coloring which agrees with i_j, for $j = 1, 2$. Thus, $p_i(G_1 \oplus_{m,n} G_2) \Leftrightarrow p_{i_1}(G_1) \wedge p_{i_2}(G_2)$ and we can take $B_{f,p_i} = x_{p_{i_1}}^1 \wedge x_{p_{i_2}}^2$.

$f = \theta_{n,\delta}$. Let $H = f(G)$, and let $i \in \{1,2,3\}^n$. We say that *i respects δ*, if $i_j = i_m$ for every pair $(j, m) \in \delta$. If i does not respect δ, then no 3-coloring of H can agree with i. If i does respect δ, then H has a 3-coloring which agrees with i if and only if G has a 3-coloring which agrees with i, thus we can take $B_{f,p} = x_p^1$.

$f = \sigma_{n,m,a}$. If c is a 3-coloring of G which agrees with $i \in \{1,2,3\}^m$, and if $l \in \{1,2,3\}^n$ then c agrees with l if and only if for all $j = 1, \dots, m$, we have that $i_j = c((\text{ext}_G)_j) = c((\text{ext}_H)_{a(j)}) = l_{a(j)}$. Thus, if we let $M_i = \{l \in \{1,2,3\}^n \,|\, l_{a(j)} = i_j$ for $j = 1, \dots, m\}$, we get $p_i(G) \Leftrightarrow \bigvee_{l \in M_i} p_l(H)$, and we let $B_{f,p_i} = \bigvee_{l \in M_i} x_{p_l}^1$.

Finally, $\mathscr{P}$ is even effectively locally finite and $\mathscr{E}$-inductive. For, with every predicate p_i represented by i, the mappings $k \mapsto P_k$ and $(p_i, G) \mapsto p_i(G)$ are clearly computable. Furthermore, the constructions given above to show $\mathscr{E}$-inductivity are easily seen to be constructive, making the mapping $(f, p_i) \mapsto B_{f,p_i}$ computable.

The following lemma will be useful in interrelating $\mathscr{F}'$-inductivity and $\mathscr{C}$-compatibility.

Lemma 5.6 *Let $\mathscr{F}' \subseteq \mathscr{F}$, and let $\mathscr{P}$ be an effectively locally finite and inductive family of predicates. If $\mathscr{P}$ is $\mathscr{F}'$-inductive, then it is also* closure$(\mathscr{F}')$*-inductive.*

Proof. Let $g \in \text{closure}(\mathscr{F}')$, $g: \mathscr{G}_{k_1} \times \cdots \times \mathscr{G}_{k_r} \to \mathscr{G}_k$. If $g \in \mathscr{F}'$ then $B_{g,p}$ can be found effectively, for every $p \in P_k$, by the hypothesis. Now, assume that $g = f(h_1, \ldots, h_s)$, for some operations $f: \mathscr{G}_{l_1} \times \cdots \times \mathscr{G}_{l_s} \to \mathscr{G}_k$, and $h_i: \mathscr{G}_{k_{i1}} \times \cdots \times \mathscr{G}_{k_{ir_i}} \to \mathscr{G}_{l_i}$, $i = 1, \ldots, s$. By the induction hypothesis there are formulae $B_{f,p}$, for every $p \in P_k$, and $B_{h_i,q}$, for $i = 1, \ldots, s$, and every $q \in P_{l_i}$. Then, using the decomposition of g and the induction hypothesis, we get

$$\begin{aligned}
&p(g(G_1, \ldots, G_r)) \\
&\quad \Leftrightarrow\ p(f(h_1(G_{k_{11}}, \ldots, G_{k_{1r_1}}), \ldots, h_s(G_{k_{s1}}, \ldots, G_{k_{sr_s}})) \\
&\quad \Leftrightarrow\ B_{f,p}[(x_q^i \mid q(h_i(G_{k_{i1}}, \ldots, G_{k_{ir_i}})))_{q \in P_{l_i}}^{i=1,\ldots,s}] \\
&\quad \Leftrightarrow\ B_{f,p}[(x_q^i \mid B_{h_i,q}[(x_w^j \mid w(G_{k_{ij}}))_{w \in P_{k_{ij}}}^{j=1,\ldots,r_i}])_{q \in P_{l_i}}^{i=1,\ldots,s}].
\end{aligned}$$

