

# Adaptive Star Grammars and Their Languages

Frank Drewes<sup>1</sup>, Berthold Hoffmann<sup>2</sup>,  
Dirk Janssens<sup>3</sup>, and Mark Minas<sup>4</sup>

<sup>1</sup> Umeå universitet, Sweden  
drewes@cs.umu.se

<sup>2</sup> Universität Bremen, Germany  
hof@informatik.uni-bremen.de

<sup>3</sup> Universiteit Antwerpen, Belgium  
dirk.janssens@ua.ac.be

<sup>4</sup> Universität der Bundeswehr München, Germany  
mark.minas@unibw.de

## Abstract

Motivated by applications that require mechanisms for describing the structure of object-oriented programs, adaptive star grammars are introduced, and their fundamental properties are studied. In adaptive star grammars, rules are actually schemata which, via the cloning of so-called multiple nodes, may adapt to potentially infinitely many contexts when they are applied. This mechanism makes adaptive star grammars more powerful than context-free graph grammars. Nevertheless, they turn out to be restricted enough to share some of the basic characteristics of context-free devices. In particular, the underlying substitution operator enjoys associativity and confluence properties quite similar to those of context-free graph grammars, and the membership problem for adaptive star grammars is decidable.

## 1 Introduction

In this paper, we introduce adaptive star grammars, and study their basic properties. The motivation behind this theoretical work comes from applications in software engineering, most notably model transformation and refactoring. Specification methods and tools in this area often represent models of programs or similar structures by graphs. The set of all graphs which are valid descriptions of a model forms a graph language. Since programs have a recursive syntactical structure, graph languages of this type cannot be specified by graph schemas or similar mechanisms [14]. In contrast, the

specification of recursively structured graph languages is the natural realm of graph grammars. However, the use of graph grammars in tools of the kind mentioned is meaningful only if they enjoy good algorithmic properties, a requirement which is fulfilled by hyperedge and node replacement grammars [11, 3, 10]. Unfortunately, too little of the structure of, e.g., object-oriented programs can be captured with these types of graph grammars. In particular, they lack the ability to generate models that, e.g., obey scope rules, capture overriding of methods or contain references that relate used program entities to their definitions.

Therefore, we propose an extension of these grammars, the *adaptive star grammar*, which is able to capture such properties.<sup>1</sup> Adaptive star grammars have been devised for specifying software models based on graphs in [5], and have been used in an extensive case study on the refactoring of graphs that represent object-oriented programs [20]. A very concrete use of this extension is made in [4], where we use adaptive star grammars to specify languages of graphs that may be substituted for a variable in a graph transformation rule. Recent work [7, 12] has shown that adaptive star grammars – with application conditions and some notational enhancements – can define static models of object-oriented programs in a more natural way than by meta-modelling.

A *star* in a graph  $G$  is a nonterminal node together with all its incident edges and adjacent nodes. The latter are called the border nodes of the star. We consider only stars having neither parallel edges nor loops; nonterminal nodes are not allowed to be adjacent. Now, a star rule replaces a star with another graph. The latter is glued to the border nodes of the star, while the nonterminal node and its incident edges are removed. This replacement process is similar to the well-known notion of hyperedge replacement, where the star corresponds to the hyperedge being replaced (and the nodes it is attached to). To surpass the generative power of hyperedge replacement, border nodes of the left-hand side of a star rule may be so-called multiple nodes. These nodes can be *cloned* prior to the application of the star rule. Cloning simply replicates a multiple node together with its incident edges any number of times (including 0). Thus, a star rule containing multiple nodes is actually a rule schema. In fact, it is often useful to allow multiple nodes even in the host graph, so that the rewriting can take place at the level

---

<sup>1</sup>Adaptive star grammars should not be confused with *adaptable grammars* in the sense of [1] where the derivation process of a word grammar (context-free, or attributed) may add completely new rules to the original grammar.

of graph schemata, which can be cloned afterwards.

We note here that the set nodes of PROGRES [18] and FUJABA [16] are similar to our multiple nodes. In the model transformation language GMORPH [19], a more general notion of cloning is provided whose *collection containers* correspond to the notion of a *multiple subgraph*. A similar concept is proposed in [13].

As our first main results, we show that cloning can be applied either early or late, without restricting the generative power. Early cloning means that all multiple nodes, both in the host graph and in the rule, are replaced by their clones before the replacement takes place, while late cloning means that cloning is postponed as long as possible. Depending on the situation, it may be convenient to restrict attention to either early or late cloning. The former works on graphs without multiple nodes throughout the entire derivation, but results usually in an infinitely branching derivation relation. It is especially useful if one is interested in generating a particular graph, e.g., when trying to solve the membership problem. In contrast, late cloning has a finitely branching derivation relation. It avoids the commitment to a particular number of clones until this decision becomes unavoidable. This is especially useful when it is not known beforehand how many clones of a multiple node will actually be needed later on in the derivation process.

Our second result provides a parsing algorithm for adaptive star languages, thus showing that the membership problem for adaptive star grammars is solvable. We mention here that it was shown in a precursor of this paper [5] that a more general variant of adaptive star grammars can generate all recursively enumerable string languages. These generalized adaptive star grammars allow for parallel edges in stars (and, thus, rule applications using non-injective occurrence morphisms). The mentioned result shows that the generalized adaptive star grammars in [5] are too powerful. Therefore, we only consider the more restricted version (called straight adaptive star grammars in [5]) in the present paper.

Finally, we discuss to which extent adaptive star grammars enjoy the properties commonly ascribed to context-free grammatical devices. Adaptive star grammars are—intentionally—not entirely context-free. We work out the difference by contrasting adaptive star grammars with the axiomatic notion of context-free devices presented by Courcelle [2]. Roughly speaking, it turns out that the only deviation lies in the fact that star replacement is a nondeterministic operation. In particular, adaptive star replacement obeys nondeterministic versions of the associativity and confluence axioms of [2].

Consequently, the language generated by an adaptive star grammar can be obtained by evaluating trees over nondeterministic operations.

The structure of this paper is as follows. In the next section, we define the basic notions regarding stars and star replacement. Section 3 introduces the cloning operation. Based on this, adaptive star grammars are introduced in Section 4. Early and late cloning are defined and studied in Section 5. In Section 6, the parsing algorithm for adaptive star languages is presented. Finally, Section 7 shows that adaptive star grammars enjoy properties quite similar to those known from context-free grammatical devices. In Section 8, we conclude with some remarks on related and future work. Some of these sections are an extension of [5].

## 2 Star Replacement

Throughout this paper,  $\mathbb{N}$  denotes the nonnegative integers (i.e.,  $\mathbb{N}$  includes zero). For  $n \in \mathbb{N}$ ,  $[n] = \{1, \dots, n\}$ . The powerset of a set  $S$  is denoted by  $\wp(S)$ . Given a binary relation  $\rightarrow \subseteq A \times A$  over a set  $A$ , we let  $\rightarrow^+$  and  $\rightarrow^*$  denote the transitive and the transitive and reflexive closure of  $\rightarrow$ . Moreover, for  $n \in \mathbb{N}$ ,  $\rightarrow^n$  denotes the  $n$ -fold composition of  $\rightarrow$  with itself (where, by convention,  $\rightarrow^0$  is the identity on  $A$ ).

The grammars considered in this article derive directed, node- and edge-labeled graphs that may contain loops and parallel edges. Throughout the paper, let  $\mathbf{S}$  be a set of labels which is partitioned into two disjoint, countably infinite sets  $\dot{\mathbf{S}}$  and  $\bar{\mathbf{S}}$  of node and edge labels, resp. A finite subset  $S$  of  $\mathbf{S}$  is called a labeling alphabet. Its two components are  $\dot{S} = S \cap \dot{\mathbf{S}}$  and  $\bar{S} = S \cap \bar{\mathbf{S}}$ .

**Definition 2.1 (Graph)** A *graph*  $G = \langle \dot{G}, \bar{G}, s_G, t_G, \dot{\ell}_G, \bar{\ell}_G \rangle$  consists of finite sets  $\dot{G}$  of *nodes* and  $\bar{G}$  of *edges*, of *source* and *target* functions  $s_G, t_G: \bar{G} \rightarrow \dot{G}$ , and of node and edge labeling functions  $\dot{\ell}_G: \dot{G} \rightarrow \dot{\mathbf{S}}$  and  $\bar{\ell}_G: \bar{G} \rightarrow \bar{\mathbf{S}}$ .

If all labels of nodes and edges in  $G$  are in  $S \subseteq \mathbf{S}$ , then  $G$  is a *graph over*  $S$ . The class of all graphs over  $S$  is denoted by  $\mathcal{G}_S$ .

