Theoretical Computer Science 89 (1991) 33–62
Elsevier

Decidable boundedness problems for sets of graphs generated by hyperedge-replacement

Annegret Habel, Hans-Jörg Kreowski

Universität Bremen, Fachbereich Mathematik und Informatik, Postfach 330440, W-2800 Bremen 33, Germany

Walter Vogler

Technische Universität München, Institut für Informatik, Postfach 202420, W-8000 München 2, Germany

Abstract

Habel, A., H.-J. Kreowski and W. Vogler, Decidable boundedness problems for sets of graphs generated by hyperedge-replacement, Theoretical Computer Science 89 (1991) 33–62.

Consider a class $\mathscr{C}$ of hyperedge-replacement graph grammars and a numeric function on graphs like the number of edges, the degree (i.e., the maximum of the degrees of all nodes of a graph), the number of simple paths, the size of a maximum set of independent nodes, etc. Each such function induces a boundedness problem for the class $\mathscr{C}$: Given a grammar HRG in $\mathscr{C}$, are the function values of all graphs in the language $L(HRG)$, generated by HRG, bounded by an integer or not? We show that the boundedness problem is decidable if the corresponding function is compatible with the derivation process of the grammars in $\mathscr{C}$ and if it is composed of maxima, sums, and products in a certain way. This decidability result applies particularly to the examples listed above.

Various significant sets of graphs such as the set of series-parallel graphs, the set of (maximum) outerplanar graphs, the set of k-trees, and the set of graphs of cyclic bandwidth $\leq k$ can be generated by hyperedge-replacement graph grammars. Hence, the study in this paper is not only attributed to the area of graph grammars but may also interest those who investigate graph-theoretic properties of particular sets of graphs.

1. Introduction

Context-free graph grammars (like edge- and hyperedge-replacement grammars as investigated, e.g., by Bauderon and Courcelle [2] or in [8, 10] or like boundary NLC grammars as introduced by Rozenberg and Welzl [17]) have been studied intensively for some time now because of—at least—two reasons.

(1) Although their generative power is intentionally restricted, they cover many graph languages interesting from the point of view of applications as well as of graph theory (for example, certain types of flow diagrams, PASCAL syntax diagrams,

certain types of Petri nets, graph representations of functional expressions, series-parallel graphs, outerplanar graphs, k-trees, graphs with cyclic bandwidth $\leq k$).

(2) Of all classes of graph grammars discussed in the literature, they seem to render the most attractive theory with a variety of results on structure, decidability and complexity (see, e.g., Arnborg, Lagergren and Seese [1], Bauderon and Courcelle [2, 3], Della Vigna and Ghezzi [4], Lautemann [15], Lengauer and Wanke [16], Rozenberg and Welzl [17, 18], Slisenko [19], and [14, 7, 8, 10, 11, 5, 6]).

In particular, Courcelle [3], Arnborg et al. [1], Lengauer and Wanke [16], and [11] present syntactic and semantic conditions such that, for a graph property P satisfying the conditions, the following hold for all context-free graph grammars of the types considered in the respective papers:

(1) It is decidable whether (or not) some graph with property P is generated.

(2) It is decidable whether (or not) all generated graphs have property P.

(3) It is decidable in linear time whether (or not) a generated graph represented by a derivation (or something equivalent) has property P.

The results apply to properties such as connectivity, planarity, k-colorability, existence of Hamiltonian and Eulerian paths and cycles.

Based on the framework of hyperedge-replacement graph grammars, we continue this line of consideration in this paper. We are going to investigate the decidability of a different type of problems concerning functions on graphs and above all numeric quantities like the numbers of nodes, edges and paths, the node degree, maximum and minimum lengths of paths and cycles, etc. The kind of question we ask for a class of grammars may be called *boundedness problem.* It is as follows:

(4) Is it decidable whether (or not), concerning a particular quantity, the values of all graphs generated by a grammar are bounded?

For example, we want to know whether the node degree or the number of paths grow beyond any bound within a graph language. In the main result, we show that such a boundedness problem is decidable for a class of hyperedge-replacement grammars if the corresponding quantity function is built up by maxima, sums and products and if the function is compatible with the derivation process of the given grammars. Examples of this kind are the bounded-node-degree problem, the bounded-maximum-path-length problem, the bounded-maximum-number-of-paths problem and others. It should be mentioned here that the only result of the same nature occurring in the literature is the decidability of the bounded-degree problem for NLC grammars (see [13]).

Various significant sets of graphs such as the set of series-parallel graphs, the set of (maximum) outerplanar graphs, the set of k-trees, and the set of graphs of cyclic bandwidth $\leq k$ can be generated by hyperedge-replacement graph grammars. Hence, the study in this paper is not only attributed to the area of graph grammars but may also interest those who investigate graph-theoretic properties of particular sets of graphs.

The paper is organized in the following way. Sections 2 and 3 comprise the preliminaries on (hyper)graphs and hyperedge-replacement grammars as needed.

In Section 4, we discuss several examples of numeric functions which are compatible with the derivation process of our grammars in a certain way. In Section 5, we introduce the general notion of compatible functions and relate them with our earlier notion of compatible predicates [11]. Finally, we show in the main result in Section 6 that the boundedness problem corresponding to a numeric function is decidable if the function is pointwise defined as the maximum of sums and products and if it is compatible.

While the general results work for arbitrary classes of hyperedge-replacement grammars, we have to admit that most of our examples are formulated for the class of edge-replacement grammars. But we are confident that all of them can be adapted to more general classes of hyperedge-replacement grammars.

A short version of this paper without proofs is given in [12].

2. Preliminaries

This section provides the basic notions on graphs and hypergraphs as far as needed in the paper. The key construction is the replacement of some hyperedges of a hypergraph by hypergraphs yielding an expanded hypergraph. In our approach, a hyperedge is an atomic item with an ordered set of incoming tentacles and an ordered set of outgoing tentacles where, intuitively, each tentacle grips at a node through the source and target functions. Correspondingly, a hypergraph is equipped with two sequences of distinguished nodes so that it is enabled to replace a hyperedge.

General assumption 2.1. Throughout the paper, let C be an arbitrary, but fixed alphabet, called a set of *labels* (or *colors*).

Definition 2.2 (hypergraphs). (1) A *hypergraph* over C is a system (V, E, s, t, l) where V is a finite set of *nodes* (or *vertices*), E is a finite set of *hyperedges*, $s: E \to V^*$ and $t: E \to V^*$[1] are two mappings assigning a sequence of *sources* $s(e)$ and a sequence of *targets* $t(e)$ to each $e \in E$, and $l: E \to C$ is a mapping *labeling* each hyperedge.

(2) A hyperedge $e \in E$ of a hypergraph (V, E, s, t, l) is called an (m, n)-*edge* for some $m, n \in \mathbb{N}$ if $|s(e)| = m$ and $|t(e)| = n$.[2] The pair (m, n) is the *type* of e, denoted by $type(e)$. e is said to be *well-formed* if its sources and targets are pairwise distinct.

(3) A *multi-pointed hypergraph* over C is a system $H = (V, E, s, t, l, begin, end)$ where the first five components define a hypergraph over C and $begin, end \in V^*$. Components of H are denoted by V_H, E_H, s_H, t_H, l_H, $begin_H$, end_H, respectively. The set of all multi-pointed hypergraphs over C is denoted by $\mathcal{H}_C$.

(4) $H \in \mathcal{H}_C$ is said to be an (m, n)-*hypergraph* for some $m, n \in \mathbb{N}$ if $|begin_H| = m$ and $|end_H| = n$. The pair (m, n) is the *type* of H, denoted by $type(H)$. H is said to

[1] For a set A, A^* denotes the set of all words over A, including the empty word λ.

[2] For a word $w \in A^*$, $|w|$ denotes its length.

be *well-formed* if all hyperedges are well-formed and the begin-nodes and end-nodes of H are pairwise distinct.

(5) Let $H \in \mathcal{H}_C$, $begin_H = begin_1 \ldots begin_m$ and $end_H = end_1 \ldots end_n$ with $begin_i$, $end_j \in V_H$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Then $EXT_H = \{begin_i \mid i = 1, \ldots, m\} \cup \{end_j \mid j = 1, \ldots, n\}$ denotes the set of *external nodes* of H. Moreover, $INT_H = V_H - EXT_H$ denotes the set of *internal* nodes of H.

Remarks. (1) There is a 1-1-correspondence between hypergraphs and $(0, 0)$-hypergraphs so that hypergraphs may be seen as special cases of multi-pointed hypergraphs.

(2) An (m, n)-hypergraph over C with $(1, 1)$-edges only is said to be an (m, n)-*graph*. The set of all $(1, 1)$-graphs over C is denoted by $\mathcal{G}_C$.

Definition 2.3 (special hypergraphs). (1) A multi-pointed hypergraph H is said to be a *singleton* if $|E_H| = 1$ and $|V_H - EXT_H| = 0$. $e(H)$ refers to the only hyperedge of H and $l(H)$ refers to its label.

(2) A singleton H is said to be a *handle* if $s_H(e) = begin_H$ and $t_H(e) = end_H$. If $l_H(e) = A$ and $type(e) = (m, n)$ for some $m, n \in \mathbb{N}$, then H is called an (m, n)-*handle* induced by A.

Remark. Given $H \in \mathcal{H}_C$, each hyperedge $e \in E_H$ induces a handle e^* by restricting the mappings s_H, t_H, and l_H to the set $\{e\}$, restricting the set of nodes to those ones occurring in $s_H(e)$ and $t_H(e)$, and choosing $begin_{e^*} = s_H(e)$ and $end_{e^*} = t_H(e)$.

Definition 2.4 (subhypergraphs and isomorphic hypergraphs). (1) Let H, $H' \in \mathcal{H}_C$. Then H is called a (*weak*) *subhypergraph* of H', denoted by $H \subseteq H'$, if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, and $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$, $l_H(e) = l_{H'}(e)$ for all $e \in E_H$. (Note that nothing is assumed on the relation of the distinguished nodes.)

(2) Let H, $H' \in \mathcal{H}_C$ and $i_V : V_H \to V_{H'}$, $i_E : E_H \to E_{H'}$ be bijective mappings. Then $i = (i_V, i_E) : H \to H'$ is called an *isomorphism* from H to H' if $i_V^*(s_H(e)) = s_{H'}(i_E(e))$, $i_V^*(t_H(e)) = t_{H'}(i_E(e))$, $l_H(e)) = l_{H'}(i_E(e))$ for all $e \in E_H$ as well as $i_V^*(begin_H) = begin_{H'}$, $i_V^*(end_H) = end_{H'}$.[3] H and H' are said to be *isomorphic*, denoted by $H \cong H'$, if there is an isomorphism from H to H'.

Now we are ready to introduce how hypergraphs may substitute hyperedges. An (m, n)-edge can be replaced by an (m, n)-hypergraph in two steps.

(1) Remove the hyperedge.

(2) Add the hypergraph except the external nodes and hand over each tentacle of a hyperedge (of the replacing hypergraph) which grips to an external node to the corresponding source or target node of the replaced hyperedge.

[3] For a mapping $f : A \to B$, the free symbolwise extension $f^* : A^* \to B^*$ is defined by $f^*(a_1 \ldots a_k) = f(a_1) \ldots f(a_k)$ for all $k \in \mathbb{N}$ and $a_i \in A$ $(i = 1, \ldots, k)$.

Moreover, an arbitrary number of hyperedges can be replaced simultaneously in this way.