Thus, $B_{g,p} = B_{f,p}[(x_q^i \mid B_{h_i,q})_{q \in P_{l_i}}^{i=1,\ldots,s}]$ is the required decomposition of p for g. $\square$

We proceed now by interrelating $\mathscr{F}'$-inductivity and $\mathscr{C}$-compatibility.

Definition 5.7 (*Associated class of grammars*). Let $\mathscr{F}' \subseteq \mathscr{F}$. The set $\mathscr{C}(\mathscr{F}')$ of hyperedge-replacement grammars *associated with* $\mathscr{F}'$ is defined by

$$\mathscr{C}(\mathscr{F}') = \{\text{HRG} \in \mathscr{HRG} \mid f_{\text{rhs}(p)} \in \mathscr{F}' \text{ for every production } p \text{ of HRG}\}.$$

Theorem 5.8 (Inductivity implies compatibility). *Let $\mathscr{F}' \subseteq \mathscr{F}$. If $(P_k)_{k \in \mathbb{N}}$ is an effectively locally finite and $\mathscr{F}'$-inductive family of predicates on $\mathscr{G}$ then every predicate p in $\bigcup_{k \in \mathbb{N}} P_k$ is $\mathscr{C}(\mathscr{F}')$-compatible.*

Proof. For every $k \in \mathbb{N}$, let $I_k = \{U \mid U \subseteq P_k\}$, and let PROP be defined by

$$\text{PROP}(G, U) \Leftrightarrow \bigwedge_{p \in U} p(G) \wedge \bigwedge_{p \in P_k - U} \neg p(G).$$

From the definition of an effectively locally finite and inductive family of predicates, it is clear that I_k is finite for every type k, and that I_k can be effectively computed from k. Furthermore, PROP is decidable, and there is a computable function which assigns to every mapping $f:\mathscr{G}_{k_1}\times\cdots\times\mathscr{G}_{k_r}\rightarrow\mathscr{G}_k$ in $\mathscr{F}'$, and every $p\in P_k$ a Boolean formula $B_{f,p}$ with variables $(x_q^i)_{q\in P_{k_i}}^{i=1,\ldots,r}$ such that for all $(G_1,\ldots,G_r)\in\mathscr{G}_{k_1}\times\cdots\times\mathscr{G}_{k_r}$

$$p(f(G_1,\ldots,G_r))\Leftrightarrow B_{f,p}[(x_q^i\,|\,q(G_i))_{q\in P_{k_i}}^{i=1,\ldots,r}].$$

Let $H\in\mathscr{G}_k$ be the right-hand side of a production of some HRG$=(N,P,S)\in\mathscr{C}(\mathscr{F}')$, let $Y_H=\{y_1,\ldots,y_r\}$, and type$(y_i)=k_i$, for $i=1,\ldots,r$. Let ass: $Y_H\rightarrow I$ be such that for $i=1,\ldots,r$ ass$(y_i)\subseteq P_{k_i}$, and let $U\subseteq P_k$. For such triples (H,ass,U), let the predicate PROP$'$ be defined by

$$\text{PROP}'(H,\text{ass},U)\Leftrightarrow\bigwedge_{p\in U}B_{f_H,p}[(x_q^i\,|\,v_q^i)_{q\in P_{k_i}}^{i=1,\ldots,r}]$$

$$\wedge\bigwedge_{p\in P_k-U}\neg B_{f_H,p}[(x_q^i\,|\,v_q^i)_{q\in P_{k_i}}^{i=1,\ldots,r}],$$

where f_H is as in Lemma 5.3(2) and

$$v_q^i=\begin{cases}1 & \text{if } q\in\text{ass}(y_i)\\ 0 & \text{if } q\notin\text{ass}(y_i).\end{cases}$$