We use common terminology regarding graphs. For instance, an edge is said to be *incident* with its source and target nodes, and makes these nodes *adjacent* to each other. An edge is a *loop* if its source and target are the same, and two edges are *parallel* if they have the same source and target nodes. *Multiple edges* are parallel edges with identical labels.  $G \subseteq H$  expresses that  $G$  is a *subgraph* of  $H$ , and  $G \uplus H$  is the *disjoint union* of  $G$  and  $H$ . For a node

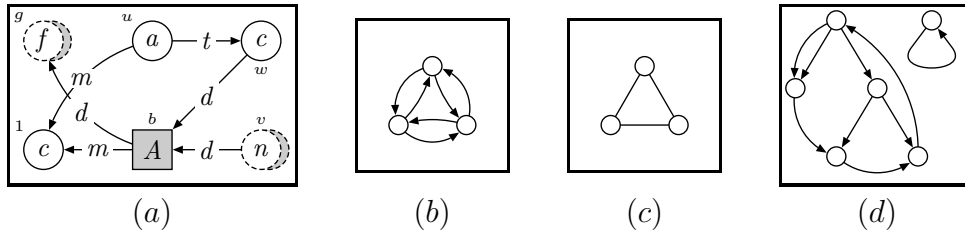


Figure 1: Some graphs

set  $A \subseteq \dot{G}$ ,  $G \setminus A$  denotes the subgraph of  $G$  induced by  $\dot{G} \setminus A$ . Morphisms and isomorphisms are defined as usual. Thus, formally, a morphism  $g$  from  $G$  to  $H$  is a pair of mappings  $\dot{g}: \dot{G} \rightarrow \dot{H}$  and  $\bar{g}: \bar{G} \rightarrow \bar{H}$  that preserves labels (i.e.,  $\dot{\ell}_G = \dot{\ell}_H \circ \dot{g}$  and  $\bar{\ell}_G = \bar{\ell}_H \circ \bar{g}$ ) as well as sources and targets (i.e.,  $s_H \circ \bar{g} = \dot{g} \circ s_G$  and  $t_H \circ \bar{g} = \dot{g} \circ t_G$ ). A morphism  $g$  is an isomorphism iff  $\dot{g}$  and  $\bar{g}$  are bijective. The notation  $G \cong_g H$  denotes the fact that graphs  $G$  and  $H$  are isomorphic via the isomorphism  $g$ .

**Example 2.2 (Graphs)** Figure 1 shows four graphs. We draw graphs as usual: Nodes are depicted as “blobs” containing their labels; edges are drawn as arrows from their source to their target nodes, and have their labels ascribed. Small numbers or letters aside a blob are used to refer to a node. The significance of the different shapes of nodes  $g, b, w$  in Figure (a) will be explained later.

To obtain unlabelled edges and nodes, as in Figures (b)–(c), we may assume that each of  $\bar{\mathbf{S}}$  and  $\dot{\mathbf{S}}$  contains an “invisible” label which, by convention, is not drawn. If, as in Figure (b), some of the edges in a graph are symmetric, i.e., are pairs of edges that carry the same label and connect the same nodes in the opposite direction, we draw one line instead of the two arrows running to and from. Figure (c) represents Figure (b) according to this convention, and may be seen as an undirected graph. Finally, Figure (d) shows a graph consisting of two leaf-connected trees with one and five nodes, respectively.

In the graph grammars defined further below, so-called stars are the items that are substituted by graphs.

**Definition 2.3 (Nonterminal and Star)** Let  $\overset{\circ}{\mathbf{S}} \subseteq \dot{\mathbf{S}}$  be an infinite supply of node labels which are said to be *nonterminal*.<sup>2</sup> The set  $\overset{\circ}{G} = \{x \in \dot{G} \mid$

<sup>2</sup>We assume that the set  $\dot{\mathbf{S}} \setminus \overset{\circ}{\mathbf{S}}$  of remaining node labels is infinite as well.

$\ell_G(x) \in \overset{\circ}{\mathbf{S}}$  denotes the *nonterminal nodes* in a graph  $G$ . For a node  $x$  in  $G$ , the subgraph consisting of  $x$  and its adjacent nodes and incident edges is denoted by  $G(x)$ .  $G(x)$  is a *star* if  $x$  is nonterminal,  $G(x)$  contains no other nonterminal nodes, and neither loops nor parallel edges. In this case, the node  $x$  is the *center node*, and the nodes adjacent to  $x$  are the *border nodes*.

We are only interested in graphs where  $G(x)$  is a star, for all  $x \in \overset{\circ}{G}$ :

**General Assumption 2.4** Throughout this paper, we consider only graphs  $G$  such that  $G(x)$  is a star, for all  $x \in \overset{\circ}{G}$ . In particular,  $\mathcal{G}_S$  is restricted to graphs satisfying this assumption.

Next, we introduce an important basic notion for this paper: a simple kind of graph transformation that replaces a star by a graph.

**Definition 2.5 (Star Replacement)** A *star rule*  $r = \langle y, R \rangle$  consists of a graph  $R$  and a distinguished nonterminal node  $y \in \overset{\circ}{R}$ . The star  $R(y)$  is its *left-hand side*, and the graph  $R \setminus \{y\}$  is its *right-hand side*.

Let  $G$  be a graph and  $x \in \overset{\circ}{G}$  such that  $G(x) \cong_g R(y)$  for some isomorphism  $g$ . Then the graph  $H = G[x /_g r]$  is obtained from the disjoint union  $G \uplus R$  by identifying  $R(y)$  with  $G(x)$  according to the isomorphism  $g$  and removing  $x$  and its incident edges.

Obviously, the construction of  $H = G[x /_g r]$  is unique only up to isomorphism. In particular, it is not affected by taking isomorphic copies of  $G$  and  $R$  (and changing  $x$  and  $g$  accordingly). To simplify proofs, we may therefore restrict our attention to the case where  $g$  is an identity morphism, and the morphisms that map  $G \setminus \{x\}$  and  $R \setminus \{y\}$  to their images in  $H$  are identities as well. If this assumption is made, we omit  $g$  in the notation  $G[x /_g r]$ , thus denoting the star replacement by  $G[x / r]$ .

**Example 2.6 (Star Replacement)** Four star rules are shown in Figure 2. When drawing a star rule  $\langle x, P \rangle$ , we depict it like a syntax rule, as “ $P(x) ::= P \setminus \{x\}$ ”; digits or letters ascribed to the border nodes in  $P(x) \setminus \{x\}$  indicate the common border nodes on the left and right-hand side of a rule. Rectangular boxes are used for nonterminal nodes, and round ones for the others. Star rules  $\langle x, P \rangle$  and  $\langle y, R \rangle$  with the same left-hand side  $P(x) = R(y)$ , are written “ $P(x) ::= P \setminus \{x\} \mid R \setminus \{y\}$ ”.

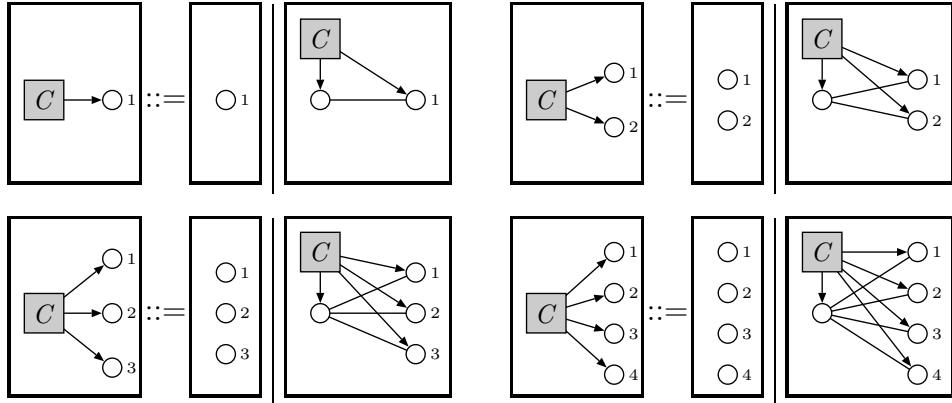


Figure 2: Some star rules

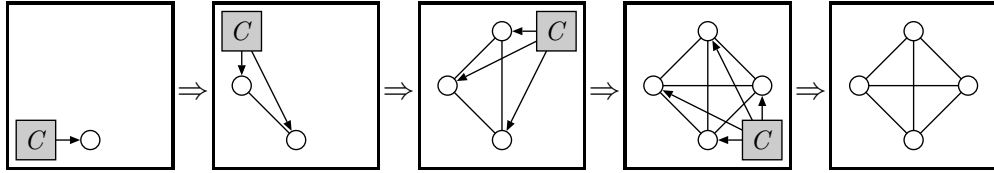


Figure 3: A sequence of star replacements

In Figure 3 we show star replacement steps that apply the rules shown in Figure 2 to the initial star in the order in which they are given. Here, a complete graph with four nodes is generated.