Definition 2.5 (hyperedge replacement). Let $H \in \mathscr{H}_C$ be a multi-pointed hypergraph, $B \subseteq E_H$, and $repl: B \to \mathscr{H}_C$ a mapping with $type(repl(b)) = type(b)$ for all $b \in B$. Then the *replacement* of B in H through *repl* yields the multi-pointed hypergraph X given by

- $V_X = V_H + \sum_{b \in B}(V_{repl(b)} - EXT_{repl(b)})$,[4]
- $E_X = (E_H - B) + \sum_{b \in B} E_{repl(b)}$,
- each hyperedge of $E_H - B$ keeps its sources and targets,
- each hyperedge of $E_{repl(b)}$ (for all $b \in B$) keeps its internal sources and targets and the external ones are handed over to the corresponding sources and targets of b, i.e., $s_X(e) = h^*(s_{repl(b)}(e))$ and $t_X(e) = h^*(t_{repl(b)}(e))$ for all $b \in B$ and $e \in E_{repl(b)}$ where $h: V_{repl(b)} \to V_X$ is defined by $h(v) = v$ for $v \in V_{repl(b)} - EXT_{repl(b)}$, $h(b_i) = s_i$ $(i = 1, \ldots, m)$ for $begin_{repl(b)} = b_1 \ldots b_m$ and $s_H(b) = s_1 \ldots s_m$, $h(e_j) = t_j$ $(j = 1, \ldots, n)$ for $end_{repl(b)} = e_1 \ldots e_n$ and $t_H(b) = t_1 \ldots t_n$.
- each hyperedge keeps its label,
- $begin_X = begin_H$ and $end_X = end_H$.

The resulting multi-pointed hypergraph X is denoted by $REPLACE(H, repl)$.

Remark. The construction above is meaningful and determines (up to isomorphism) a unique hypergraph X if h is a mapping. This is automatically fulfilled whenever the *begin*-nodes and *end*-nodes of each replacing hypergraph are pairwise distinct. If one wants to avoid such a restriction, one has to require that the following application condition is satisfied for each $b \in B$: If $begin_{repl(b)} = x_1 \ldots x_m$ and $end_{repl(b)} = x_{m+1} \ldots x_{m+n}$ as well as $s_H(b) = y_1 \ldots y_m$ and $t_H(b) = y_{m+1} \ldots y_{m+n}$, then, for $i, j = 1, \ldots, m+n$, $x_i = x_j$ implies $y_i = y_j$.

3. Hyperedge-replacement grammars and languages

In this section we give a short summary of the basic notions on hyperedge-replacement grammars generalizing edge-replacement grammars as investigated e.g. in [7, 8] and context-free string grammars. Details and examples can be found in [9, 10].

Based on hyperedge replacement, one can derive multi-pointed hypergraphs from multi-pointed hypergraphs by applying productions of a simple form.

Definition 3.1 (productions and derivations). (1) Let $N \subseteq C$. A *production* (over N) is an ordered pair $p = (A, R)$ with $A \in N$ and $R \in \mathscr{H}_C$. A is called the *left-hand side*

[4] The sum symbols + and $\sum$ denote the disjoint union of sets; the symbol − denotes the set-theoretic difference.

of p and is denoted by $lhs(p)$, R is called the *right-hand side* and is denoted by $rhs(p)$. The type of p, denoted by $type(p)$, is given by the type of R.

(2) Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and P be a set of productions. A mapping $prod: B \to P$ is called a *production base* in H if $l_H(b) = lhs(prod(b))$ and $type(b) = type(rhs(prod(b)))$ for all $b \in B$.

(3) Let $H, H' \in \mathcal{H}_C$ and $prod: B \to P$ be a production base in H. Then H *directly derives* H' *through* $prod$ if H' is isomorphic to $REPLACE(H, repl)$ where $repl: B \to \mathcal{H}_C$ is given by $repl(b) = rhs(prod(b))$ for all $b \in B$. We write $H \underset{P}{\Rightarrow} H'$ or $H \Rightarrow H'$ in this case.

(4) A sequence of direct derivations $H_0 \underset{P}{\Rightarrow} H_1 \underset{P}{\Rightarrow} \cdots \underset{P}{\Rightarrow} H_k$ is called a *derivation* from H_0 to H_k (of length k). Additionally, in the case $H \cong H'$, we speak of a *derivation* from H to H' of length 0. A derivation from H to H' is shortly denoted by $H \overset{*}{\underset{P}{\Rightarrow}} H'$ or $H \overset{*}{\Rightarrow} H'$. If the length of the derivation should be stressed, we write $H \overset{k}{\underset{P}{\Rightarrow}} H'$ or $H \overset{k}{\Rightarrow} H'$.

(5) A direct derivation through $prod: \emptyset \to P$ is called a *dummy*.[5] A derivation is said to be *valid* if at least one of its steps is not a dummy.

Remarks. (1) The application of a production $p = (A, R)$ of type (m, n) to a multi-pointed hypergraph H requires the following two steps only:

(a) Choose a hyperedge e of type (m, n) with label A.

(b) Replace the hyperedge e in H by R.

(2) Some significant properties of direct derivations are the following: On the one hand, the definition of a direct derivation includes the case that no hyperedge is replaced. This dummy step derives a hypergraph isomorphic to the initial one. On the other hand, it includes the case that all hyperedges are replaced in one step. Moreover, whenever some hyperedges can be replaced in parallel, they can be replaced one after the other leading to the same derived hypergraph.

Using the introduced concepts of productions and derivations hyperedge-replacement grammars and languages can be introduced in a straightforward way.

Definition 3.2 (hyperedge-replacement grammars and languages). (1) A *hyperedge-replacement grammar* is a system $HRG = (N, T, P, Z)$ where $N \subseteq C$ is a set of *nonterminals*, $T \subseteq C$ is a set of *terminals*, P is a finite set of *productions* over N, and $Z \in \mathcal{H}_C$ is the *axiom*. The class of all hyperedge-replacement grammars is denoted by $\mathcal{HRG}$.

(2) HRG is said to be *typed* if there is a mapping $ltype: N \cup T \to \mathbb{N} \times \mathbb{N}$ such that, for each production $(A, R) \in P$, $ltype(A) = type(R)$ and $ltype(l_R(e)) = type(e)$ for all $e \in E_R$ and $ltype(l_Z(e)) = type(e)$ for all $e \in E_Z$. HRG is said to be be *well-formed*

[5] A production base $prod: B \to P$ in H may be *empty*, i.e., $B = \emptyset$. In this case $H \Rightarrow H'$ through $prod$ implies $H \cong H'$, and there is always a trivial direct derivation $H \Rightarrow H$ through $prod$.

if the right-hand sides of the productions are well-formed and all hyperedges in Z are well-formed.

(3) The *hypergraph language* $L(HRG)$ *generated by HRG* consists of all hypergraphs which can be derived from Z applying productions of P and which are terminally labeled,

$$L(HRG) = \{H \in \mathcal{H}_T \mid Z \overset{*}{\underset{P}{\Rightarrow}} H\}.$$

Remarks. (1) Even if one wants to generate graph languages rather than hypergraph languages, one may use nonterminal hyperedges because the generative power of hyperedge-replacement grammars increases with the maximum number of tentacles of a hyperedge involved in the replacement (see [10]).

(2) Without effecting the generative power, we will assume in the following that N and T are finite, $N \cap T = \emptyset$, and Z is a singleton with $l(Z) \in N$. Furthermore, we will assume that the hyperedge-replacement grammars considered in this paper are typed and well-formed.

The results presented in the following sections are mainly based on some fundamental aspects of hyperedge-replacement derivations. Roughly speaking, hyperedge-replacement derivations cannot interfere with each other as long as they handle different hyperedges. On the one hand, a collection of derivations of the form $e^\bullet \overset{*}{\Rightarrow} H(e)$ for $e \in E_R$ can be simultaneously embedded into R leading to a single derivation $R \overset{*}{\Rightarrow} H$. On the other hand, restricting a derivation $R \overset{*}{\Rightarrow} H$ to the handle $e^\bullet$ induced by the hyperedge $e \in E_R$ one obtains a so-called "restricted" derivation $e^\bullet \overset{*}{\Rightarrow} H(e)$ where $H(e) \subseteq H$. Finally, restricting a derivation to the handles induced by the hyperedges, and subsequently embedding them again returns the original derivation. In other words, hyperedge-replacement derivations can be distributed to the handles of the hyperedges without losing information. We state and use this result in the following recursive version concerning terminal hypergraphs which are derivable from handles.

Theorem 3.3. *Let* $HRG = (N, T, P, Z)$ *be a typed and well-formed hyperedge-replacement grammar,* $A \in N \cup T$, *and* $H \in \mathcal{H}_T$. *Then there is a derivation* $A^\bullet \Rightarrow R \overset{k}{\Rightarrow} H$ *for some* $k \geq 0$ [6] *if and only if* $A^\bullet \Rightarrow R$ *and, for each* $e \in E_R$, *there is a derivation* $l_R(e)^\bullet \overset{k}{\Rightarrow} H(e)$ *with* $H(e) \subseteq H$ *such that* $H \cong REPLACE(R, repl)$ *with* $repl(e) = H(e)$ *for* $e \in E_R$.

Remarks. (1) The derivation $l_R(e)^\bullet \overset{k}{\Rightarrow} H(e)$ may be valid or not. In the first case, it has the same form as the original derivation, but it is shorter as the original one. In the latter case, $H(e)$ is isomorphic to $e^\bullet$ (resp. $l_R(e)^\bullet$) and hence a terminal handle.

[6] For a symbol $A \in N \cup T$ with $ltype(A) = (m, n)$, $A^\bullet$ denotes an (m, n)-handle induced by A. (Note that (m, n)-handles induced by a symbol A are isomorphic).

(2) Given a derivation $R \overset{k}{\Rightarrow} H$, the derivation $l_R(e)^\bullet \overset{k}{\Rightarrow} H(e)$ for each $e \in E_R$ is called the *fibre* of e and—the other way round—the given derivation is the *joint embedding* of its fibres.

(3) Theorem 3.3 is a direct consequence of the Context-Freeness Lemma for hyperedge replacement presented in [6, Theorem II.2.4]. The proof of the theorem is given in [6, V.1.1].

4. Some graph-theoretic functions compatible with derivations

A hyperedge-replacement grammar as a generating device specifies a (hyper)graph language. Unfortunately, in a finite amount of time, the generating process only produces a finite section of the language explicitly (and even this may consume much time). Hence one may wonder what the hyperedge-replacement grammar can tell us about the generated language. As a matter of fact, by Theorem 3.3, we have the following nice situation. Given a hyperedge-replacement grammar and an arbitrary terminal (hyper)graph H with derivation $A^\bullet \Rightarrow R \overset{*}{\Rightarrow} H$, we get a decomposition of H into "smaller" components which are derivable from the handles of the hyperedges in R. If one is interested in values of graph-theoretic functions of derived (hyper)graphs, one may ask how a certain value of a derived (hyper)graph depends on values of the components. A function is said to be "compatible" with the derivation process of hyperedge-replacement grammars if it can be computed for each derived (hyper)graph H by computing the values (or related values) for the components and composing the values to the value of H.

In this section, we pick up several graph-theoretic functions and show that they are "compatible" with the replacement process of hyperedges. A formal definition of compatibility is given in the next section. We discuss the number of nodes and hyperedges, the number of paths and cycles, the length of a shortest path, the length of a longest simple path, the minimum and maximum degree, and the number of components.