Then PROP$'$ is decidable since the $B_{f_H,p}$ can be effectively constructed and decided. Furthermore, for every graph G such that $H\overset{*}{\underset{P}{\Rightarrow}}G$ there is a well-typed mapping repl: $Y_H\rightarrow\mathscr{G}$ such that $G=\text{REPLACE}(H,\text{repl})=f_H(\text{repl}(y_1),\ldots,\text{repl}(y_r))$. Thus,

$$\text{PROP}(G,U)\Leftrightarrow\bigwedge_{p\in U}p(G)\wedge\bigwedge_{p\in P_k-U}\neg p(G)$$

$$\Leftrightarrow\bigwedge_{p\in U}B_{f_H,p}[(x_q^i\,|\,q(\text{repl}(y_i)))_{q\in P_{k_i}}^{i=1,\ldots,r}]$$

$$\wedge\bigwedge_{p\in P_{k_i}-U}\neg B_{f_H,p}[(x_q^i\,|\,q(\text{repl}(y_i)))_{q\in P_{k_i}}^{i=1,\ldots,r}].$$

If we choose ass$(y_i)=\{q\in P_{k_i}\,|\,q(\text{repl}(y_i))\}$, then

$$\text{PROP}(G,U)\Leftrightarrow\text{PROP}'(H,\text{ass},U)\wedge\bigwedge_{i=1}^{r}\text{PROP}(\text{repl}(y_i),\text{ass}(y_i))$$

and, therefore,

$$\text{PROP}(G,U)\Rightarrow\exists(\text{ass}: Y_H\rightarrow I)\text{: PROP}'(H,\text{ass},U)$$

$$\wedge\bigwedge_{i=1}^{r}\text{PROP}(\text{repl}(y_i),\text{ass}(y_i)).$$

If, on the other hand, there is a mapping ass: $Y_H\rightarrow I$ such that

$$\text{PROP}'(H,\text{ass},U)\wedge\bigwedge_{i=1}^{r}\text{PROP}(\text{repl}(y_i),\text{ass}(y_i)),$$

then, from the definition of PROP we get for $i=1,\dots,r$

$$\mathrm{PROP}(\mathrm{repl}(y_i),\mathrm{ass}(y_i)) \Leftrightarrow \bigwedge_{q\in\mathrm{ass}(y_i)} q(\mathrm{repl}(y_i)) \wedge \bigwedge_{q\notin\mathrm{ass}(y_i)} \neg q(\mathrm{repl}(y_i))$$

and, therefore,

$$q(\mathrm{repl}(y_i))=\begin{cases}1 & \text{if } q\in\mathrm{ass}(y_i)\\ 0 & \text{if } q\notin\mathrm{ass}(y_i).\end{cases}$$

Then, for every $p\in P_k$ we have

$$B_{f_H,p}[(x_q^i \mid q(\mathrm{repl}(y_i)))_{q\in P_{k_i}}^{i=1,\dots,r}] \Leftrightarrow B_{f_H,p}[(x_q^i \mid v_q^i)_{q\in P_{k_i}}^{i=1,\dots,r}],$$

where

$$v_q^i=\begin{cases}1 & \text{if } q\in\mathrm{ass}(y_i)\\ 0 & \text{if } q\notin\mathrm{ass}(y_i).\end{cases}$$