Star replacement corresponds to a very simple form of *double-pushout graph transformation* [8], where rules have simple left-hand sides, and the occurrence morphisms are required to be injective. This will make it easy to prove some basic properties of star replacement.

Grammars based on star replacement, called star grammars, can now be defined in the usual manner.

**Definition 2.7 (Star Grammar)** A *star grammar* is a triple  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  consisting of

- a labeling alphabet  $S$ ,
- a finite set  $\mathcal{P}$  of star rules with labels in  $S$ , and
- a star  $Z \in \mathcal{G}_S$ , called *initial star* of  $\Gamma$ .

For graphs  $G, H \in \mathcal{G}_S$ , there is a *derivation step*  $G \rightarrow_{\mathcal{P}} H$  if  $H = G[x \downarrow_g r]$

for some  $x \in \overset{\circ}{G}$ ,  $r = \langle y, R \rangle \in \mathcal{P}$ , and an isomorphism  $g: R(y) \rightarrow G(x)$ . The language generated by  $\Gamma$  is  $L(\Gamma) = \{G \in \mathcal{G}_{S \setminus \overset{\circ}{S}} \mid Z \rightarrow_{\mathcal{P}}^+ G\}$ .

If nothing else is explicitly mentioned, we shall always silently assume that the initial star of a star grammar consists of an isolated nonterminal node. Note that, by the usual argument, this assumption does not restrict the generative power of star grammars, even compared to the case where an arbitrary initial graph  $Z'$  may be used. To simulate the latter, choose a new nonterminal node label as the label of the initial star  $Z$ , and extend  $\mathcal{P}$  by a rule that turns  $Z$  into  $Z'$ .

The reader may have noticed that star replacement can be seen as a variant of hyperedge replacement [11, 3]. Indeed, we may define a (graph-generating) hyperedge replacement grammar to be a star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  such that every star in  $\{Z\} \cup \{R \mid \langle y, R \rangle \in \mathcal{P}\}$  is a hyperedge.<sup>3</sup> Here, a hyperedge is a star whose edges  $e_1, \dots, e_n$  originate at the center node and are labeled with  $1, \dots, n$ . We now prove that this restriction does not affect the generative power, i.e., hyperedge replacement grammars are as powerful as star grammars:

**Theorem 2.8** For every star grammar  $\Gamma$ , there is a hyperedge replacement grammar  $\Gamma'$  such that  $L(\Gamma') = L(\Gamma)$ .

*Proof* We sketch how to construct  $\Gamma'$  from  $\Gamma = \langle S, \mathcal{P}, Z \rangle$ . In a first step, we reverse the directions of edges which point towards center nodes. For this, add a fresh edge label  $a^{\text{out}}$  for every  $a \in \bar{S}$ . Now, in every star which occurs in the graph  $R$  of a rule  $\langle y, R \rangle \in \mathcal{P}$ , reverse each edge  $e$  originating at a border node, and turn its label into  $\bar{\ell}_G(e)^{\text{out}}$ . Obviously, the resulting star grammar  $\Gamma^{\text{out}}$  satisfies  $L(\Gamma^{\text{out}}) = L(\Gamma)$ .

Finally, for a graph  $G$ , let  $G^{\text{hyp}}$  be the set of all graphs  $H$  obtained from  $G$ , as follows. For every  $x \in \overset{\circ}{G}$ , choose an arbitrary order  $e_1, \dots, e_n$  on the edges of  $G(x)$ , turn the label of  $x$  into  $\langle \bar{\ell}_G(x), \bar{\ell}_G(e_1) \cdot \dots \cdot \bar{\ell}_G(e_n) \rangle$  (considered as a new nonterminal node label), and then replace the label of every  $e_i$  by  $i$ . Now,  $\Gamma'$  is obtained from  $\Gamma^{\text{out}}$  by replacing every rule  $\langle x, R \rangle$  with the set of all rules  $\langle x, R' \rangle$  such that  $R' \in R^{\text{hyp}}$ . Additionally,  $Z$  is turned into the unique element of  $Z^{\text{hyp}}$ .

---

<sup>3</sup>In fact, the so-defined hyperedge replacement grammars are slightly more general than those commonly found in the literature, because they generate graphs in which not only the edges, but also the nodes are labeled.

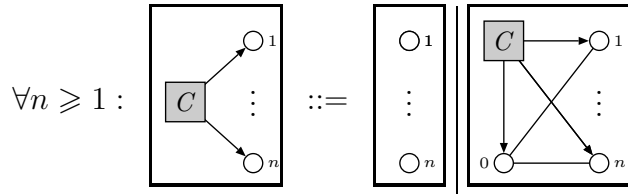


Figure 4: Star rule schemes for deriving complete graphs

It follows by induction on the length of derivations that, for all graphs  $G$ ,  $Z \rightarrow_{\Gamma^{\text{out}}}^* G$  if and only if  $Z \rightarrow_{\Gamma'}^* G'$  for some  $G' \in G^{\text{hyp}}$ . This completes the proof, because  $G^{\text{hyp}} = \{G\}$  for graphs  $G$  without nonterminal nodes.  $\square$

Taking into account the well-studied relation between hyperedge and node replacement (see, e.g., [9, 10]), Theorem 2.8 shows that not all node replacement grammars can be turned into equivalent star replacement grammars. The intuitive reason for this is that the left-hand side of a star rule has a fixed number of edges, whereas nonterminal nodes in node-replacement grammars can be adjacent to any number of nodes. We illustrate this well-known effect by means of an easy example, in order to motivate the concepts introduced in the following sections.

**Example 2.9 (Generating Complete Graphs)** It is well-known that the language of all complete graphs can be generated by node replacement (see [9, Figure 4.5]). The star rules in Figure 2 indicate why we cannot generate this language by star replacement: Intuitively, we would need an infinite number of rules, because the replaced nonterminal must, in each step, be adjacent to all nodes in the complete graph being generated. Thus, the rule to be applied in step  $n$  must have a left-hand side with  $n$  border nodes.

However, all these rules are built according to a common schema that is shown in Figure 4. The ellipses “ $\circ_1 \cdots \circ_n$ ” in the rule stand for any number  $n \geq 1$  of nodes that are connected to the  $C$ -node and node 0. In other words,  $r_n$  is a schema for an infinite number of star rules.

The notion of cloning introduced in the next section formalizes the ellipses used in Figure 4: certain nodes in a star rule are considered to be placeholders for an arbitrary number of nodes. Thus rules become schemes that can be adapted to have as many copies of such nodes before they are applied.

### 3 Cloning

We introduce the notion of a *multiple node* representing  $n \geq 0$  ordinary nodes; these nodes are called *clones* as each of them has equal incident edges, and is adjacent to the same nodes as the multiple node. A similar concept exists in the graph transformation languages PROGRES and FUJABA [18, 16].

We use a special set of node labels designating multiple nodes. Formally, we assume from now on that  $\dot{\mathbf{S}} \setminus \check{\mathbf{S}}$  contains a subset  $\ddot{\mathbf{S}}$  of *multiple node labels*: these designate nodes that may be replicated. Nodes which are neither multiple nor nonterminal are said to be *singular*. We assume that there is a bijective correspondence between the sets  $\dot{\mathbf{S}} \setminus (\check{\mathbf{S}} \cup \ddot{\mathbf{S}})$  and  $\check{\mathbf{S}}$  of singular and multiple node labels: for a singular node label  $l$ , the corresponding multiple node label is denoted by  $\check{l}$ . In other words, every singular node label  $l$  has a copy  $\check{l}$  among the multiple node labels. A node is said to be singular or multiple depending on its label. The set of multiple nodes in a graph  $G$  is denoted by  $\ddot{G}$ , i.e.,  $\ddot{G} = \{v \in \dot{G} \mid \check{l}_G(v) \in \ddot{\mathbf{S}}\}$ . In figures, multiple nodes are drawn like set nodes in PROGRES, with a “shadow” and dashed lines, as is seen in Figure 5. As we will see later, only graphs containing neither nonterminal nor multiple nodes are considered as final results of derivations in adaptive star grammars. Therefore, we call a graph *terminal* if it contains neither nonterminal nor multiple nodes.

We can now define the cloning operation, by which a multiple node can be turned into any number of singular nodes, its clones. However, we also want to be able to create clones that are multiple nodes. Thus, we define  $G[x \cdot \frac{m}{k}]$  to be obtained from  $G$  by replacing the multiple node  $x$  with  $m$  multiple and  $k$  singular clones.