Number of nodes and hyperedges

To illustrate our kind of investigation, we first consider the computation of the number of nodes and hyperedges in a hypergraph. In the following, let $|V_H|$ denote the number of nodes, $|INT_H|$ the number of internal nodes, and $|E_H|$ the number of hyperedges in a hypergraph H. Let $A^\bullet \Rightarrow R \overset{*}{\Rightarrow} H$ be a derivation of H, and, for $e \in E_R$, $l_R(e)^\bullet \overset{*}{\Rightarrow} H(e)$ be the fibre of $R \overset{*}{\Rightarrow} H$ induced by e. Then the node set of H consists of the nodes of R and the internal nodes of the components $H(e)$. Hence, the number of nodes in H can be computed from the number of nodes in R and the number of internal nodes in the $H(e)$ by summing up the numbers. For example, the hypergraph H in Fig. 1 has 8 nodes; 4 nodes are already in R, $H(e)$ and $H(e'')$ possess one internal node each, $H(e')$ has 2 internal nodes. Even simpler, the number of hyperedges in H can be determined by the number of hyperedges

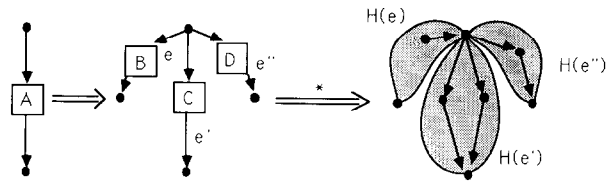

Fig. 1. A derivation of the form $A' \Rightarrow R \overset{*}{\Rightarrow} H$.

in $H(e)$ by summing up the numbers of the $H(e)$. In our example, $H(e)$, $H(e')$ and $H(e'')$ possess one, four and two hyperedges, respectively; therefore, the whole hypergraph H has seven hyperedges.

Theorem 4.1. *Let $H \in \mathscr{H}_T$, $A' \Rightarrow R \overset{*}{\Rightarrow} H$ be a derivation of H, and, for $e \in E_R$, $l_R(e)' \overset{*}{\Rightarrow} H(e)$ the fibre of $R \overset{*}{\Rightarrow} H$ induced by e. Then*

$$|V_H| = |V_R| + \sum_{e \in E_R} |INT_{H(e)}|,$$

$$|INT_H| = |INT_R| + \sum_{e \in E_R} |INT_{H(e)}|,$$

$$|E_H| = \sum_{e \in E_R} |E_{H(e)}|.$$

Proof. Without loss of generality, we can assume $H = REPLACE(R, repl)$ with $repl(e) = H(e)$ for $e \in E_R$ (cf. Theorem 3.3). By definition of replacement, $V_H = V_R + \sum_{e \in E_R} INT_{H(e)}$, $E_H = \sum_{e \in E_R} E_{H(e)}$, and $EXT_H = EXT_R$. Hence, we have $INT_H = V_H - EXT_H = V_R - EXT_R + \sum_{e \in E_R} INT_{H(e)} = INT_R + \sum_{e \in E_R} INT_{H(e)}$. Now the cardinality statements with respect to the number of nodes and the number of hyperedges follow directly from the set-theoretic statements. □

Remarks. (1) Similarly, the composed function *size* given by $size(H) = |V_H| + |E_H|$ can be handled. It uses the auxiliary function *intsize* given by $intsize(H) = |INT_H| + |E_H|$.

$$size(H) = |V_R| + \sum_{e \in E_R} intsize(H(e)),$$

$$intsize(H) = |INT_R| + \sum_{e \in E_R} intsize(H(e)).$$

(2) The density function *dens* given by $dens(H) = |E_H|/|V_H|$ if $|V_H| > 0$ (and $dens(H) = \diamond$ [7] otherwise) can also be expressed in the following way:

$$dens(H) = \frac{\sum_{e \in E_R} |E_{H(e)}|}{|V_R| + \sum_{e \in E_R} |INT_{H(e)}|}.$$

[7] *Dens*, *minipath*, and *maxpath* are defined to be functions with values in $\mathbb{N}^\diamond = \mathbb{N} + \{\diamond\}$, the set of all nonnegative integers plus a special symbol $\diamond$. We use this special symbol $\diamond$, if the considered function has no sensible integer value. We calculate with $\diamond$ as follows: $\forall i \in I\ \forall n_i \in \mathbb{N} + \{\diamond\}$,

- $\sum_{i \in I} n_i = \diamond$ and $\prod_{i \in I} n_i = \diamond$ if and only if $n_j = \diamond$ for some $j \in I$,
- $\min_{i \in I} n_i = \min_{i \in I'} n_i$ and $\max_{i \in I} n_i = \max_{i \in I'} n_i$ for $I' = \{i \in I \mid n_i \neq \diamond\}$, and $\min_{i \in I} n_i = \diamond$ and $\max_{i \in I} n_i = \diamond$ for $I = \emptyset$.

The expression for computing $dens(H)$ makes use of the possibility to compute the number of internal nodes as well as the number of hyperedges of the $H(e)$'s. It does not make use of the density of some of the $H(e)$'s.

For simplifying the technicalities, we restrict our following consideration to the class $\mathcal{ERG}$ of edge-replacement grammars in the sense of [7, 8]. To be more explicit, a (typed and well-formed) hyperedge-replacement grammar ERG is in $\mathcal{ERG}$ if and only if the right-hand sides of the productions as well as the axiom are (1, 1)-graphs. Note that, in this case, each $G \in L(ERG)$ is a (1, 1)-graph, i.e., a graph with two distinguished nodes $begin_G$ and end_G.

General assumption 4.2. For the rest of this section, let ERG be an edge-replacement grammar, G be a graph with distinguished nodes $begin_G$ and end_G, $A^{\bullet} \Rightarrow R \overset{*}{\Rightarrow} G$ be a derivation of G in ERG, and, for $e \in E_R$, $l_R(e)^{\bullet} \overset{*}{\Rightarrow} G(e)$ be the fibre of $R \overset{*}{\Rightarrow} G$ induced by e.

Simples path: number, minimum and maximum length

We are going to discuss how the number of simple paths of a graph G can be computed from the number of simple paths of the graphs $G(e)$. Considering for example the graph G in Fig. 2, one may observe that G contains five simple paths connecting $begin_G$ and end_G; one path is created by the edge e and lies completely in $G(e)$, the other four ones are created by the edges e' and e'' (forming a simple path connecting $begin_R$ and end_R) and are composed by a simple path of $G(e')$ joining $begin_{G(e')}$ and $end_{G(e')}$ and a simple path of $G(e'')$ joining $end_{G(e')} = begin_{G(e'')}$ and $end_{G(e'')}$. The product of the number of simple paths of $G(e')$ (joining $begin_{G(e')}$ and $end_{G(e')}$) and the number of simple paths of $G(e'')$ (joining $begin_{G(e'')}$ and $end_{G(e'')}$) decribes the number of simple paths created by the simple path in R built from e' and e''. The sum of all numbers of simple paths created by a simple path in R connecting $begin_R$ and end_R describes the number of all simple paths in G connecting $begin_G$ and end_G.

Although $G(e)$ contains a path connecting $begin_G$ and end_G, this path is not a shortest one in G. $G(e')$ contains a path of length one connecting $begin_G = begin_{G(e')}$ and $end_{G(e')}$ and $G(e'')$ contains a path of length one connecting $end_{G(e')} = begin_{G(e'')}$ and $end_{G(e'')} = end_G$. Hence, the path in R connecting $begin_R$ and end_R built from

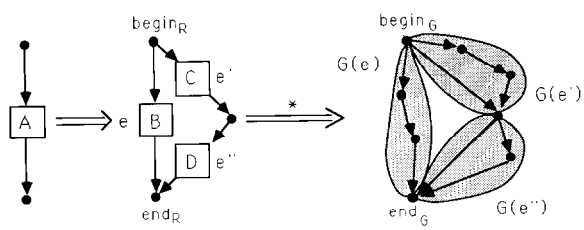

Fig. 2. A derivation of the form $A^{\bullet} \Rightarrow R \overset{*}{\Rightarrow} G$.

e' and e'' creates a path of length two. The sum of the minimum path length of $G(e')$ (joining $begin_{G(e')}$ and $end_{G(e')}$) and the minimum path length of $G(e'')$ (joining $begin_{G(e'')}$ and $end_{G(e'')}$) describes the minimum length of paths created by the path in R built from e' and e''. The minimum of all the minimum path lengths created by paths in R connecting $begin_R$ and end_R describes the minimum length of paths in G connecting $begin_G$ and end_G. Analogously, the maximum of all the maximum simple-path lengths created by simple paths in R connecting $begin_R$ and end_R describes the maximum length of simple paths in G connecting $begin_G$ and end_G.

We will use the following notions: Given a graph G, a *path* joining v_0 and v_n is a sequence $p = v_0, e_1, v_1, e_2, \ldots, e_n, v_n$ of alternating nodes and edges such that for $1 \leq i \leq n$, v_{i-1} and v_i are the nodes incident with e_i. If $v_0 = v_n$ then p is said to be a *cycle*. In this case, we do not distinguish p either from the cycle $v_i, e_{i+1}, \ldots, e_n, v_n, e_1, \ldots, e_i, v_i$ or from the cycle $v_i, e_i, \ldots, e_1, v_0, e_n, \ldots, e_{i+1}, v_i$. If in a path each node appears once, then the sequence is called a *simple* path. If each node appears once except that $v_0 = v_n$ and $n \geq 3$ then p is a *simple cycle*. The *length* of a path or a cycle p, denoted $length(p)$, is the number of edges it contains. "e *on* p" denotes the fact that e occurs in p.

Theorem 4.3. *For a* $(1, 1)$*-graph* G*, let* $PATH_G$ *denote the set of simple paths joining* $begin_G$ *and* end_G *and* $numpath(G)$ *the number of these paths in* G*. Moreover, let* $minpath(G)$ *and* $maxpath(G)$ *denote the minimum resp. maximum simple-path length, if any (and* $minpath(G) = maxpath(G) = \diamond$ *otherwise, see footnote* 7*). Then*

$$numpath(G) = \sum_{p \subset PATH_R} \prod_{e on p} numpath(G(e)),$$

$$minpath(G) = \min_{p \in PATH_R} \sum_{e on p} minpath(G(e)),$$

$$maxpath(G) = \max_{p \in PATH_R} \sum_{e on p} maxpath(G(e)).$$

Proof. Let G be a $(1, 1)$-graph and $\hat{p}$ be a simple path joining $begin_G$ and end_G. Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. Theorem 3.3). Particularly, $V_R \subseteq V_G$. Starting from $begin_G$ and running through $\hat{p}$, nodes of R are visited in a certain sequence $v_0, \ldots, v_n$. Cutting $\hat{p}$ in these nodes, one gets a sequence of simple paths $p_1, \ldots, p_n$ (where p_i connects v_{i-1} and v_i for $i = 1, \ldots, n$). For $i = 1, \ldots, n$, there exists $e_i \in E_R$ such that p_i belongs to $G(e_i)$. Otherwise, if p_i would be covered by more components, it would visit at least three nodes of R. By the same reason, e_i connects v_{i-1} and v_i (for $i = 1, \ldots, n$). Thus, $p = v_0, e_1, \ldots, e_n, v_n$ forms a simple path joining $begin_R$ and end_R and for $i = 1, \ldots, n$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$. Conversely, if $p = v_0, e_1, \ldots, e_n, v_n$ is a simple path joining $begin_R$ and end_R and, for $i = 1, \ldots, n$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$, then the replacement of the edges $e_1, \ldots, e_n$ by the corresponding paths $p_1, \ldots, p_n$ yields a simple

path joining $begin_G$ and end_G. Therefore,

$$PATH_G = \bigcup_{p \in PATH_R} \{replace(p, path) \mid path(e) \in PATH_{G(e)}\},$$

where $replace(p, path)$ denotes the path obtained from p by replacing all edges e on p by the corresponding paths $path(e)$.

The statements on the number of simple paths as well as the minimum and maximum simple-path length follow immediately from the following set-theoretical statement: Since, for a simple path $p \in PATH_R$, $replace(p, path) \neq replace(p, path')$ if and only if $path(e) \neq path'(e)$ for some e *on* p, the number of simple paths induced by p is $\prod_{e on p} numpath(G(e))$. Since different paths in R yield different paths in G, the number of simple paths in G is the sum of the numbers of simple paths induced by some simple path in R.