Since $\mathrm{PROP}'(H,\mathrm{ass},U)$ is true, by the definition of PROP′, the formula

$$\bigwedge_{p\in U} B_{f_H,p}[(x_q^i \mid q(\mathrm{repl}(y_i)))_{q\in\mathrm{ass}(y_i)}^{i=1,\dots,r}]$$

$$\wedge \bigwedge_{p\in P_{k_i}-U} \neg B_{f_H,p}[(x_q^i \mid q(\mathrm{repl}(y_i)))_{q\in\mathrm{ass}(y_i)}^{i=1,\dots,r}]$$

holds true. Since $f_H\in\mathscr{F}'$, and $\mathscr{P}$ is $\mathscr{F}'$-inductive this is equivalent to

$$\bigwedge_{p\in U} p(G) \wedge \bigwedge_{p\in P_k-U} \neg p(G)$$

and this to $\mathrm{PROP}(G,U)$. This shows that PROP is $\mathscr{C}(\mathscr{F}')$-compatible and, therefore, $\mathrm{PROP}(-,U)$ is compatible, for every $U\subseteq P_k$, $k\in\mathbb{N}$. Let $p\in P_k$. Then for all $G\in\mathscr{G}_k$,

$$p(G)\Leftrightarrow \bigvee_{U:\,p\in U}\left[\bigwedge_{q\in U} q(G) \wedge \bigwedge_{q\in P_k-U} \neg q(G)\right]$$

$$\Leftrightarrow \bigvee_{U:\,p\in U} \mathrm{PROP}(G,U),$$

where $\bigvee_{U:\,p\in U}$ means that the disjunction is taken over all those sets U that contain p. Since compatibility is closed under Boolean operations (cf. Fact 3.4) it follows that p is $\mathscr{C}(\mathscr{F}')$-compatible. □

Corollary 5.9. *If $(P_k)_{k\in\mathbb{N}}$ is effectively locally finite and $\mathscr{E}$-inductive, then every predicate $p\in\bigcup_{k\in\mathbb{N}} P_k$ is $\mathscr{HRG}$-compatible.*

Proof. By Lemma 5.6, $(P_k)_{k\in\mathbb{N}}$ is $\mathscr{F}$-inductive, and Definition 5.7 implies that $\mathscr{C}(\mathscr{F})=\mathscr{HRG}$. Thus, the result follows from Theorem 5.8. □

Similarly, compatibility implies inductivity. To show this, we need the following notion.

Definition 5.10 (*Associated set of operations*). (1) Let $f: \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r} \to \mathcal{G}_k$ be an operation in $\mathcal{F}$ and let $Y_{H_f} = \{y_1, \ldots, y_r\}$, where H_f is the decorated graph corresponding to f according to Lemma 5.3. A hyperedge-replacement grammar HRG $= (N, P, S)$ is said to *emulate* f for $(G_1, \ldots, G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$, if there is a rule $(A, H_f) \in P$, for some $A \in N$, and if $(A_i, k_i)^\bullet \overset{*}{\underset{P}{\Rightarrow}} G_i$, with $A_i = \mathrm{lab}_{H_f}(y_i)$, for $i = 1, \ldots, r$.

(2) A set $\mathcal{C}$ of hyperedge-replacement grammars is said to *emulate* the operation f if for every $(G_1, \ldots, G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$ there is a grammar HRG$\in\mathcal{C}$ which emulates f for $(G_1, \ldots, G_r)$.

(3) Let $\mathcal{C} \subseteq \mathcal{HRG}$ be a class of hyperedge-replacement grammars. Then the *operation set* $\mathcal{F}(\mathcal{C})$ *associated with* $\mathcal{C}$ is defined by

$$\mathcal{F}(\mathcal{C}) = \{ f \in \mathcal{F} \mid \mathcal{C} \text{ emulates } f \}.$$

Theorem 5.11 (Compatibility implies inductivity). *Let* PROP *be* ($\mathcal{C}$, PROP′)*-compatible over the index set* $I = (I_k)_{k \in \mathbb{N}}$. *Moreover, let, for every* $k \in \mathbb{N}$, $P_k = \{\mathrm{PROP}(-, i) \mid i \in I_k\}$. *Then* $(P_k)_{k \in \mathbb{N}}$ *is effectively locally finite and* $\mathcal{F}(\mathcal{C})$*-inductive.*