**Definition 3.1 (Cloning Operation)** Let  $G$  be a graph,  $x \in \ddot{G}$  a multiple node, and  $m, k \geq 0$ . *Cloning  $x$*  yields the graph  $G[x \cdot \frac{m}{k}]$  constructed as follows. Let  $G'(x)$  be obtained from  $G(x)$  by replacing the label  $\check{l}$  of  $x$  by  $l$ . Then take the disjoint union of the graph  $G \setminus \{x\}$ ,  $m$  copies of  $G(x)$ , and  $k$  copies of  $G'(x)$ . Finally, identify the  $m + k + 1$  copies of each node in  $\dot{G}(x) \setminus \{x\}$  with each other.

The  $m + k$  copies of  $x$  in  $G[x \cdot \frac{m}{k}]$  are called the *clones of  $x$* . Obviously, Definition 3.1 determines  $G[x \cdot \frac{m}{k}]$  only up to isomorphism. However, to make it easier to refer to specific nodes in proofs, we assume from now on that cloning does not rename nodes and edges in  $G \setminus \{x\}$ . Thus, if  $C$  is the set of clones of  $x$  in  $G[x \cdot \frac{m}{k}]$ , then  $G[x \cdot \frac{m}{k}] \setminus C = G \setminus \{x\}$ . Obviously, if

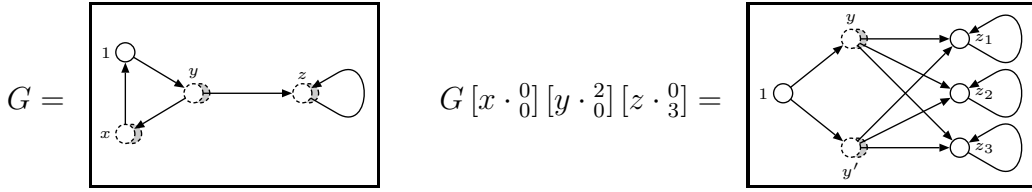


Figure 5: Cloning a graph

$m > 0$ , we may even assume that cloning only adds new nodes and edges incident with them, so that  $G \subseteq G[x \cdot \frac{m}{k}]$  in this case.

**Example 3.2 (Cloning)** In Figure 5 we show a graph  $G$  with three multiple nodes  $x$ ,  $y$ , and  $z$ . If we make zero multiple and singular clones of  $x$ ,  $x$  is removed with its incident edges. If we make two multiple clones of  $y$ , and three singular clones of  $z$ , the edges incident with  $y$  and  $z$  are copied as well. This also holds for loops. If two adjacent multiple nodes have  $m$  and  $n$  clones, resp., edges between these nodes are copied  $m \cdot n$  times. The edge from  $y$  to  $z$  is thus copied six times.

Some simple properties of the cloning operation are worth mentioning.

**Fact 3.3** For all graphs  $G$  and multiple nodes  $x \in \ddot{G}$ , the following hold:

- (1)  $G[x \cdot \frac{0}{0}] \cong G \setminus \{x\}$ .
- (2)  $G[x \cdot \frac{1}{0}] \cong G$ .
- (3)  $G[x \cdot \frac{2}{0}][x \cdot \frac{0}{0}] \cong G$ .

The cloning operation can be broken into sequences of three basic cloning operations that either remove a multiple node  $x$  altogether, or add one singular or one multiple clone of  $x$ ; this is useful for proofs.

**Lemma 3.4 (Basic Cloning Steps)** For all graphs  $G$ ,  $x \in \ddot{G}$ , and  $m, k \in \mathbb{N}$ , there are  $n \in \mathbb{N}$  and  $(m_1, k_1), \dots, (m_n, k_n) \in \{(0, 0), (1, 1), (2, 0)\}$  such that

$$G[x \cdot \frac{m_1}{k_1}] \cdots [x \cdot \frac{m_n}{k_n}] \cong G[x \cdot \frac{m}{k}].$$

Cloning operations of these three types are said to be *basic*.

*Proof* For  $m, k \geq 0$ , we have

$$\begin{aligned} G[x \cdot \frac{m}{k+1}] &\cong G[x \cdot \frac{1}{1}][x \cdot \frac{m}{k}] \\ G[x \cdot \frac{m+1}{k}] &\cong G[x \cdot \frac{2}{0}][x \cdot \frac{m}{k}] \end{aligned}$$

so that, by induction, every cloning operation  $G[x \cdot \frac{m}{k}]$  can be broken into a sequence of basic operations that add one singular or multiple clone, at the end leaving us with the basic removal operation  $G[x \cdot \frac{0}{0}]$ .  $\square$

We mention that the basic cloning steps cannot be described by finitely many rules in the double-pushout approach, because the node to be cloned may be incident with an arbitrary number of edges. In fact, cloning is closely related to node replacement; it could easily be described using edNCE node-replacement rules extended to graphs with loops and multiple edges.

The graph obtained by cloning a number of nodes is independent of the order in which those nodes are treated.

**Lemma 3.5 (Cloning is Commutative)** For a graph  $G$  with distinct multiple nodes  $x$  and  $y$ , and for  $m, k, m', k' \geq 0$ ,

$$G[x \cdot \frac{m}{k}][y \cdot \frac{m'}{k'}] \cong G[y \cdot \frac{m'}{k'}][x \cdot \frac{m}{k}]$$

*Proof* By Lemma 3.4, it suffices to consider only the cases where  $(m, k), (m', k') \in \{(0, 0), (1, 1), (2, 0)\}$ . The only cases which might not be entirely obvious are those where  $x$  and  $y$  are adjacent and  $(m, k), (m', k') \in \{(1, 1), (2, 0)\}$ . As these cases differ only in the labels of clones created, it suffices to consider  $(m, k) = (1, 1) = (m', k')$ . An entirely formal proof could be given by viewing these two cloning operations as graph transformations in the DPO approach. However, since the situation should be clear enough, we just note that each edge between  $x$  and  $y$  gives rise to exactly one edge between each clone of  $x$  and each clone of  $y$  in  $G_1 = G[x \cdot \frac{m}{k}][y \cdot \frac{m'}{k'}]$  and  $G_2 = G[y \cdot \frac{m'}{k'}][x \cdot \frac{m}{k}]$ . On the other hand, each clone of  $x$  (and of  $y$ , respectively) is connected to the nodes that are not clones in  $G_1$  and  $G_2$  in the same way as  $x$  ( $y$ , respectively) is connected to the nodes of  $G \setminus \{x, y\}$  in  $G$ . Finally, the clones of  $x$  and  $y$  have the same loops as  $x$  and  $y$  in  $G$ . Thus the edges of  $G_1$  and  $G_2$  correspond, and it is easily seen that so do the labels.  $\square$

Using the previous result, we can define a cloning operation that clones all multiple nodes in a graph simultaneously. For each multiple node, the necessary information about the number of desired clones is given by a so-called multiplicity function.

**Definition 3.6 (Simultaneous Cloning)** Let  $G$  be a graph. A function  $\mu: \check{G} \rightarrow \mathbb{N}^2$  is called a *multiplicity function* (*multiplicity*, for short) for  $G$ .

If  $\ddot{G} = \{x_1, \dots, x_k\}$  (where  $x_1, \dots, x_k$  are pairwise distinct), then  $G^\mu$  is the graph defined by

$$G^\mu = G[x_1 \cdot \mu(x_1)] \cdots [x_k \cdot \mu(x_k)]$$

By Lemma 3.5,  $G^\mu$  is uniquely defined (up to isomorphism). In the following, we drop the requirement that a multiplicity for  $G$  has to have exactly  $\ddot{G}$  as its domain. Multiplicities specified for multiple nodes that do not belong to  $G$  are simply disregarded. For multiple nodes  $x \in \ddot{G}$  not in the domain of  $\mu$ , we set  $\mu(x) = (1, 0)$ .

In formal constructions, it is sometimes inconvenient that  $G^\mu$  is determined only up to isomorphism. Therefore, we assume from now on that  $G^\mu$  is constructed in some deterministic manner, yielding a concrete graph that depends only on  $G$  and  $\mu$ . In particular, this means that multiplicities can be applied in sequence, i.e., it makes sense to consider  $G^{\mu\nu}$ , where  $\mu$  and  $\nu$  are multiplicities. Clearly, in this case, there is another multiplicity  $\eta$  such that  $G^{\mu\nu} \cong G^\eta$ . We shall in the following denote  $\eta$  by  $\nu \circ \mu$  and assume that cloning is done in such a way that  $G^{\mu\nu} \cong G^{\nu \circ \mu}$ .

## 4 Adaptive Star Grammars

In this section, we define adaptive star grammars. The rules of these grammars are star rules which may contain multiple nodes that can be cloned before a rule is applied. The graphs being derived may contain multiple nodes as well, and so they may also be cloned in order to make a rule applicable. Let us first define adaptive star replacement.