Since, for a simple path $p \in PATH_R$,

$$length(replace(p, path)) = \sum_{e on p} length(path(e)),$$

$$\min_{p \in PATH_G} length(p) = \min_{p \in PATH_R} \sum_{e on p} \min_{path(e) \in PATH_{G(e)}} length(path(e)).$$

Replacing min by max wherever it occurs, we obtain the statement for $\max_{p \in PATH_G} length(p)$. Using the definition of *minpath* and *maxpath*, we can derive the claimed minimum- and maximum-statements. □

Simple cycles: number, minimum and maximum length

The number of simple cycles, the minimum cycle length, and the maximum simple-cycle length of a graph can be determined using the computation of the number of simple paths, the minimum path length, and the maximum simple-path length, respectively. Considering for example the graph G in Fig. 2, G contains six simple cycles, one completely lying in $G(e')$, one completely lying in $G(e'')$, and four ones running through $G(e)$, $G(e')$, and $G(e'')$. The latter are composed by simple paths of $G(e)$, $G(e')$, and $G(e'')$. The product of the number of simple paths of $G(e)$, $G(e')$, and $G(e'')$ describes the number of simple cycles created by the simple cycle in R built from e, e', and e''. The sum of all numbers of simple cycles created by simple cycles in R describes the number of all composed simple cycles in G. Adding the numbers of simple cycles in the $G(e)$ $(e \in E_R)$, we obtain the number of all simple cycles in G. The minimum resp. maximum simple-cycle length can be handled in a similar way.

Theorem 4.4. *For a* $(1, 1)$*-graph* G*, let* $CYCLE_G$ *denote the set of simple cycles and* $numcycle(G)$ *the number of these cycles in* G*. Moreover, let* $mincycle(G)$ *and* $maxcycle(G)$ *denote the minimum resp. maximum simple-cycle length, if any; otherwise, let* $mincycle(G) = \diamond = maxcycle(G)$*. Then*

$$numcycle(G) = \sum_{c \in CYCLE_R} \prod_{e on c} numpath(G(e)) + \sum_{e \in E_R} numcycle(G(e)),$$

$$mincycle(G) = \min\left\{ \min_{c \in CYCLE_R} \sum_{e on c} minpath(G(e)), \min_{e \in E_R} mincycle(G(e)) \right\},$$

$$maxcycle(G) = \max\left\{ \max_{c \in CYCLE_R} \sum_{e on c} maxpath(G(e)), \max_{e \in E_R} maxcycle(G(e)) \right\}.$$

Proof. Let G be a $(1, 1)$-graph and $\hat{c}$ be a simple cycle. Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. Theorem 3.3). Particularly, $V_R \subseteq V_G$. Starting somewhere on $\hat{c}$ and running through $\hat{c}$, two cases may occur.

Case 1: *The cycle $\hat{c}$ belongs to a single component, say $G(e)$.*

Case 2: *The cycle $\hat{c}$ does not belong to a single component.* Then we consider the sequence $v_0, \ldots, v_n$ of visited nodes of R. Cutting $\hat{c}$ in these nodes, one gets a sequence of simple paths $p_1, \ldots, p_{n+1}$ (where p_i connects v_{i-1} and v_i for $i = 1, \ldots, n$ and p_{n+1} connects v_n and v_0). For $i = 1, \ldots, n+1$, there exists $e_i \in E_R$ such that p_i belongs to $G(e_i)$. Otherwise, if p_i would be covered by more components, it would visit at least three nodes of R. By the same reason, e_i connects v_{i-1} and v_i (for $i = 1, \ldots, n+1$). Thus, $c = v_0, e_1, \ldots, e_{n+1}, v_{n+1}$ forms a simple cycle in R and for $i = 1, \ldots, n+1$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$.

Conversely, if $c = v_0, e_1, \ldots, e_{n+1}, v_{n+1}$ is a simple cycle in R and, for $i = 1, \ldots, n+1$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$, then the replacement of the edges $e_1, \ldots, e_{n+1}$ by the paths $p_1, \ldots, p_{n+1}$ yields a simple cycle in G. Moreover, if $e \in E_R$ and c is a simple cycle in $G(e)$, then c is a cycle in G, too. Therefore,

$$CYCLE_G = \bigcup_{c \in CYCLE_R} \{replace(c, path) \mid path(e) \in PATH_{G(e)}\}$$

$$\cup \bigcup_{e \in E_R} \{c \mid c \in CYCLE_{G(e)}\}$$

where $replace(c, path)$ denotes the cycle obtained from c by replacing all edges e on c by the corresponding simple paths $path(e)$.

The statements on the number of simple cycles as well as the minimum and maximum simple-cycle length are immediate consequences of the following set-theoretical statement: Since, for a simple cycle c in R, $replace(c, path) \neq replace(c, path')$ if and only if $path(e) \neq path'(e)$ for some e *on* c, the number of simple cycles induced by c is $\prod_{e on c} numpath(G(e))$. Different simple cycles in R yield different cycles in G. Finally, cycles belonging to different components are different. Therefore, we obtain the claimed cardinality statement from the set-theoretic statement.

Since, for a simple cycle $c \in CYCLE_R$,

$$length(replace(c, path)) = \sum_{e on c} length(path(e)),$$

we get

$$\min_{c \in CYCLE_G} length(c) = \min\Big\{ \min_{c \in CYCLE_R} \sum_{e \text{ on } c} \min_{path(e) \in PATH_{G(e)}} length(path(e)),$$

$$\min_{e \in E_R} \min_{c \in CYCLE_{G(e)}} length(c) \Big\}.$$

A corresponding statement for $\max_{p \in CYCLE_G} length(c)$ can be obtained. Using the definition of *mincycle* and *maxcycle*, we can derive the claimed minimum- and maximum-statements. □

Minimum and maximum degree

We continue in discussing how the minimum (resp. maximum) degree of a graph G can be computed from the minimum (resp. maximum) degree of the $G(e)$. With respect to the determination of the minimum (maximum) degree of a graph G it is useful to know the minimum (maximum) degree of the internal nodes of $G(e)$ as well as the degrees of $begin_{G(e)}$ and $end_{G(e)}$. As illustrated in Fig. 3, the degree $D_G(v)$ of a node v in $V_R \subseteq V_G$ can be determined by summing up the degrees of the nodes $begin_{G(e)}$, $begin_{G(e')}$, $begin_{G(e'')}$ (e, e', and e'' are the outgoing edges from v), and $end_{G(e''')}$ (e''' is the only incoming edge in v). Building the minimum (maximum) of the $D_G(v)$ ($v \in V_R$), we obtain the minimum (maximum) degree of the nodes $v \in V_R$. Moreover, the degrees of the internal nodes of the $G(e)$ have to be taken into account.

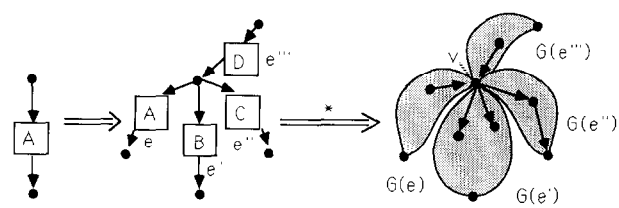

Fig. 3. A derivation of the form $A^* \Rightarrow R \stackrel{*}{\Rightarrow} G$.

Theorem 4.5. *For a graph G, let $mindegree(G)$ and $maxdegree(G)$ denote the minimum resp. maximum degree among the nodes of G. Moreover, let $minintdegree(G)$ denote the minimum degree among the internal nodes of G and $bdegree(G)$ and $edegree(G)$ the degree of $begin_G$ resp. end_G. Then*

$$mindegree(G) = \min\Big\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} minintdegree(G(e)) \Big\},$$

$$minintdegree(G) = \min\Big\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} minintdegree(G(e)) \Big\},$$

$$maxdegree(G) = \max\Big\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} maxdegree(G(e)) \Big\},$$

$$bdegree(G) = D_G(begin_G) \text{ and } edegree(G) = D_G(end_G),$$

where, for $v \in V_R$,

$$D_G(v) = \sum_{e \in s_R^{-1}(v)} bdegree(G(e)) + \sum_{e \in t_R^{-1}(v)} edegree(G(e)).$$

Proof. Let G be a $(1,1)$-graph and v be a node in G. Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. Theorem 3.3). Particularly, $V_R \subseteq V_G$. Thus, the degree of v in G, $d_G(v)$, can be determined as follows:

$$d_G(v) = \begin{cases} D_G(v) & \text{if } v \in V_R, \\ d_{G(e)}(v) & \text{if } v \in INT_{G(e)} \text{ for some } e \in E_R, \end{cases}$$

where

$$D_G(v) = \sum_{e \in s_R^{-1}(v)} d_{G(e)}(begin_{G(e)}) + \sum_{e \in t_R^{-1}(v)} d_{G(e)}(end_{G(e)}).$$

Using the notations introduced above, we get

$$D_G(v) = \sum_{e \in s_R^{-1}(v)} bdegree(G(e)) + \sum_{e \in t_R^{-1}(v)} edegree(G(e)).$$

By definition of the minimum degree of G and the minimum degree of G among the internal nodes,

$$\begin{aligned} mindegree(G) &= \min_{v \in V_G} d_G(v) \\ &= \min\left\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} \min_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \min\left\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} minintdegree(G(e)) \right\}, \end{aligned}$$

$$\begin{aligned} minintdegree(G) &= \min_{v \in INT_G} d_G(v) \\ &= \min\left\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} \min_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \min\left\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} minintdegree(G(e)) \right\}. \end{aligned}$$

Let $maxintdegree(G)$ denote the maximum degree among the internal nodes of G. Then we have analogously

$$\begin{aligned} maxdegree(G) &= \max_{v \in V_G} d_G(v) \\ &= \max\left\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} \max_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \max\left\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} maxintdegree(G(e)) \right\}. \end{aligned}$$

Since $maxdegree(G(e)) = \max\{maxintdegree(G(e)), d_{G(e)}(begin_{G(e)}), d_{G(e)}(end_{G(e)})\}$ and

$$d_{G(e)}(begin_{G(e)}),\ d_{G(e)}(end_{G(e)}) \leq \max_{v \in V_R} D_G(v),$$

the last expression can be simplified to

$$\max\left\{\max_{v \in V_R} D_G(v),\ \max_{e \in E_R} maxdegree(G(e)\right\}.$$

Finally, by the definition of *bdegree* and *edegree*, we get $bdegree(G) = D_G(begin_G)$ and $edegree(G) = D_G(end_G)$. This completes the proof. □

Number of components

Finally, we will show how the the number of components of a graph G can be computed provided that a derivation $A^{\cdot} \Rightarrow R \overset{*}{\Rightarrow} G$ of G with fibres $l_R(e)^{\cdot} \overset{*}{\Rightarrow} G(e)$ is given. We count the number of components of the graph R' obtained from R by removing all edges e for which $numpath(G(e))$ is zero (nodes which are in the same component of R' are in the same component of G) and look for "new" components obtained by the replacement of e by $G(e)$ neither containing $s_R(e)$ nor $t_R(e)$. In our example in Fig. 4, $numpath(G(e))$ as well as $numpath(G(e'))$ are zero, thus R' consists of two components. e creates no new component, its created nodes belong to the component containing the source resp. the target of e; e' creates one new component with two nodes, all other nodes belong to the component containing the source resp. the target of e'; e'' and e''' do not create new components. Therefore, G consists of three components.

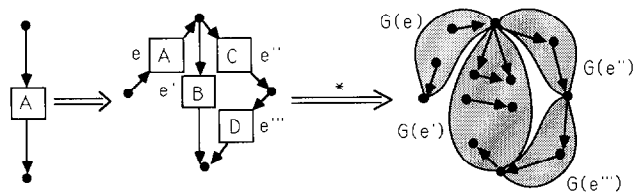

Fig. 4. A derivation of the form $A^{\cdot} \Rightarrow R \overset{*}{\Rightarrow} G$.