Proof. By definition of compatibility, every P_k is finite, and the mapping $(i, G) \mapsto \mathrm{PROP}(G, i)$ is computable. Let $f \in \mathcal{C}(\mathcal{F})$, $f: \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r} \to \mathcal{G}_k$, let $(G_1, \ldots, G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$, and let $p \in P_k, p = \mathrm{PROP}(-, j)$. There is a grammar HRG $= (N, P, S)$ in $\mathcal{C}$ which emulates f for $(G_1, \ldots, G_r)$ and, thus, $H_f \overset{*}{\underset{P}{\Rightarrow}} f(G_1, \ldots, G_r)$. By definition of compatibility it follows that

$$\begin{aligned} &p(f(G_1, \ldots, G_r)) \\ &\Leftrightarrow \mathrm{PROP}(f(G_1, \ldots, G_r), j) \\ &\Leftrightarrow \exists (\mathrm{ass}: Y_{H_f} \to I)\text{: } \mathrm{PROP}'(H_f, \mathrm{ass}, j) \wedge \bigwedge_{i=1}^{r} \mathrm{PROP}(G_i, \mathrm{ass}(y_i)). \end{aligned}$$

Since $\mathrm{ass}(y_i) \in I_{k_i}$, there are only finitely many candidates for ass, and we can write

$$p(f(G_1, \ldots, G_r)) \Leftrightarrow \bigvee_{\mathrm{ass} \in M} \bigwedge_{i=1}^{r} \mathrm{PROP}(G_i, \mathrm{ass}, (y_i)),$$

where M is the set of all those ass for which $\mathrm{PROP}'(H_f, \mathrm{ass}, j)$ holds. Letting $P' \subseteq P_{k_1} \times \cdots \times P_{k_r}$ be the finite set of tuples

$$P' = \{(p_1, \ldots, p_r) \mid \text{there is an ass} \in M\text{: } p_i = \mathrm{PROP}(-, \mathrm{ass}(y_i)), i = 1, \ldots, r\},$$

the last equivalence can be rewritten as

$$p(f(G_1, \ldots, G_r)) \Leftrightarrow \bigvee_{(p_1, \ldots, p_r) \in P'} \bigwedge_{i=1}^{r} p_i(G_i).$$

We define the Boolean formula $B_{f,p}$ by

$$B_{f,p}((x_q^i)_{q\in P_{k_i}}^{i=1,\ldots,r}) = \left(\bigvee_{(p_1,\ldots,p_r)\in P'} \bigwedge_{i=1}^{r} x_{p_i}^i \right) \vee \left(\bigwedge_{i=1}^{r} \bigwedge_{q\in P_{k_i}} x_q^i \wedge \neg x_q^i \right).$$

Note that the second part of $B_{f,p}$ is unsatisfiable, and is added only in order to make all variables x_q^i appear in $B_{f,p}$, as is formally required in Definition 5.4. This gives us

$$p(f(G_1,\ldots,G_r)) \Leftrightarrow B_{f,p}[(x_q^i \mid q(G_i))_{q\in P_{k_i}}^{i=1,\ldots,r}].$$

Now $B_{f,p}$ does not depend on $G_1,\ldots,G_r$; in fact, given f and p it can be constructed as follows.

(1) Determine H_f and j such that $p = \text{PROP}(-,j)$.

(2) Determine M, the set of those ass for which $\text{PROP}'(H_f, \text{ass}, j)$ holds true.

(3) For each $\text{ass}\in M$, find the tuple

$$(p_1,\ldots,p_r) = (\text{PROP}(-,\text{ass}(y_1)),\ldots,\text{PROP}(-,\text{ass}(y_r))).$$

(4) For each such tuple form the conjunct $\bigwedge_{i=1}^{r} x_{p_i}^i$.