**Definition 4.1 (Adaptive Star Replacement)** Let  $G$  be a graph,  $x \in \ddot{G}$ , and  $r = \langle y, R \rangle$  a star rule with  $\ell_G(x) = \ell_R(y)$  and  $\ddot{G} \cap \ddot{R} = \emptyset$ .<sup>4</sup>

Given a multiplicity  $\mu: \ddot{G} \cup \ddot{R} \rightarrow \mathbb{N}^2$  and an isomorphism  $g: R^\mu(y) \rightarrow G^\mu(x)$ , the *adaptive star replacement*  $G[x \overset{\mu}{\parallel}_g r]$  is given by

$$G[x \overset{\mu}{\parallel}_g r] = G^\mu[x \ /_g r^\mu],$$

where  $r^\mu = \langle y, R^\mu \rangle$ .

If  $\mathcal{P}$  is a set of star rules such that  $H = G[x \overset{\mu}{\parallel}_g r]$  for some  $r \in \mathcal{P}$  (for suitable  $\mu$  and  $g$ ), we write  $G \Rightarrow_{\mathcal{P}} H$ .

---

<sup>4</sup>If the latter assumption is not satisfied, we assume implicitly that a suitable isomorphic copy of  $r$  is taken.

Adaptive star replacement thus combines two cloning operations with an ordinary star replacement step:

**Adaptation.** The cloning operations  $G^\mu$  and  $r^\mu$  adapt the star rule to the star  $G(x)$  in the host graph.

**Application.** Then a star replacement step  $G^\mu[x /_g r^\mu]$  applies the adapted rule to the adapted star.

We can now define adaptive star grammars and the graph languages they generate. In fact, an adaptive star grammar is simply a star grammar. However, we use the term adaptive star grammar to indicate that we are interested in the language it generates by adaptive star replacement rather than by ordinary star replacement.

**Definition 4.2 (Adaptive Star Grammar)** An *adaptive star grammar*  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  consists of the same components as an ordinary star grammar. The *adaptive star language generated by  $\Gamma$*  is  $\mathbf{L}(\Gamma) = \ddot{\mathbf{L}}(\Gamma) \cap \mathcal{G}_{S \setminus \dot{S}}$ , where

$$\ddot{\mathbf{L}}(\Gamma) = \{G \in \mathcal{G}_{S \setminus \dot{S}} \mid Z \Rightarrow_{\mathcal{P}}^+ G\}.$$

Thus,  $\mathbf{L}(\Gamma)$  consists of terminal graphs, whereas the graphs in  $\ddot{\mathbf{L}}(\Gamma)$  may still contain multiple nodes (but no nonterminal nodes). Note that we use the notation  $\mathbf{L}(\Gamma)$  rather than  $L(\Gamma)$  if we refer to the language generated by adaptive star replacement rather than by ordinary star replacement.

Note that every adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  can be extended by rules that just implement cloning. For this, and also for later use, let us define some terminology.

The set of *border node types* of stars over  $S$  is given by

$$\mathcal{B}_S = \{\text{in}, \text{out}\} \times (\dot{S} \setminus \ddot{S}) \times \bar{S}.$$

A border node  $z$  incident with an edge  $e$  in a star  $X$  is of type  $(d, l, l')$  if  $l' = \bar{l}_X(e)$ ,  $\dot{l}_X(z) \in \{l, \ddot{l}\}$  and

$$d = \begin{cases} \text{in} & \text{if } s_X(e) = z \\ \text{out} & \text{if } t_X(e) = z. \end{cases}$$

Note that border node types do not distinguish between singular and multiple border nodes.

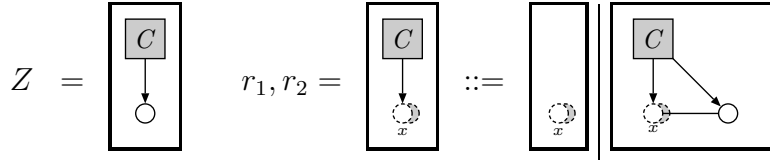


Figure 6: An adaptive star grammar for deriving complete graphs

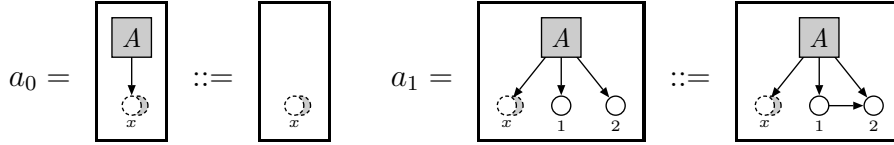


Figure 7: An adaptive star grammar deriving all graphs

For a nonterminal label  $A \in \overset{\circ}{S}$ , the *universal  $A$ -star* (with respect to  $S$ ) is the star  $univ(A)$  with center node labeled  $A$ , containing exactly one multiple border node of type  $t$ , for every  $t \in \mathcal{B}_S$  (and no further nodes).

To define rules that solely implement cloning, let  $\mathcal{P}_c$  be the set of all star rules  $r = \langle x, R \rangle$  such that  $R(x) \cong R \setminus \{x\} \cong univ(A)$  for some  $A \in \overset{\circ}{S}$ . Thus, apart from cloning, the application of  $r$  simply replaces a star by itself. Clearly, for every nonterminal graph  $G$ ,  $G \Rightarrow_{\mathcal{P}_c} G'$  if and only if  $G' = G^\mu$  for some multiplicity  $\mu$ . Thus,  $\Gamma' = \langle S, \mathcal{P} \cup \mathcal{P}_c, Z \rangle$  generates the same language as  $\Gamma$ , and every derivation step  $G \Rightarrow_{\mathcal{P}} H$  can be split into  $G \Rightarrow_{\mathcal{P}_c} G' \Rightarrow_{\mathcal{P}} H$ , where  $H = G'[x / r^\mu]$  for some  $r \in \mathcal{P}$  and a suitable multiplicity  $\mu$ .

**Example 4.3 (Generating All Complete Graphs)** We can now define the language of complete graphs that has been discussed in Examples 2.6 and 2.9. The initial star and the adaptive star rules are shown in Figure 6. Note that, via cloning, we can turn the rules into those depicted schematically in Figure 4.

The rules in Figure 2 are clones  $r_1[x \cdot \overset{0}{i}]$ ,  $r_2[x \cdot \overset{0}{i}]$  of these adaptive star rules (for  $1 \leq i \leq 4$ ) so that the derivation in Figure 3 is a sequence of adaptive star replacements with  $\{r_1, r_2\}$ .

**Example 4.4 (Generating All Graphs)** The adaptive star rules in Figure 7 generate, from an initial star  $Z_A$  that equals the left-hand side of rule  $a_0$ , the class of all unlabeled graphs without loops. (For generating loops, we would need an additional rule, to be obtained from  $a_1$  by identifying nodes

1 and 2, and the edges leading to them from the nonterminal nodes.)

Figure 8 shows the derivation of a graph that uses the clones  $a'_0 = a_0 [x \cdot \frac{0}{3}]$  and  $a'_1 = a_1 [x \cdot \frac{0}{1}]$  of the rules in this grammar. Note that the (singular) nodes of the graph are not generated by the rules, but by cloning the multiple node of the initial star  $Z_A$ .

**Example 4.5 (Generating All Bipartite Graphs)** The adaptive star rules in Figure 9 generate, from an initial star that equals the left-hand side of rule  $b_1$ , the class of all bipartite graphs, where white nodes are only incident with dark nodes, and vice versa. This grammar does not generate counter-parallel edges between white and dark nodes. Such edges could be generated after a simple modification of rule  $b_2$  that is left to the reader.

**Example 4.6 (Generating All Dags)** The adaptive star grammar shown in Figure 10 generates the set of all unlabeled nonempty directed acyclic graphs without parallel edges (*DAGs*). To understand how the grammar works, note that the order in which the  $D$ -labeled nonterminal generates nodes corresponds to a topological sorting. For every node  $v$  added, an  $E$ -labeled nonterminal node is added, with incoming edges from the set  $V$  of nodes generated earlier, and an outgoing edge to  $v$ . This nonterminal is responsible for inserting edges from some of the nodes in  $V$  to  $v$ . Figure 11 shows how a DAG with three nodes (two roots and one leaf) is derived. The rows in the figure show cloning operations, whereas the columns show