Theorem 4.6. *For a* $(1, 1)$*-graph* G*, let* $comp(G)$ *denote the number of connected components in* G *and* $newcomp(G)$ *denote the number of connected components neither containing* $begin_G$ *nor* end_G*. Then*

$$comp(G) = comp(R') + \sum_{e \in E_R} newcomp(G(e)),$$

$$newcomp(G) = newcomp(R') + \sum_{e \in E_R} newcomp(G(e))$$

where R' *is obtained from* R *by removing all* $e \in E_R$ *with* $numpath(G(e)) = 0$.

Remark. *newcomp*(G) can be expressed directly by

$$newcomp(G) = \begin{cases} comp(G)-1 & \text{if } numpath(G) \geq 1, \\ comp(G)-2 & \text{otherwise.} \end{cases}$$

Proof of Theorem 4.6. Let *COMP* be a connected component in G. Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. Theorem 3.3). Particularly, $V_R \subseteq V_G$. Now two cases may occur.

Case 1: $V_{COMP} \cap V_R = \emptyset$. Then $V_{COMP} \subseteq INT_{G(e)}$ for some $e \in E_R$, i.e., *COMP* is a "new" component created by e neither containing $begin_{G(e)}$ nor $end_{G(e)}$.

Case 2: $V_{COMP} \cap V_R \neq \emptyset$. Then the subgraph $COMP' \subseteq R$ with node set $V_{COMP'} = V_{COMP} \cap V_R$ and edge set $E_{COMP'} = \{e \in E_R \mid begin_{G(e)}, end_{G(e)} \in V_{COMP} \text{ and } numpath(G(e)) \geq 1\}$ forms a connected component of R' (since *COMP* is connected, *COMP'* is connected, too). Suppose now that there is a node $v \in V_{R'} - V_{COMP'}$ which is joined with a node $v' \in V_{COMP'}$ by an edge e in R'. Then $numpath(G(e)) \geq 1$ which means that v is joined with v' by a path in G. Since *COMP* is a connected component containing v', v is in $V_{COMP} \cap V_R = V_{COMP'}$, a contradiction. Consequently,

$$comp(G) \leq comp(R') + \sum_{e \in E_R} newcomp(G(e)).$$

Analogously,

$$newcomp(G) \leq newcomp(R') + \sum_{e \in E_R} newcomp(G(e)).$$

Conversely, let *COMP'* be a component in R'. Then, for each pair of nodes v, $v' \in V_{COMP'}$, v and v' are connected in G (they are connected in R' and, for $e \in E_{R'}$, $numpath(G(e)) \geq 1$). Consider now an arbitrary edge $e \in E_R - E_{R'}$. Two cases may occur. (a) The source and target node of e are in the same component of R'. Then all nodes of $G(e)$ which are connected with $begin_{G(e)}$ or $end_{G(e)}$ are in the same component as $s_R(e)$ and $t_R(e)$. (b) The source and the target node are in different components of R'. In any case, the nodes of $G(e)$ which are connected with $begin_{G(e)}$ are in the same component as $s_R(e)$ and all nodes of $G(e)$ which are connected with $end_{G(e)}$ are in the same component as $t_R(e)$. Different components in R' yield different components in G. Moreover, for an edge $e \in E_R$, the components in $G(e)$ which neither contain $begin_{G(e)}$ nor $end_{G(e)}$ yield additional components. Thus, we have $comp(R') + \sum_{e \in E_R} newcomp(G(e)) \leq comp(G)$ and $newcomp(R') + \sum_{e \in E_R} newcomp(G(e)) \leq newcomp(G)$. □

5. Compatible functions

In this section we introduce the notion of compatible functions in such a way that all functions considered in the previous section are special cases. Roughly speaking, a function f_0 on hypergraphs is said to be compatible with the derivation process of hyperedge-replacement grammars if, for each hypergraph H and each derivation of it, the value of H, $f_0(H)$, can be computed from the values of some specific subhypergraphs $H(e)$ determined by the fibres of the derivation. As the

examples will show, this view is oversimplified for most applications. To compute the value of H, it might be necessary to compute the values of some other related functions for the $H(e)$'s. Therefore, we use families of functions indexed by some finite set I and we need a mapping *assign* which determines the values for the $H(e)$'s with respect to the different value functions.

The notion of compatible functions generalizes obviously our earlier notion of compatible predicates (see [11]). More interestingly, a certain type of compatible functions that are composed of minima, maxima, sums, and products induce compatible predicates of the form: the function value of a graph exceeds a given fixed integer, or the function value does not exceed a fixed integer. Consequently, we get the decidability of the problems (1), (2), and (3) in the introduction for these predicates as a corollary.

Definition 5.1 (compatible functions). (1) Let $\mathscr{C}$ be a class of hyperedge-replacement grammars, I a finite index set, VAL a set of values, $f: \mathscr{H}_C \times I \to VAL$ a function[8], and f' a function defined on triples $(R, assign, i)$ with $R \in \mathscr{H}_C$, $assign: E_R \times I \to VAL$, $i \in I$, and values in VAL. Then f is called $(\mathscr{C}, f')$-*compatible* if, for all $HRG = (N, T, P, Z) \in \mathscr{C}$ and all derivations of the form $A^{\cdot} \Rightarrow R \overset{*}{\Rightarrow} H$ with $A \in N$ and $H \in \mathscr{H}_T$, and for all $i \in I$,

$$f(H, i) = f'(R, f(H(-), -), i).^{9}$$

(2) A function $f_0: \mathscr{H}_C \to VAL$ is called $\mathscr{C}$-*compatible* if functions f and f' and an index i_0 exist such that $f_0 = f(-, i_0)$ and f is $(\mathscr{C}, f')$-compatible.

Remarks. (1) Intuitively, a function is compatible if it can be computed for a large hypergraph derived by a fibre by computing some values for the smaller components of the corresponding shorter fibres. Such a function must be closed under isomorphism because the derivability of hypergraphs is independent of the representation of nodes and hyperedges.

(2) $\mathscr{C}$-compatibility is concerned with the productions of the grammars in $\mathscr{C}$ and not with their axioms. Therefore, we may assume that, for each $HRG = (N, T, P, Z) \in \mathscr{C}$ and each $Z' \in \mathscr{H}_C$, $HRG' = (N, T, P, Z')$ belongs to the class $\mathscr{C}$, too.

Examples 5.2. (a) Let $\mathscr{C} = \mathscr{HRG}$, $I = \{all, int\}$, $VAL = \mathbb{N}$, and f and f' be given by

$$f(H, all) = |V_H| \quad \text{(the number of all nodes in } H),$$

$$f(H, int) = |INT_H| \quad \text{(the number of all internal nodes in } H)$$

$$f'(R, assign, all) = |V_R| + \sum_{e \in E_R} assign(e, int),$$

$$f'(R, assign, int) = |INT_R| + \sum_{e \in E_R} assign(e, int).$$

[8] We assume that all considered functions are *closed under isomorphism*, i.e., for a function f, if $H \cong H'$ for some H, $H' \in \mathscr{H}_C$, then $f(H, i) = f(H', i)$ (resp. $f(H, assign, i) = f(H', assign, i)$) for all $i \in I$.

[9] $f(H(-), -)$ denotes the function defined by $f(H(-), -)(e, j) = f(H(e), j)$ for $e \in E_R$, $j \in I$. For $i \in I$, $f(-, i)$ denotes the unary function defined by $f(-, i)(H) = f(H, i)$ for all $H \in \mathscr{H}_C$.

By Theorem 4.1, we get the following nice relationship:

$$f(H, all) = |V_H| = |V_R| + \sum_{e \in E_R} |INT_{H(e)}|$$

$$= |V_R| + \sum_{e \in E_R} f(H(e), int) = f'(R, f(H(-), -), all).$$

Analogously, we obtain $f(H, int) = f'(R, f(H(-), -), int)$. Therefore, f is $(\mathcal{HRG}, f')$-compatible and $f_0 = f(-, all)$, concerning

(1) the number of nodes,

is $\mathcal{HRG}$-compatible.

(b) By Theorem 4.1, the following functions on hypergraphs are $\mathcal{HRG}$-compatible:

(2) the number of hyperedges,

(3) the size, and

(4) the density of a hypergraph.

(c) Let $\mathcal{C} = \mathcal{ERG}$, $I = \{np\}$, $VAL = \mathbb{N}^{\diamond}$, and f and f' be given by

$$f(G, np) = numpath(G) \quad \text{(the number of simple paths in } G),$$

$$f'(R, assign, np) = \sum_{p \in PATH_R} \prod_{e on p} assign(G(e), np).$$

By Theorem 4.3, we get the following relationship:

$$f(G, np) = numpath(G)$$

$$= \sum_{p \subset PATH_R} \prod_{e on p} numpath(G(e))$$

$$= \sum_{p \subset PATH_R} \prod_{e on p} f(G(e), np) = f'(R, f(G(-), -), np).$$

Therefore f is $(\mathcal{ERG}, f')$-compatible and $f_0 = f(-, np)$, concerning

(5) the number of simple paths connecting the external nodes, is $\mathcal{ERG}$-compatible.

(d) By Theorems 4.3–4.6, the following functions on graphs are $\mathcal{ERG}$-compatible:

(6) the minimum-path length (of paths connecting the external nodes),

(7) the maximum-simple-path length (of paths connecting the external nodes),

(8) the number of simple cycles,

(9) the minimum-cycle length,

(10) the maximum-simple-cycle length,

(11) the minimum degree,

(12) the maximum degree, and

(13) the number of components of a graph.

We recall now the notion of compatible predicates and relate it with a special type of compatible functions.

Definition 5.3 (compatible predicates). (1) Let $\mathscr{C}$ be a class of hyperedge-replacement grammars, I a finite index set, *PROP* a decidable predicate[10] defined on pairs (H, i) with $H \in \mathscr{H}_C$ and $i \in I$, and *PROP'* a decidable predicate on triples $(R, assign, i)$ with $R \in \mathscr{H}_C$, a mapping $assign: E_R \to I$, and $i \in I$. Then *PROP* is called $(\mathscr{C}, PROP')$-*compatible* if, for all $HRG = (N, T, P, Z) \in \mathscr{C}$ and all derivations $A^{\bullet} \Rightarrow R \overset{*}{\Rightarrow} H$ with $A \in N$ and $H \in \mathscr{H}_T$, and for all $i \in I$, $PROP(H, i)$ holds if and only if there is a mapping $assign: E_R \to I$ such that $PROP'(R, assign, i)$ holds and furthermore $PROP(H(e), assign(e))$ holds for each $e \in E_R$.

(2) A predicate $PROP_0$ on $\mathscr{H}_C$ is called $\mathscr{C}$-*compatible* if predicates *PROP* and *PROP'* and an index i_0 exist such that $PROP_0 = PROP(-, i_0)$[11] and *PROP* is $(\mathscr{C}, PROP')$-compatible.

Remarks. (1) Intuitively, a property is compatible if it can be tested for a large hypergraph with a long fibre by checking the smaller components of the corresponding shorter fibres.

(2) Examples of compatible properties are: connectivity, planarity, existence of Hamiltonian and Eulerian paths and cycles, k-colorability for each $k \geq 0$ (see [11, 5]).

(3) In [11] it is shown, that, for all $\mathscr{C}$-compatible properties $PROP_0$, it is decidable whether, given any hyperedge-replacement grammar $HRG \in \mathscr{C}$, $PROP_0$ holds for some $H \in L(HRG)$ and $PROP_0$ holds for all $H \in L(HRG)$.

Definition 5.4 (special types of compatible functions). Recall $\mathbb{N}^{\diamond} = \mathbb{N} + \{\diamond\}$.

(1) A function $f: \mathscr{H}_C \times I \to \mathbb{N}^{\diamond}$ is said to be $(\mathscr{C}, \min, \max, +, \cdot)$-*compatible* if there exists an f' such that for each right-hand side R of some production in $\mathscr{C}$ and each $i \in I$, $f'(R, -, i)$ corresponds to an expression formed with variables $assign(e, j)$ $(e \in E_R, j \in I)$ and constants from $\mathbb{N}$ by addition, multiplication, minimum, and maximum, and f is $(\mathscr{C}, f')$-compatible. The function is $(\mathscr{C}, \max, +, \cdot)$-*compatible* if the operation min does not occur.