(5) $B_{f,p}$ is the disjunction of all the conjuncts formed in (4), and the constant

$$\left(\bigwedge_{i=1}^{r} \bigwedge_{q\in P_{k_i}} x_q^i \wedge \neg x_q^i \right). \quad \square$$

Corollary 5.12. *If* PROP *is* ($\mathcal{HRG}$, PROP′)*-compatible, then the family* $\mathcal{P} = (P_k)_{k\in\mathbb{N}}$, *with* $P_k = \{\text{PROP}(-,i) \mid i\in I_k\}$, *is effectively locally finite and* $\mathcal{F}$*-inductive.*

Proof. For $f: \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r} \to \mathcal{G}_k$ and $(G_1,\ldots,G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$, the hyperedge-replacement grammar $\text{HRG}(f, G_1,\ldots,G_r) = (N, P, S)$ with starting rule (S, H_f) and the terminating rules $(\text{lab}_{H_f}(y_i), G_i)$, emulates f for $(G_1,\ldots,G_r)$. Therefore, $\mathcal{HRG}$ emulates f, for all $f\in\mathcal{F}$, i.e., $\mathcal{F}(\mathcal{HRG}) = \mathcal{F}$. By Theorem 5.10, $\mathcal{P}$ is effectively locally finite and $\mathcal{F}$-inductive. $\square$

6. Discussion

From the results in this paper it is clear that compatibility, finiteness, and inductivity are nearly identical. However, since these notions have been developed in different contexts, there are still some differences. In particular, it should be noted that inductivity and compatibility are defined relative to a class of operations, or grammars, respectively, whereas finiteness is a global notion. Therefore, it seems conceivable that there are properties which are compatible with respect to restricted grammar classes, but are not finite. Similarly, $\mathcal{F}'$-inductivity, for a restricted operation set $\mathcal{F}'$ does not necessarily imply finiteness.

In comparing inductivity and compatibility, we note that $\mathcal{F}'$-inductivity seems to be a coarser notion than $\mathcal{C}$-compatibility, since the recursive condition

$$p(f(G_1,\dots,G_r)) \Leftrightarrow B_{f,p}(x_q^i \mid q(G_i))$$

has to hold for *all* $(G_1,\dots,G_r) \in \mathcal{G}_{k_1} \times \cdots \times \mathcal{G}_{k_r}$, whereas, in the case of $\mathcal{C}$-compatibility, the corresponding condition

$$\mathrm{PROP}(R[y_1/G_1,\dots,y_r/G_r], i) \Leftrightarrow \exists \mathrm{ass}: \mathrm{PROP}'(R, \mathrm{ass}, i) \wedge \bigwedge_{i=1}^{r} \mathrm{PROP}(G_i, \mathrm{ass}(y_i))$$

is only required to hold if $Y_R = \{y_1,\dots,y_r\}$ and for all y_i, $(\mathrm{lab}_R(y_i), \mathrm{type}_R(y_i))^\bullet \overset{*}{\Rightarrow} G_i$. So, again, there may be families of graph properties which are $\mathcal{C}$-compatible for some $\mathcal{C}$, but not $\mathcal{F}'$-inductive for any $\mathcal{F}' \neq \emptyset$.

The subtle difference in the formulations of Theorems 4.10 and 4.11, namely that 4.10 requires computability of $k \mapsto |M_\Pi(k)|$, whereas 4.11 only guarantees that there exists a computable function f with $|M_\Pi(k)| \leqslant f(k)$ for $k \in \mathbb{N}$, does not seem to be avoidable with the given definitions. One possible solution would be to make PROP depend, not only on G and i, but also on the grammar. This would, however, lead to a fairly cumbersome definition.

The three notions that we have compared in this paper have been developed in connection with the decidability of questions of the following form for hyperedge-replacement grammars.