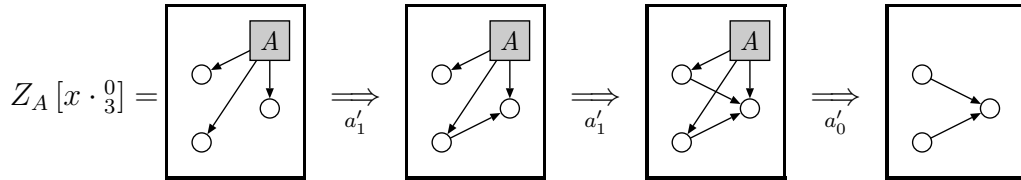


Figure 8: A derivation with the adaptive star rules of Figure 7

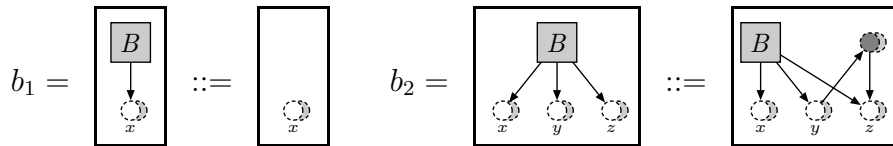


Figure 9: An adaptive star grammar deriving all bipartite graphs

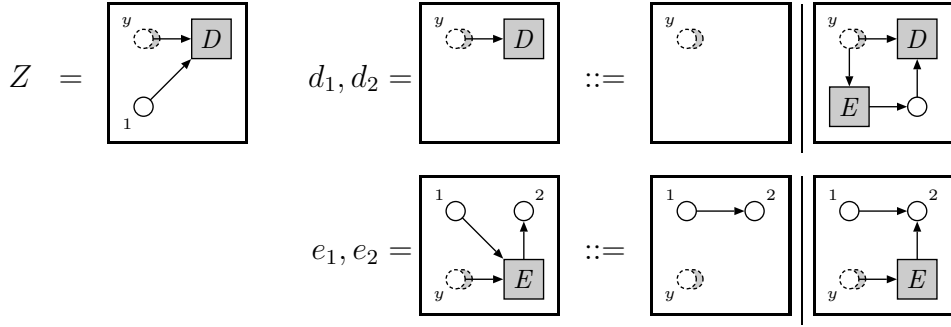


Figure 10: An adaptive star grammar deriving all directed acyclic graphs

applications of (clones of) rules. In this grid, every path of downward or rightward moves from the initial star (in the upper left corner) to the terminal graph (in the lower right corner) corresponds to a derivation of the terminal graph.

Variations of the grammar are easily obtained. If we remove node 1 from  $Z$ , the empty graph can be generated as well. With the initial star  $Z[y \cdot 0]$  instead of  $Z$ , the grammar would derive all rooted DAGs. Finally, if we added an edge from node 1 to the  $E$ -labeled node in the right-hand side of rule  $e_2$ , the grammar would derive DAGs with parallel edges.

In Section 2, we saw that star replacement can simulate hyperedge replacement (see Theorem 2.8). Adaptive star replacement is quite easily seen to be able to simulate node replacement under the boundary condition. In fact, for this, a very special type of adaptive star grammars suffices.

**Definition 4.7 (NR-like Adaptive Star Grammar)** Let  $S$  be an labeling alphabet. An adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  is *NR-like* if the following hold for every rule  $\langle x, R \rangle \in \mathcal{P}$ :

- $R(x)$  is the universal  $\bar{\ell}_R(x)$ -star,
- all nodes in  $\dot{R} \setminus \dot{R}(x)$  are singular, and
- $R$  contains neither loops nor multiple edges nor edges incident with two border nodes of  $R(x)$ .

The following theorem establishes the equivalence of NR-like adaptive star grammars with the well-studied *boundary edNCE (B-edNCE) grammars*. Here, we assume that the reader is familiar with B-edNCE grammars and use the definitions, terminology, and notation from [10]. We consider only

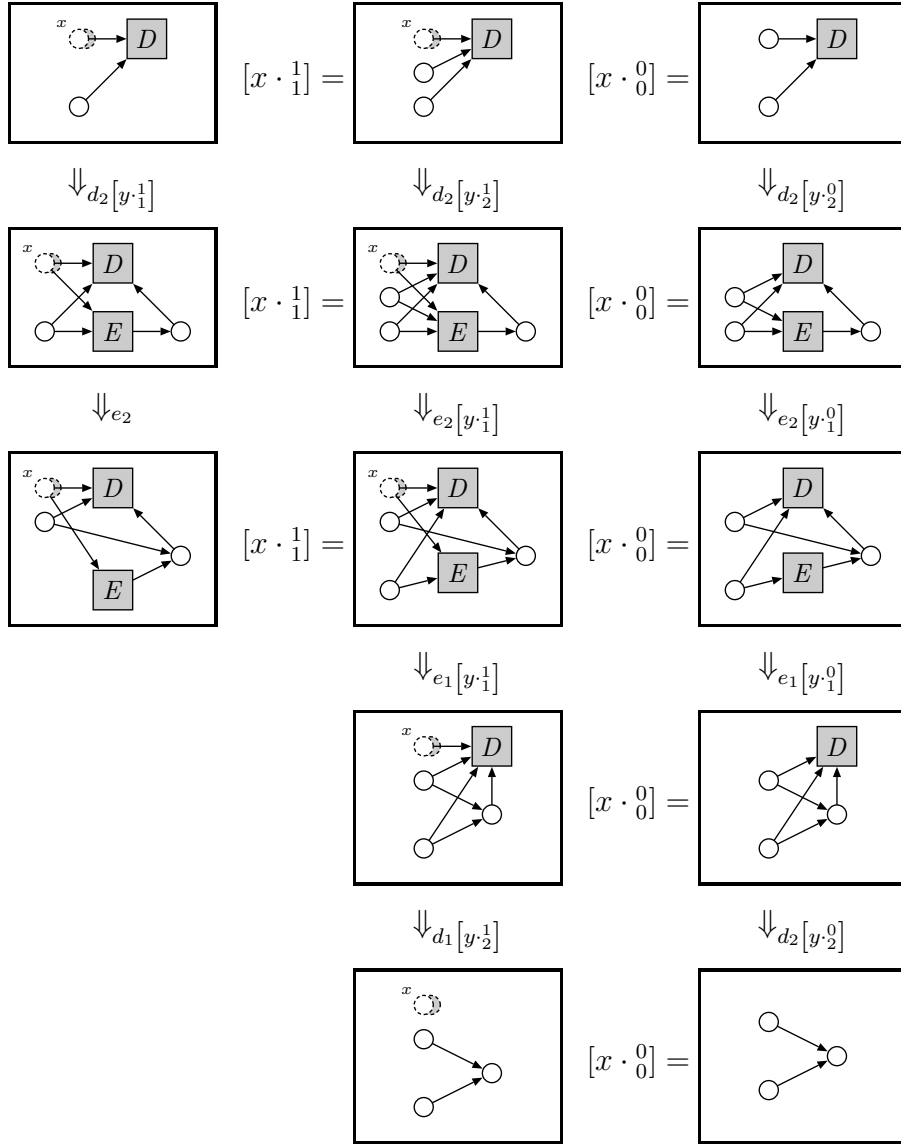


Figure 11: Derivations of a directed acyclic graphs

B-edNCE grammars in which all edge labels are final (i.e., are allowed to occur in terminal graphs), as this does not restrict the generative power.

**Theorem 4.8** A graph language can be generated by an NR-like adaptive star grammar if and only if it can be generated by an B-edNCE grammar.

*Proof* Let us first show that B-edNCE grammars can be simulated by NR-like adaptive star grammars. First of all, it is easy to show that a given B-edNCE grammar can, without affecting the language generated, be turned into one in which the right-hand sides of all rules (and, hence, the sentential forms) satisfy Assumption 2.4. In other words, every nonterminal node is the center of a star.

Now, consider a B-edNCE grammar  $\Gamma$  of this form, with labels in  $S$ , rule set  $\mathcal{P}$ , and initial nonterminal  $A_0$ . Consider a rule  $r = (l \rightarrow (D, C)) \in \mathcal{P}$ , where  $A \in \dot{S}$ , and  $(D, C)$  consists of the replacement graph  $D$  and the set  $C$  of connection instructions<sup>5</sup>. Let  $r'$  be the star rule  $\langle x, \bar{R} \rangle$  given as follows:

- $R \setminus \bar{R}(x) = D$ ,
- $R(x) = \text{univ}(A)$ ,
- for every connection instruction  $(B, a/b, u, d) \in C$ , if  $v$  is the node of type  $(d, B, a)$  in  $R(x)$ , then there is an edge  $e \in \bar{R}$  with

$$(s_R(e), \bar{\ell}_R(e), t_R(e)) = \begin{cases} (v, b, u) & \text{if } d = \text{in} \\ (u, b, v) & \text{otherwise.} \end{cases}$$

Now, the adaptive star grammar  $\Gamma' = \langle S, \mathcal{P}', Z \rangle$ , where  $\mathcal{P}' = \{r' \mid r \in \mathcal{P}\}$  and  $Z$  consists of an isolated node labeled with  $A_0$ , generates the same language as  $\Gamma$ . We leave the explicit verification to the reader.

For the other direction of the proof, it is not difficult to reverse the construction above. This yields a B-edNCE grammar with generalized connection instructions that may reverse the direction of edges. However, the latter is known to have no effect on the generative power of B-edNCE grammars.  $\square$

It is well-known that the language of all graphs (see Example 4.4) cannot be generated by confluent edNCE grammars. Thus, the following corollary is obtained.