(2) A function $f_0: \mathscr{H}_C \to \mathbb{N}^{\diamond}$ is $(\mathscr{C}, \min, \max, +, \cdot)$-*compatible* (resp. $(\mathscr{C}, \max, +, \cdot)$-*compatible*) if a function f and an index i_0 exist such that $f_0 = f(-, i_0)$ and f is $(\mathscr{C}, \min, \max, +, \cdot)$-compatible (resp. $(\mathscr{C}, \max, +, \cdot)$-compatible).

Theorem 5.5. *Let $PROP_0$ be a $\mathscr{C}$-compatible predicate. Then the function $f_0: \mathscr{H}_C \to \mathbb{N}^{\diamond}$ given by*

$$f_0(H) = \begin{cases} 1 & \text{if } PROP_0(H) \text{ holds,} \\ 0 & \text{otherwise,} \end{cases}$$

is $(\mathscr{C}, \max, +, \cdot)$-compatible.

[10] We assume that all considered predicates are *closed under isomorphism*, i.e., if a predicate Φ holds for $H \in \mathscr{H}_C$ and $H \cong H'$, then Φ holds for H', too.

[11] For $i \in I$, $PROP(-, i)$ denotes the unary predicate defined by $PROP(-, i)(H) = PROP(H, i)$ for all $H \in \mathscr{H}_C$.

Proof. Let $PROP_0$ be a $\mathscr{C}$-compatible predicate, $PROP$, $PROP'$ the corresponding predicates, I the corresponding index set, and i_0 the index such that $PROP_0 = PROP(-, i_0)$. Then we define functions f, f' as follows:

$$f(H, i) = \begin{cases} 1 & \text{if } PROP(H, i) \text{ holds,} \\ 0 & \text{otherwise,} \end{cases}$$

$$f'(R, assign', i) = \begin{cases} 1 & \text{if } PROP'(R, assign, i) \text{ holds for some } assign: E_R \to I \\ & \quad \text{with } assign'(e, assign(e)) = 1 \text{ for all } e \in E_R, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, $f_0(H) = f(H, i_0)$. Hence it remains to show that f is $(\mathscr{C}, f')$-compatible. Let $H \in \mathscr{H}_T$, $A^* \Rightarrow R \overset{*}{\Rightarrow} H$ a derivation of H in HRG, and, for $e \in E_R$, $l_R(e)^* \overset{*}{\Rightarrow} H(e)$ be the fibre of $R \overset{*}{\Rightarrow} H$ inducted by e.

If $f(H, i) = 1$, then $PROP(H, i)$ is satisfied. By the $(\mathscr{C}, PROP')$-compatibility of $PROP$, there is a mapping $assign: E_R \to I$ such that $PROP'(R, assign, i)$ and $PROP(H(e), assign(e))$ for $e \in E_R$ hold. Thus, $f(H(e), assign(e)) = 1$ for all $e \in E_R$. Using the definition of f', we obtain $f'(R, f(H(-), -), i) = 1$. Conversely, if $f'(R, f(H(-), -), i) = 1$, then there exists a mapping $assign: E_R \to I$ with $f(H(e), assign(e)) = 1$ for $e \in E_R$ such that $PROP'(R, assign, i)$ holds. Moreover, $PROP(H(e), assign(e))$ holds for $e \in E_R$ because $f(H(e), assign(e)) = 1$ for $e \in E_R$. Now the $(\mathscr{C}, PROP')$-compatibility of $PROP$ implies that $PROP(H, i)$ holds and the definition of f implies that $f(H, i) = 1$. Now $f(H, i) = f'(R, f(H(-), -), i)$ for all $i \in I$, i.e., f is $(\mathscr{C}, f')$-compatible. Moreover, $f'(R, assign', i)$ can be expressed as

$$\max\left\{0, \max_{as \in AS} \prod_{e \in E_R} assign'(e, as(e))\right\}$$

where $AS = \{assign: E_R \to I \mid PROP'(R, assign, i)\}$. Hence, f is $(\mathscr{C}, \max, +, \cdot)$-compatible. □

Conversely, $(\mathscr{C}, \min, \max, +, \cdot)$-compatible functions induce certain $\mathscr{C}$-compatible predicates.

Theorem 5.6. *Let $f_0: \mathscr{H}_C \to \mathbb{N}^\diamond$ be a $(\mathscr{C}, \min, \max, +, \cdot)$-compatible function for some class $\mathscr{C}$ of hyperedge-replacement grammars. Moreover, let $n \in \mathbb{N}^\diamond$. Then the predicates given by "$f_0(H) < n$", "$f_0(H) \leq n$", "$f_0(H) = n$", "$f_0(H) \geq n$", and "$f_0(H) > n$" are $\mathscr{C}$-compatible.*[12]

Proof. Let $f_0: \mathscr{H}_C \to \mathbb{N}^\diamond$ be a $(\mathscr{C}, \min, \max, +, \cdot)$-compatible function, f and f' the corresponding functions, I the corresponding index set and i_0 the index such that $f_0 = f(-, i_0)$. Let $J = \{0, 1, \ldots, n\} + \{\diamond, big\}$ for $n \in \mathbb{N}$ and $J = \{\diamond, big\}$ for $n = \diamond$. Moreover, let $[-]: \mathbb{N}^\diamond \to J$ be the mapping with $[m] = big$ for $m > n$ and $[m] = m$ otherwise. Now we define a new index set $\hat{I}$ and predicates $PROP$, $PROP'$ as follows:

- $\hat{I}$ is the set of all mappings from I to J, denoted by $\hat{I} = [I \to J]$.

[12] We assume that, for all $n \in \mathbb{N}^\diamond$, $\diamond \leq n$.

- $PROP(H, p)$ holds if and only if $[f(H, -)] = p$.
- $PROP'(R, assign, p)$ holds if and only if $[f'(R, assign', -)] = p$ for all $assign'$: $E_R \times I \to \mathbb{N}^\diamond$ with $[assign'(e, -)] = assign(e)$ $(e \in E_R)$.

First, we will show that $PROP$ is $(\mathscr{C}, PROP')$-compatible. If $PROP(H, p)$ holds, then $[f(H, -)] = p$. By the $(\mathscr{C}, f')$-compatibility of f, we have $[f'(R, f(H(-), -), i)] = [f(H, i)] = p(i)$ for $i \in I$. Now choose $assign: E_R \to [I \to J]$ as $assign(e) = [f(H(e), -)]$ for $e \in E_R$. Then, $PROP(H(e), assign(e))$ holds for all $e \in E_R$. Since f' is formed by addition, multiplication, minimum, and maximum, $[f'(R, assign', -] = p$ for all $assign'$ with $[assign'(e, -)] = assign(e)$ $(e \in E_R)$. Thus, by definition of $PROP'$, $PROP'(R, assign, p)$ is satisfied. Conversely, let $assign: E_R \to [I \to J]$ be a mapping such that $PROP'(R, assign, p)$ and $PROP(H(e), assign(e))$ for $e \in E_R$ hold. Then, by definition of $PROP$, $[f(H(e), -)] = assign(e)$ for $e \in E_R$. Moreover, by definition of $PROP'$, we obtain $[f'(R, f(H(-), -), i)] = p(i)$ for $i \in I$. Finally, by the $(\mathscr{C}, f')$-compatibility of f, $[f(H, i)] = [f'(R, f(H(-), -), i)]$ for $i \in I$. Thus, $[f(H, -)] = p$, i.e., $PROP(H, p)$ is satisfied.

Therefore, $PROP$ is $(\mathscr{C}, PROP')$-compatible. In particular, for each $p: I \to J$, $PROP$ $(-, p)$ is $\mathscr{C}$-compatible. Now,

$$f_0(H) > n \Leftrightarrow \bigvee_{p: I \to J \text{ with } p(i_0) = big} PROP(H, p).$$

Since the predicates $PROP(-, p)$ are $\mathscr{C}$-compatible and $\mathscr{C}$-compatible predicates are closed under disjunctions [5, Theorem 4.3], the predicate given by "$f_0(H) > n$" is $\mathscr{C}$-compatible. Analogously, it can be shown that the predicates given by "$f_0(H) < n$", "$f_0(H) \leq n$", "$f_0(H) = n$", and "$f_0(H) \geq n$" are $\mathscr{C}$-compatible. □

Corollary 5.7. *Let f_0 be a $(\mathscr{C}, \min, \max, +, \cdot)$-compatible function for some class $\mathscr{C}$ of hyperedge-replacement grammars. Moreover, let $n \in \mathbb{N}^\diamond$. Then, for all $HRG \in \mathscr{C}$, the following statements hold*:

(1) *It is decidable whether there is some $H \in L(HRG)$ with $f_0(H) \leq n$.*

(2) *It is decidable whether, for all $H \in L(HRG)$, $f_0(H) \leq n$.*

(3) *It is decidable in linear time whether a generated hypergraph $H \in L(HRG)$ represented by a derivation (resp. a derivation tree) has a value $f_0(H) \leq n$.*

Proof. Corollary 5.7 follows immediately from the $\mathscr{C}$-compatibility of the predicate "$f_0(-) \leq n$" (see Theorem 5.6) and the theorems for $\mathscr{C}$-compatible predicates given in [11]. □

Remark. Analogous statements hold for the relations $<$, $=$, $\geq$, and $>$.

6. A metatheorem for boundedness problems

Given a graph-theoretic function f_0 and a class $\mathscr{C}$ of hyperedge-replacement grammars, we are going to study the following type of questions for all $HRG \in \mathscr{C}$: "Is it decidable whether the values of all hypergraphs generated by HRG are

bounded?" The question turns out to be decidable provided that f_0 is $(\mathscr{C}, \max, +, \cdot)$-compatible. We call this result "metatheorem" because of its generic character: Whenever one can prove the $(\mathscr{C}, \max, +, \cdot)$-compatibility of a function (and we have given various examples in Section 4), one gets a particular decision result for this function as corollary of the metatheorem.

Theorem 6.1. *Let f_0 be a $(\mathscr{C}, \max, +, \cdot)$-compatible function for some class $\mathscr{C}$ of hyperedge-replacement grammars. Then, for all $HRG \in \mathscr{C}$, it is decidable whether or not there is a natural number $n \in \mathbb{N}$ such that $f_0(H) \leq n$ for all $H \in L(HRG)$.*

Proof. Let f_0 be a $(\mathscr{C}, \max, +, \cdot)$-compatible function. Let f and f' be the corresponding functions over the index set I so that f is $(\mathscr{C}, f')$-compatible and $f_0 = f(-, i_0)$ for some $i_0 \in I$. Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar in $\mathscr{C}$. Moreover, we may assume that, for each $A \in N$, the grammar $HRG(A) = (N, T, P, A^\bullet)$ is in $\mathscr{C}$, too (compare Remark 5.1).

The proof is based on the following idea. We construct a directed graph D containing all relevant information on derivations in HRG and look for certain cyclic structures in D. This enables us to decide whether or not the values may grow beyond any bound.