– Does some member of the generated language satisfy Π?
– Do infinitely many members of the generated language satisfy Π?
– Do all members of the generated language satisfy Π?

It seems that all known decidability results have been (or could be) proved by showing the finiteness (compatibility, inductivity) of Π. For example, in [4], Courcelle proves that the questions above are decidable, if Π is definable in the language of so-called counting monadic second-order logic (CMSO), by showing that from a CMSO-formula which defines Π, we can construct an inductive family of predicates.

There are, however, limitations to this method. As seen in Example 4.7, regularity of a graph is not finite, hence neither compatible nor inductive.[11] Nevertheless, whether a hyperedge-replacement grammar generates some (infinitely many, only) regular graphs can be decided, by deciding the corresponding problems for k-regularity, where k ranges over a finite set of values which can be determined from the grammar.

Another apparent limitation of the method lies in the fact that finiteness (compatibility, inductivity) is a property of graph properties. Thus, it does not seem possible to tackle the decidability of questions which are not directly involved with graph properties, such as, e.g.,

– Does the grammar generate graphs with arbitrarily high degree?

[11] In fact, it can be shown that regularity cannot be defined by an emso-formula, as defined in [1] (although nonregularity can).

In order to deal with problems of this type, both the notion of compatibility and the approach via monadic-second-order logic, have been further developed: in [11], the decidability of certain boundedness problems is investigated, and in [5], the concept of a monadic second-order evaluation is developed.

References

[1] S. Arnborg, J. Lagergren and D. Seese, Problems easy for tree-decomposable graphs, *J. Algorithms* **12** (1991) 308–340.

[2] M. Bauderon and B. Courcelle, Graph expressions and graph rewriting, *Math. Systems Theory* **20** (1987) 83–127.

[3] B. Courcelle, On context-free sets of graphs and their monadic second-order theory, in: H. Ehrig et al., eds., *Graph-Grammars and Their Application to Computer Science*, Lecture Notes in Computer Science, Vol. 291 (Springer, Berlin, 1987) 237–247.

[4] B. Courcelle, The monadic second-order logic of graphs I: recognizable sets of finite graphs, *Inform. and Comput.* **85** (1990) 12–75.

[5] B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, *Theoret. Comput. Sci.* **109** (1993) 49–82.

[6] J. Feder, Plex languages, *Inform. Sci.* **3** (1971) 225–241.

[7] A. Habel, Hyperedge replacement: grammars and languages, Ph.D. Thesis, Bremen 1989; to appear in Lecture Notes in Computer Science.

[8] A. Habel and H.-J. Kreowski, Some structural aspects of hypergraph languages generated by hyperedge replacement, in: F.J. Brandenburg et al., eds., *Proc. STACS* 87, Lecture Notes in Computer Science, Vol. 247 (Springer, Berlin, 1987) 207–219.

[9] A. Habel and H.-J. Kreowski, Filtering hyperedge-replacement languages through compatible properties, in: M. Nagl, ed., *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, Vol. 411 (Springer, Berlin, 1990) 107–120.

[10] A. Habel, H.-J. Kreowski and W. Vogler, Metatheorems for decision problems on hyperedge replacement graph languages, *Acta Inform.* **26** (1989) 657–677.

[11] A. Habel, H.-J. Kreowski and W. Vogler, Decidable boundedness problems for sets of graphs generated by hyperedge replacement, *Theoret. Comput. Sci.* **89** (1991) 33–62.

[12] T. Lengauer and E. Wanke, Efficient analysis of graph properties on context-free graph languages, in T. Lepistö and A. Salomaa, eds., *Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 379–393; revised version as Tech. Report 45, Paderborn, 1989.

[13] E. Wanke, Algorithmen und Komplexitätsanalyse für die Verarbeitung hierarchisch definierter Graphen und hierarchisch definierter Graphfamilien, Ph.D. Thesis, Paderborn, 1989.