**Corollary 4.9** Adaptive star grammars are strictly more powerful than B-edNCE grammars.

We conjecture that adaptive star grammars are even strictly more powerful than arbitrary confluent edNCE grammars, but this remains an open

---

<sup>5</sup>Connection instructions are of the form  $(B, a/b, u, d)$ , where  $B \in \dot{S}$ ,  $\beta, \gamma \in \bar{S}$ ,  $u \in \dot{D}$ , and  $d \in \{\text{in}, \text{out}\}$ . Roughly speaking, the semantics of such an instruction is: *For every in-edge (resp. out-edge) labeled  $a$  between the replaced node and a node  $v$  labeled  $B$  adjacent with it, add an edge labeled  $b$  (with the same direction) between  $u$  and  $v$ .*

question.

## 5 Early and Late Cloning

In this section, we will study an important aspect regarding derivations, namely the interplay between cloning and rule application. Cloning can be performed early, where cloning on the host graph is done as early as possible. This is a consequence of the following lemma, which shows that cloning distributes over star replacement.

**Lemma 5.1 (Cloning Distributes)** Consider a graph  $H = G[x /_g r]$  and a multiplicity  $\mu: \ddot{H} \rightarrow \mathbb{N}^2$ . Then  $G^\mu[x /_{g'} r^\mu] \cong H^\mu$  for some isomorphism  $g'$ .

*Proof* Thanks to Lemmas 3.4 and 3.5, it suffices to consider a multiplicity  $\mu: \{z\} \rightarrow \{(0, 0), (1, 1), (2, 0)\}$ , for some  $z \in \ddot{H}$ . Without loss of generality, let  $H = G[x / r]$ .

Let  $r = \langle y, R \rangle$ . If  $z$  is not a border node of  $G(x)$ , the statement is obviously correct. Thus, assume that  $z$  is a border node of  $G(x)$  (and, thus, of  $R(y)$ ). We distinguish the three cases for  $\mu(z)$ .

If  $\mu(z) = (0, 0)$ , then  $H^\mu = H \setminus \{z\}$ ,  $G^\mu = G \setminus \{z\}$ , and  $R^\mu = R \setminus \{z\}$  by Fact 3.3(1), so that  $G^\mu[x / r^\mu] \cong H^\mu$ .

If  $\mu(z) = (2, 0)$ ,  $H^\mu$  consists of  $H$ , with a copy of  $H(z)$  glued to the border nodes of  $H(z)$ . Since  $G^\mu$  and  $R^\mu$  also have copies of  $G(z)$  and  $R(z)$  glued to the border nodes of  $G(z)$  and  $R(z)$ , resp., we have  $G^\mu[x / r^\mu] \cong H^\mu$ .

For  $\mu(z) = (1, 1)$ , the situation is analogous, the only difference being that the clone of  $z$  is singular in this case.  $\square$

As a corollary, namely by applying Lemma 5.1 to the last step of a derivation, we obtain the following corollary, stating that  $\ddot{\mathbf{L}}(\Gamma)$  is closed under cloning. In particular,  $\mathbf{L}(\Gamma)$  is equal to the set of all singular clones of graphs in  $\ddot{\mathbf{L}}(\Gamma)$ . To express this fact, and also for later use, let us introduce a notation for taking the closure under cloning: for a graph  $G$ ,

$$cl G = \{G^\mu \mid \mu \text{ a multiplicity}\},$$

and for a set  $L$  of graphs,  $cl L = \bigcup_{G \in L} cl G$ . We shall also use this notation for (sets of) star rules:  $cl r = \{r^\mu \mid \mu \text{ a multiplicity}\}$  for every star rule  $r$ , and  $cl \mathcal{R} = \bigcup_{r \in \mathcal{R}} cl r$  for every set  $\mathcal{R}$  of star rules.

**Corollary 5.2** For every adaptive star grammar  $\Gamma$ ,  $\ddot{\mathbf{L}}(\Gamma) = cl \ddot{\mathbf{L}}(\Gamma)$ . In particular,  $\mathbf{L}(\Gamma) = cl \ddot{\mathbf{L}}(\Gamma) \cap \mathcal{G}_{\mathbf{S} \setminus \mathfrak{s}}$ .

The more important consequence of Lemma 5.1 is that derivation steps  $G \Rightarrow_{\mathcal{P}} H$  may be assumed to clone “early”, replacing all multiple nodes by the desired number of singular clones before a rule is applied. Let us first define the notion of early cloning formally.

**Definition 5.3 (Early Cloning)** Let  $\mathcal{P}$  be a set of star rules. An adaptive star replacement step  $G \Rightarrow_{\mathcal{P}} H$  uses early cloning if  $H \in \mathcal{G}_{S \setminus \mathfrak{S}}$ . Such an adaptive star replacement step is denoted by  $G \xRightarrow{\mathcal{P}} H$ .

Note that, as the graphs  $G_0, \dots, G_n$  in a derivation  $Z = G_0 \xRightarrow{\mathcal{P}} G_1 \xRightarrow{\mathcal{P}} \dots \xRightarrow{\mathcal{P}} G_n$  are singular, the star replacements (except the first one) never adapt the host graph but only the rule, i.e., they are of the form  $G_i = G_{i-1}[x /_g r^\mu]$ . (To be precise, one must add that this holds for  $i = 1$  only if  $Z$  is singular. However, this is of course the case if  $Z$  consists of an isolated nonterminal node; see the remark after Definition 2.7.)

**Example 5.4 (Early Cloning)** The derivation in Figure 8 uses early cloning. In Figure 11, the derivation that corresponds to the rightmost column (i.e., turns  $x$  into two singular clones in the first step) uses early cloning.

Using Lemma 5.1, we can easily prove that early cloning is an adequate strategy as it yields all graphs of an adaptive star replacement language.

**Theorem 5.5 (Early Cloning is Adequate)** For every adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$ ,

$$\mathbf{L}(\Gamma) = \{G \in \mathcal{G}_{S \setminus \mathfrak{S}} \mid Z \xRightarrow{\mathcal{P}}^+ G\}.$$

*Proof* Consider a graph  $H \in \mathcal{G}_S$  and a multiplicity  $\mu$  such that  $H^\mu \in \mathcal{G}_{S \setminus \mathfrak{S}}$ . Since  $\mathbf{L}(\Gamma) \subseteq \mathcal{G}_{S \setminus \mathfrak{S}}$ , it suffices to show that  $Z \Rightarrow_{\mathcal{P}}^+ H$  implies  $Z \xRightarrow{\mathcal{P}}^+ H^\mu$ . We proceed by induction on the length  $n$  of the derivation  $Z \Rightarrow_{\mathcal{P}}^n H$ . Thus, let  $Z \Rightarrow_{\mathcal{P}}^{n-1} G \Rightarrow_{\mathcal{P}} H$ , where  $H = G^\nu[x / r^\nu]$ . By Lemma 5.1, we have  $H^\mu = G^\eta[x / r^\eta]$ , where  $\eta = \mu \circ \nu$ . It follows that  $G \xRightarrow{\mathcal{P}} H^\mu$  and  $G^\eta \xRightarrow{\mathcal{P}} H^\mu$ . The former proves the claim for  $n = 1$ , whereas the latter proves it for  $n > 1$ , because the induction hypothesis yields  $Z \xRightarrow{\mathcal{P}}^n G^\eta$ .  $\square$

Derivation steps  $Z \xRightarrow{\mathcal{P}} H$  using early cloning have the advantage that we only have to work with host graphs and cloned star rules that are singular. However, a drawback of this strategy is that a single step  $G \xRightarrow{\mathcal{P}} H$  requires

to choose from an infinite set of multiplicities  $\mu$  if the applied star rule introduces new multiple nodes. In other words,  $\xrightarrow{e}_{\mathcal{P}}$  is not finitely branching. Another drawback is that an early choice restricts the subsequently possible derivations earlier than necessary.

It is, therefore, sometimes desirable to postpone cloning as much as possible. To develop such a notion of *late cloning*, we study the interplay of  $\mu$  and  $g$  in an adaptive star replacement  $G[x \mu //_g r]$  in more detail. Intuitively, to apply a star rule  $r = \langle y, R \rangle$  to a nonterminal node  $x$  in  $G$ , we have to establish a relation between the border nodes of  $G(x)$  and  $R(y)$  that respects border node types. For border nodes  $u$  and  $v$  of  $R(y)$  and  $G(x)$ , resp., being in this relation means that  $g$  will map a clone of  $u$  to a clone of  $v$ . Multiple border nodes in  $R(y)$  and  $G(x)$  may correspond to any number of border nodes of  $G(x)$  and  $R(y)$ , resp., whereas singular ones must correspond to exactly one. This leads to the following formal definition.