We need first some auxiliary notions. Let $J = \{\diamond, 0, 1, big\}$ and $[-]: \mathbb{N}^\diamond \to J$ be the mapping given by $[m] = big$ if $m \geq 2$ and $[m] = m$ otherwise. Given $A \in N$ and a mapping $p: I \to J$, a hypergraph $H \in \mathscr{H}_T$ is said to be an (A, p)-*hypergraph* if there is a derivation $A^\bullet \overset{*}{\Rightarrow} H$ of H such that $[f(H, -)] = p$. By Corollary 5.7, we can effectively determine the set

$$EXIST = \{(A, p) \mid \text{there exists an } (A, p)\text{-hypergraph}\}.$$

Furthermore, we need a function f'' which is derived from f' and is defined on hypergraphs R, functions $q: E_R \times I \to J$, and indices $i \in I$ by

$$f''(R, q, i) = [f'(R, assign, i)]$$

for some $assign: E_R \times I \to \mathbb{N}^\diamond$ with $[assign(e, j)] = q(e, j)$. f'' is always defined because $assign$ can be chosen as $assign(e, j) = q(e, j)$ if $q(e, j) \in \{\diamond, 0, 1\}$ and $assign(e, j) = 2$ otherwise. The well-definedness of f'' follows from the fact that $[f'(R, assign, i)] = [f'(R, assign', i)]$ if $[assign(e, j)] = [assign'(e, j)]$ for $e \in E_R$, $j \in I$, which holds for $(\mathscr{C}, \min, \max, +, \cdot)$-compatible functions.

By assumption, the function f is $(\mathscr{C}, \max, +, \cdot)$-compatible. Since multiplication distributes over addition and maximum and addition distributes over maximum, we may assume that $f'(R, -, i)$ is a maximum of sums where each sum is a product of constants and variables $assign(e, j)$ $(e \in E_R, j \in I)$. Substituting $assign(e, j)$ by $q(e, j)$ if $q(e, j) \in \{\diamond, 0, 1\}$, that means that the variables $assign(e, j)$ are kept as variables if $q(e, j) = big$, and simplifying the expression, i.e., deleting all sums that evaluate to $\diamond$, all products that evaluate to 0 and all factors that evaluate to 1, we obtain an expression $EXP(f'(R, -, i), q)$ formed as a maximum of sums where each sum is a product of constants and variables $assign(e, j)$ $(e \in E_R, j \in I)$, again. Let

$SIMPLE(f'(R,-,i),q)$ denote the set of all $assign(e,j)$ for which one sum in $EXP(f'(R,-,i),q)$ simply is $assign(e,j)$ and $NONTRIVIAL(f'(R,-,i),q)$ denote the set of all $assign(e,j)$ for which some sum in $EXP(f'(R,-,i),q)$ contains $assign(e,j)$, but also a nontrivial factor of some other product.

We can now define the directed graph D. As node set of D we choose

- $V=\{(A,p,i)\,|\,(A,p)\in EXIST \wedge p(i)=big\}$.

The set of edges of D contains two types of edges, called *greaterequal-edges* and *greater-edges* which can be given as follows.

- For nodes (A,p,i), (B,p',j), there is a *greaterequal-edge* (resp. *greater-edge*) connecting (A,p,i) and (B,p',j), denoted by $(A,p,i)\Rightarrow(B,p',j)$ (resp. $(A,p,i)\rightarrow(B,p',j)$) if there is a production $(A,R)\in P$ and a mapping $q: E_R\times I\rightarrow J$ such that, $p=f''(R,q,-)$, for all $e\in E_R$, $(l_R(e),q(e,-))\in EXIST$, and there exists an $e\in E_R$ such that $l_R(e)=B$, $q(e,-)=p'$, and $assign(e,j)$ is in $SIMPLE(f'(R,-,i),q)$ (resp. $assign(e,j)$ is in $NONTRIVIAL(f'(R,-,i),q)$).

In the following, we will show that the graph D contains all information to decide whether or not some function values grow beyond any bound. It turns out that the greater-edges of D play an important role. Remember that for each (B,p',j) in D, there is at least one derivation $B^{\bullet}\overset{*}{\Rightarrow}G$ in HRG with $[f(G,-)]=p'$ and $f(G,j)\geqslant 2$. We will show that, whenever we have a derivation $B^{\bullet}\overset{*}{\Rightarrow}G$ in HRG with $[f(G,-)]=p'$ and $f(G,j)\geqslant 2$ and there is a greater-edge $(A,p,i)\rightarrow(B,p',j)$ in D, then there exists a derivation $A^{\bullet}\overset{*}{\Rightarrow}H$ in HRG with $[f(H,-)]=p$ and $f(H,i)>f(G,j)$.

Claim 1. *Let (A,p,i), $(B,p',j)\in V$ and G be a (B,p')-hypergraph.*

(1) *If $(A,p,i)\rightarrow(B,p',j)$, then there is an (A,p)-hypergraph H with $f(H,i)>f(G,j)$.*

(2) *If $(A,p,i)\Rightarrow(B,p',j)$, then there is an (A,p)-hypergraph H with $f(H,i)\geqslant f(G,j)$.*

Proof of Claim 1. Let $(A,p,i)\rightarrow(B,p',j)$ be a greater-edge in D and G be an arbitrary (B,p')-hypergraph. By construction of D, there is some production $(A,R)\in P$ and some $q: E_R\times I\rightarrow J$ such that $p=f''(R,q,-)$ and, for all $e\in E_R$, $(l_R(e),q(e,-))\in EXIST$. Moreover, there is some $e'\in E_R$ with $l_R(e')=B$ and $q(e',-)=p'$. By definition of $EXIST$, for each $e\in E_R$, there exists a derivation $l_R(e)^{\bullet}\overset{*}{\Rightarrow}H(e)$ such that $[f(H(e),-)]=q(e,-)$. Since $l_R(e')=B$ and G is an (B,p')-hypergraph, there exists a derivation $l_R(e')^{\bullet}\overset{*}{\Rightarrow}G$ such that $[f(G,-)]=p'$. Joint embedding of the derivations $l_R(e)^{\bullet}\overset{*}{\Rightarrow}H(e)$ for $e\in E_R-\{e'\}$ and the derivation $l_R(e')^{\bullet}\overset{*}{\Rightarrow}G$—instead of $l_R(e')^{\bullet}\overset{*}{\Rightarrow}H(e')$—into R yields a derivation $R\overset{*}{\Rightarrow}H$. Combining it with the direct derivation $A^{\bullet}\Rightarrow R$, we get a derivation $A^{\bullet}\overset{*}{\Rightarrow}H$. By the $(\mathscr{C},\max,+,\cdot)$-compatibility of f, H is an (A,p)-hypergraph: Let $assign'$ be defined by $assign'(e',-)=f(G,-)$ and $assign'(e,-)=f(H(e),-)$ otherwise. Then we have $[f(H,-)]=[f'(R,assign',-)]=f''(R,[assign'],-)=f''(R,[f(H(-),-)],-)=f''(R,q,-)=p$. Since $assign(e',j)$ is in $NONTRIVIAL(f'(R,-,i),q)$, $f(H,i)=$

$f'(R, assign', i) > assign'(e', j) = f(G, j)$. (Observe that in the sum leading to the creation of $(A, p, i) \to (B, p', j)$ all remaining variables are substituted by at least 2.) Analogously, if $(A, p, i) \Rightarrow (B, p', j)$ is a greaterequaledge in D, we get $f(H, i) \geq f(G, j)$. □

In the following, we will look for special structures in D, called lasso structures. A subgraph L of D is called a *lasso structure* if it contains for each node a unique outgoing edge and each cycle contains a greater-edge. A node (A, p, i) of D is said to be *unbounded*, if, for all $n \in \mathbb{N}$, there is an (A, p)-hypergraph H with $f(H, i) > n$; otherwise it is said to be *bounded*.

Claim 2. *Let L be a lasso structure in D. Then every (A, p, i) in L is unbounded.*

Proof of Claim 2. Assume to the contrary and let k be minimal such that, for some (A, p, i) in L, for every (A, p)-hypergraph H we have $f(H, i) \leq k$. By the above claim, we have for the unique successor (B, p', j) of (A, p, i) in L and every (B, p')-hypergraph G that $f(G, j) \leq k$. By choice of k, there must exist a (B, p')-hypergraph G with $f(G, j) = k$ and we have $(A, p, i) \Rightarrow (B, p', j)$. Repeating this consideration we eventually get a lasso[13] in L whose cycle has greaterequal-edges only, a contradiction. □

Claim 3. *There exists a lasso structure L in D containing all unbounded (A, p, i).*

Proof of Claim 3. Let k be the maximal $f(H, i)$, where H is an (A, p)-hypergraph with derivation $A^\bullet \Rightarrow R \overset{*}{\Rightarrow} H$ such that (A, p, i) is bounded, but at least 2. Moreover, let $\Phi(k) = f'(R, as_k, i) + 1$ where $as_k(-, -) = k$. Then we define a subgraph L of D iteratively using sets OK and NOK, such that the following properties hold after each step:

(1) $OK \cup NOK = \{(A, p, i) \in V \mid (A, p, i)$ is unbounded$\}$;
(2) $OK \cap NOK = \emptyset$;
(3) $OK \subseteq V_L \subseteq OK \cup NOK$;
(4) each node in OK has a unique outgoing edge in L;
(5) each cycle of L contains a greater-edge;
(6) each maximal path[14] of L ends with a greater-edge.

Initially, let $OK = \emptyset$, $NOK = \{(A, p, i) \in V \mid (A, p, i)$ is unbounded$\}$, and L be the empty graph. For the iteration step, choose a derivation $A^\bullet \overset{*}{\Rightarrow} H$ of minimal length such that H is an (A, p)-hypergraph with $f(H, i) \geq \Phi(k)$ and $(A, p, i) \in NOK$. Let (A, R) be the first production of this derivation. We have hypergraphs $H(e)$, $e \in E_R$, and some $q : E_R \times I \to J$ such that $H(e)$ is an $(l_R(e), q(e, -))$-hypergraph for $e \in E_R$. $f'(R, -, i)$ in its simplified normal form is a maximum of sums, and, by definition of k and Φ, the maximum is attained for a sum containing a variable $assign(e, j)$

[13] If we add to a simple path $v_0, e_1, \ldots, e_n, v_n$, which has distinct nodes by definition, an edge connecting v_n and v_i $(i \in \{0, \ldots, n-1\})$, then the resulting graph is called a *lasso*.

[14] We call a path *maximal*, if its last node has outdegree 0.

such that $f(H(e), j) > k$. Put $(B, p', j) = (l_R(e), q(e, -), j)$, $OK = OK \cup \{(A, p, i)\}$, $NOK = NOK - \{(A,p, i)\}$, add to L the corresponding edge from (A, p, i) to (B, p', j) and—if necessary—(A, p, i) and/or (B, p', j). The first four conditions on L given above hold true (we have $f(H(e), j) > k$, therefore, $(B, p', j) \in OK \cup NOK$). If the new edge is a greater-edge, then each new cycle contains it, each new nontrivial maximal path ends with it ($(B, p', j) \notin OK$) or ends with a nontrivial maximal path that already existed ($(B, p', j) \in OK$). If the new edge is a greaterequal-edge, we must have $f(H(e), j) \geq \Phi(k)$, thus $(B, p', j) \in OK$, since we have chosen a shortest derivation. Hence any new nontrivial maximal path ends with an old one starting at (B, p', j). If there are new cycles, then we already had $(A, p, i) \in V_L$ and any edge leading to (A, p, i) is a greater-edge. Since the set $\{(A, p, i) \in V \mid (A, p, i)$ is unbounded$\}$ is finite, the construction is finished after a finite number of steps. After these steps, $OK = \{(A, p, i) \in V \mid (A, p, i)$ is unbounded$\}$ and $NOK = \emptyset$. Moreover, by (3), (4), and (5), $V_L = OK$, each node of L has a unique outgoing edge in L, and each cycle of L contains a greater-edge. Consequently, the constructed L is a lasso structure and, since $V_L = \{(A, p, i) \in V \mid (A, p, i)$ is unbounded$\}$, L contains all unbounded (A, p, i). □

Now we may proceed as follows:

(1) Construct the graph D for HRG.

(2) Check for each subgraph of D whether it is a lasso structure.

(3) Check for each lasso structure L whether it contains $(l(Z), p, i_0)$ for some $p: I \to J$. If there is a lasso structure L in D containing $(l(Z), p, i_0)$ (for some p), then, by Claim 2, $(l(Z), p, i_0)$ is unbounded, meaning that, for all $n \in \mathbb{N}$, there is an $(l(Z), p)$-hypergraph H with $f(H, i_0) > n$. Hence, for all $n \in \mathbb{N}$, there is a hypergraph $H \in L(HRG)$ with $f_0(H) > n$. Conversely, if, for all $n \in \mathbb{N}$, there is a hypergraph $H \in L(HRG)$ with $f_0(H) > n$, then, for all $n \in \mathbb{N}$, there is a p and an $(l(Z), p)$-hypergraph H with $f(H, i_0) > n$. Since the number of p's is finite, we can find some p such that, for all $n \in \mathbb{N}$, there is an $(l(Z), p)$-hypergraph H with $f(H, i_0) > n$. Therefore, $(l(Z), p, i_0)$ is unbounded and, by Claim 3, there exists a lasso structure containing $(l(Z), p, i_0)$. This completes the proof of the theorem. □

Combining the compatibility results of Section 4 and Theorem 6.1, one obtains a list of decidability results concerning boundedness problems.

Corollary 6.2. *For each edge-replacement grammar $ERG \in \mathscr{ERG}$ and each function in the following list, it is decidable whether (or not) the function values of the graphs in $L(ERG)$ grow beyond any bound:*

(1) *the number of nodes,*

(2) *the number of edges,*

(3) *the size,*

(4) *the number of simple paths connecting the external nodes,*

(5) *the number of simple cycles,*

(6) *the maximum-simple-path length (of paths connecting the external nodes),*
(7) *the maximum-simple-cycle length,*
(8) *the maximum degree, and*
(9) *the number of components of a graph.*

Proof. The statements (1)–(8) follow directly from Theorems 4.1, 4.3–4.5, and 6.1. Statement (9) follows from Theorems 4.6 and 6.1 as follows: Theorem 4.6 makes use of $comp(R')$ and $newcomp(R')$ where R' is obtained from R by removing the edges $e \in E_R$ for which $numpath(G(e)) = 0$. It can be shown that $comp(R')$ and $newcomp(R')$ can be expressed as maxima of products. Let bec, $nbec: \mathcal{G}_C \to \mathbb{N}^{\diamond}$ be the functions defined by $bec(G) = 1$ if there is a path from $begin_G$ to end_G, $bec(G) = 0$ otherwise, and $nbec(G) = 1 - bec(G)$. By results of [11] and Theorem 5.5, these functions are $(\mathcal{ERG}, \max, +, \cdot)$-compatible, and, thus, we may use $bec(G(e))$ and $nbec(G(e))$ when giving a formula for $comp(R')$ and $newcomp(R')$.

Let, for $E \subseteq E_R$, $\langle E \rangle$ denote the spanning subgraph of R with edge set E. Then

$$comp(R') = \max_{E \subseteq E_R} \left[\prod_{e \in E} bec(G(e)) \cdot \prod_{e \in E_R - E} nbec(G(e)) \right] \cdot comp(\langle E \rangle)$$

and

$$newcomp(R') = \max_{E \subseteq E_R} \left[\prod_{e \in E} bec(G(e)) \cdot \prod_{e \in E_R - E} nbec(G(e)) \right] \cdot newcomp(\langle E \rangle).$$

(Observe that the product of products gives 1 for exactly one E, namely the one that induces R', and 0 otherwise. Furthermore, $comp(\langle E \rangle)$ and $newcomp(\langle E \rangle)$ are constants.) □

Remarks. (1) Remember that the functions "number of nodes", "number of hyperedges", and "size" are compatible for arbitrary hyperedge-replacement grammars $HRG \in \mathcal{HRG}$.

(2) Although we avoided the troublesome technicalities in this paper, we are convinced that the other considerations of this section work for more general types of hyperedge-replacement grammars, too. For example, all the statements should hold even if the class $\mathcal{ERG}$ is replaced by the class of all hyperedge-replacement grammars which generate ordinary graph languages and use hyperedges with a bounded number of tentacles as nonterminals. We even think that the considered functions are compatible for arbitrary hyperedge-replacement grammars if their definition is properly adapted to hypergraphs.

Let us mention that some problems—like the connectivity problem, the maximum-clique-size problem, and the chromatic-number problem—are trivial in the following sense: For all hyperedge-replacement grammars HRG, there is a bound (depending only on HRG) such that the function values of all graphs do not exceed the bound. This knowledge can be used to show that other boundedness problems—as the

minimum-clique-covering problem and the maximum-independent-set problem—are decidable.

The *clique partition number* of a graph G, $C(G)$, is the smallest number of cliques that form a partition of the node set V_G. A set of nodes in a graph G is *independent* if no two of them are adjacent. The largest number of nodes in such a set is called the *independence number* of G and is denoted by $I(G)$.

Theorem 6.3. *For each hyperedge-replacement grammar $HRG \in \mathcal{HRG}$ generating a set of graphs, it is decidable whether the clique partion number and the independence number of graphs in $L(HRG)$ grows beyond any bound.*

Proof. Since for each hyperedge-replacement grammar HRG, the maximum clique size is bounded on $L(HRG)$, say by $c(HRG) \geq 1$, and, for each $G \in L(HRG)$,

$$\frac{|V_G|}{c(HRG)} \leq C(G) \leq |V_G|,$$

the clique partition number is bounded on $L(HRG)$ if and only if the number of nodes is bounded on $L(HRG)$. Since for each hyperedge-replacement grammar HRG the chromatic number is bounded on $L(HRG)$, say by $k(HRG) \geq 1$, for each $H \in L(HRG)$, the maximum number of equally colored nodes in a $k(HRG)$-coloring of G, $MAX(G)$, is a lower bound of $I(G)$. On the other side, $|V_G| \leq k(HRG) \cdot MAX(G)$. Thus,

$$\frac{|V_G|}{k(HRG)} \leq I(G) \leq |V_G|.$$

Therefore, the independence number is bounded on $L(HRG)$ if and only if the number of nodes is bounded on $L(HRG)$. □

Certainly, there are boundedness problems where Theorem 6.1 fails. For example, we can give a function for which the corresponding boundedness problem is undecidable.

Theorem 6.4. *Let $\mathcal{C} \subseteq \mathcal{HRG}$ be a class of hyperedge-replacement grammars such that each edge-replacement grammar with a single nonterminal is in $\mathcal{C}$. Let automorph be the function with automorph$(H) = |V_H|$ if H has a nontrivial automorphism group and automorph$(H) = 0$ otherwise. Then it is undecidable whether, for a given $HRG \in \mathcal{C}$, automorph grows beyond any bound.*

Proof. For each context-free string grammar $CFSG$ (without ε-productions), there is a hyperedge-replacement grammar obtained as follows: Each production (A, w) is transformed into a production (A, w^*) where w^* is a directed path whose edges are labeled with the letters of w. A further transformation yields the hyperedge-replacement grammar HRG: The terminals of $CFSG$ are added to the nonterminals,

and, for each terminal a of $CFSG$, we have a new terminal symbol a' and a production $(a, R(a))$ where $R(a)$ consists of two parallel edges in opposite directions each labeled with a'. Then the function *automorph* grows beyond any bound if and only if $L(CFSG)$ contains palindromes of unbounded length. A well-known application of the Post Correspondence Problem (PCP) shows that it is undecidable whether a context-free string grammar generates a palindrome. This application involves context-free string grammars with one nonterminal which generate palindromes of unbounded length if they generate a palindrome at all. (The latter fact follows from the observation that any repetition of a solution to a PCP is a solution, too.) □

7. Discussion

Each class $\mathscr{C}$ of graph grammars and each function f on graphs with integer values establish a boundedness problem: *Is it decidable, for all graph languages $L(GG)$ generated by GG in $\mathscr{C}$, whether or not there is a bound n such that $f(G) \leq n$ for all $G \in L(GG)$?*

In this paper, we have been able to show that the boundedness problem is solvable for classes of hyperedge-replacement grammars and functions that are compatible with the derivation process and where the values of derivable graphs are composed of maxima, sums, and products of component values. Although this result applies to a variety of examples it seems to be strangely restricted. Further research should clarify the situation.

(1) We would expect that the metatheorem holds under more general or modified assumptions. Especially, we would like to know how functions given by minima or differences or divisions work.

(2) We suspect that certain combinations of arithmetic operations are not allowed. For instance, maxima and minima seem to antagonize each other, at least sometimes.

(3) Compatible functions are defined for arbitrary domains. But we have got significant results only for boolean and integer values. What about other domains? How can arbitrary compatibility be exploited? How do other meaningful interpretations look like?

Acknowledgment

We are very grateful to our referees whose comments led to various improvements.

References

[1] A. Arnborg, J. Lagergren and D. Seese, Problems easy for tree-decomposable graphs, in: *Proc. ICALP'88*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 38-51.

[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, Lecture Notes in Computer Science, Vol. 291 (Springer, Berlin, 1987) 133–146.
[4] P. Della Vigna and C. Ghezzi, Context-free graph grammars, *Inform. and Control* **37** (1978) 207–233.
[5] A. Habel, Graph-theoretic properties compatible with graph derivations, in: *Proc. WG'88*, Lecture Notes in Computer Science, Vol. 344 (Springer, Berlin, 1989) 11–29.
[6] A. Habel, Hyperedge-replacement: grammars and languages, Ph.D. Thesis, Univ. Bremen, Germany, 1989.
[7] A. Habel and H.-J. Kreowski, On context-free graph languages generated by edge replacement, Lecture Notes in Computer Science, Vol. 153 (Springer, Berlin, 1983) 143–158.
[8] A. Habel and H.-J. Kreowski, Characteristics of graph languages generated by edge replacement, *Theoret. Comput. Sci.* **51** (1987) 81–115.
[9] A. Habel and H.-J. Kreowski, May we introduce to you: hyperedge replacement, Lecture Notes in Computer Science, Vol. 291 (Springer, Berlin, 1987) 15–26.
[10] A. Habel and H.-J. Kreowski, Some structural aspects of hypergraph languages generated by hyperedge replacement, in: *Proc. STACS'87*, Lecture Notes in Computer Science, Vol. 247 (Springer, Berlin, 1987) 207–219.
[11] A. Hable, H.-J. Kreowski and W. Vogler, Metatheorems for decision problems on hyperedge replacement graph languages, *Acta Inform.* **26** (1989) 657–677.
[12] A. Habel, H.-J. Kreowski and W. Vogler, Decidable boundedness problems for hyperedge-replacement graph grammars, in: *Proc. TAPSOFT'89*, Lecture Notes in Compuer Science, Vol. 351 (Springer, Berlin, 1989) 275–289.
[13] D. Janssens, G. Rozenberg and E. Welzl, The bounded degree problem for NLC grammars is decidable, *J. Comput. System Sci.* **33** (1986) 415–422.
[14] H.-J. Kreowski, A pumping lemma for context-free graph languages, Lecture Notes in Computer Science, Vol. 73 (Springer, Berlin, 1979) 270–283.
[15] C. Lautemann, Decomposition trees: structured graph representation and efficient algorithms, in: *Proc. CAAP'88*, Lecture Notes in Computer Science, Vol. 299 (Springer, Berlin, 1988) 28–39.
[16] T. Lengauer and E. Wanke, Efficient analysis of graph properties on context-free graph languages, in: *Proc. ICALP'88*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 379–393.
[17] G. Rozenberg and E. Welzl, Boundary NLC graph grammars—Basic definitions, normal forms, and complexity, *Inform. and Control* **69** (1986) 136–167.
[18] G. Rozenberg and E. Welzl, Graph theoretic closure properties of the family of boundary NLC graph languages, *Acta Inform.* **23** (1986) 289–309.
[19] A.O. Slisenko, Context-free graph grammars as a tool for describing polynomial-time subclasses of hard problems, *Inform. Process. Lett.* **14** (1982) 52–56.