**Definition 5.6 (Correspondence)** Let  $G \in \mathcal{G}_{\mathbf{S}}$ ,  $x \in \overset{\circ}{G}$ , and let  $r = \langle y, R \rangle$  be a star rule with  $\dot{\ell}_R(y) = \dot{\ell}_G(x)$ . A *correspondence of  $r$  and  $G$  at  $x$*  is a binary relation  $C \subseteq (\dot{R}(y) \setminus \{y\}) \times (\dot{G}(x) \setminus \{x\})$  such that the following hold.

- (1) For all  $(u, v) \in C$ , the border node types of  $u$  and  $v$  in  $R(y)$  and  $G(x)$ , resp., are equal.
- (2) Every singular border node of  $R(y)$  and  $G(x)$  occurs in exactly one pair in  $C$  (as the first or second component, resp.).

Given a correspondence  $C$  of  $r$  and  $G$  at  $x$  as in the definition, we can construct a multiplicity  $\mu_C: \ddot{G}(x) \cup \ddot{R}(y) \rightarrow \mathbb{N}^2$  and an isomorphism  $g_C: R^{\mu_C}(y) \rightarrow G^{\mu_C}(x)$ , as follows.

- Let  $M = C \cap (\ddot{R} \times \ddot{G})$  be the set of all pairs of multiple nodes in  $C$ , and  $\overline{M} = C \setminus M$ . Then

$$\mu_C(z) = (|\{(u, v) \in M \mid u = z \text{ or } v = z\}|, |\{(u, v) \in \overline{M} \mid u = z \text{ or } v = z\}|),$$

for all  $z \in \ddot{R}(y) \cup \ddot{G}(x)$ .

- By construction,  $R^{\mu_C}(y)$  contains a clone of  $u$  for all  $(u, v) \in C$ . Let us denote this clone by  $u_v$  (where  $u_v = u$  if  $u$  is singular). Similarly,  $G^{\mu_C}(x)$  contains a clone  $v_u$  of  $v$  for all  $(u, v) \in C$ . By Definition 5.6(1), the types of  $u_v$  and  $v_u$  are equal. Moreover, either both are multiple nodes, or both are singular nodes. Thus, we obtain an isomorphism  $g_C: R^{\mu_C}(y) \rightarrow G^{\mu_C}(x)$  by defining  $g_C(u_v) = v_u$  for all  $(u, v) \in C$ .

We can now define late cloning in adaptive star replacement.

**Definition 5.7 (Late Cloning)** An adaptive star replacement of the form  $G[x \overset{\mu_C}{\parallel}_{g_C} r]$ , where  $C$  is a correspondence of  $r$  and  $G$  at  $x$  is said to use *late cloning* and is denoted by  $G[x \parallel_C r]$ . Given a set  $\mathcal{P}$  of star rules, an adaptive star replacement step  $G \Rightarrow_{\mathcal{P}} H$  uses late cloning if the adaptive star replacement performed during this step does. Such steps are denoted by  $G \xRightarrow{\mathcal{P}} H$ .

The reader should note that, in contrast to early cloning, late cloning results in a finitely branching derivation relation, because there are only finitely many correspondences for given  $G$ ,  $x$ , and  $r$ .

We can now show that every adaptive star replacement can be decomposed into an adaptive star replacement using late cloning followed by a cloning step.

**Lemma 5.8** For every adaptive star replacement  $H = G[x \overset{\mu}{\parallel}_g r]$ , there exist a correspondence  $C$  of  $r$  and  $G$  at  $x$  and a multiplicity  $\nu$  such that  $H = G[x \parallel_C r]^{\nu}$ .

*Proof* Consider  $H = G[x \overset{\mu}{\parallel}_g r] = G^{\mu}[x / r^{\mu}]$ , where  $r = \langle y, R \rangle$ , and assume without loss of generality that  $G \cap \dot{R} = \emptyset$ . For  $K \in \{G, R\}$  and  $z \in \dot{K}^{\mu}$ , let  $orig_K(z)$  denote the node in  $K$  of which  $z$  is a clone (where the clone of a singular node is the node itself). Now, define

$$C = \{(orig_R(z), orig_G(z)) \mid z \in \dot{G}^{\mu}(x) \setminus \{x\}\}.$$

(This definition of  $C$  makes sense because  $\dot{G}^{\mu}(x) \setminus \{x\} = \dot{R}^{\mu}(y) \setminus \{y\}$ , thanks to the assumption that the isomorphism  $g$  in the star replacement  $G^{\mu}[x / r^{\mu}] = G^{\mu}[x / r^{\mu}]$  is the identity. Thus, in particular,  $orig_R(z)$  is defined.)

Now, let  $G_0 = G^{\mu_C}$  and  $r_0 = \langle y, R_0 \rangle = r^{\mu_C}$ , assuming without loss of generality that  $\dot{G}_0 \cap \dot{R}_0 = \emptyset$ . We construct  $\nu: \dot{G}_0 \cup \dot{R}_0 \rightarrow \mathbb{N}^2$ , as follows. First, consider a multiple node  $z' \in \dot{G}_0$ . If  $z' \notin \dot{G}_0(x)$ , let  $\nu(z') = \mu(z')$ . Otherwise, using the same notation as in the definition of  $g_C$ ,  $z'$  is of the form  $orig_G(z)_{orig_R(z)}$  for some  $z \in \dot{G}^{\mu}(x) \setminus \{x\}$ . Now, let  $\nu(z')$  reflect how many different (multiple or singular) nodes  $z \in \dot{G}^{\mu}(x)$  give rise to  $z' = orig_G(z)_{orig_R(z)}$ : With  $M = \dot{G}^{\mu}(x)$  and  $\overline{M} = \dot{G}^{\mu}(x) \setminus (\dot{G}^{\mu}(x) \cup \{x\})$ ,

$$\nu(z') = (|\{z \in M \mid z' = orig_G(z)_{orig_R(z)}\}|, |\{z \in \overline{M} \mid z' = orig_R(z)_{orig_R(z)}\}|).$$

For  $z' \in \ddot{R}_0$ , the construction is analogous. If  $z' \notin \dot{R}_0(y)$ , we define  $\nu(z') = \mu(z')$ . Otherwise, with  $M = \ddot{R}^\mu(y)$  and  $\overline{M} = \dot{R}^\mu(y) \setminus (\ddot{R}^\mu(y) \cup \{y\})$ ,

$$\nu(z') = (|\{z \in M \mid z' = \text{orig}_R(z)_{\text{orig}_G(z)}\}|, |\{z \in \overline{M} \mid z' = \text{orig}_R(z)_{\text{orig}_G(z)}\}|).$$

The reader may now easily check that  $\nu \circ \mu_C = \mu$ . Consequently, using Lemma 5.1, we obtain

$$H = G^\mu[x / r^\mu] = G^{\nu \circ \mu_C}[x / r^{\nu \circ \mu_C}] = G^{\mu_C}[x / r^{\mu_C}]^\nu = G[x //_C r]^\nu,$$

which completes the proof.  $\square$

We can now prove that late cloning is adequate.

**Theorem 5.9 (Late Cloning is Adequate)** For every adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$ ,

$$\ddot{\mathbf{L}}(\Gamma) = \text{cl}\{G \in \mathcal{G}_S \mid Z \xrightarrow{\ell}^+_{\mathcal{P}} G\}.$$

*Proof* The direction ‘ $\supseteq$ ’ follows directly from Corollary 5.2. It remains to be shown that  $G_0 \Rightarrow_{\mathcal{P}}^* G$  implies  $G_0 \xrightarrow{\ell}^*_{\mathcal{P}} H$ , where  $G = H^\mu$  for some multiplicity  $\mu$ . We proceed by induction, using the previous lemma and the fact that, for all graphs  $K, K'$  and all multiplicities  $\mu$ ,  $K^\mu \Rightarrow_{\mathcal{P}} K'$  implies  $K \Rightarrow_{\mathcal{P}} K'$ . The latter follows directly from the definition of adaptive star replacement, because  $K' = K^{\mu \circ \nu}[x / r^\mu]$  can be rewritten as  $K' = K^\eta[x / r^\eta]$ , where  $\eta$  equals  $\mu \circ \nu$  on  $K$  and  $\mu$  on  $r$ .

For derivations of length 0, there is nothing to show. Thus, consider some derivation  $G_0 \Rightarrow_{\mathcal{P}} G_1 \Rightarrow_{\mathcal{P}}^n G$ . Lemma 5.8 yields a step  $G_0 \xrightarrow{\ell}^n_{\mathcal{P}} H_1$  such that  $G_1 = H_1^\nu$  for some multiplicity  $\nu$ . By the fact mentioned above, we have  $H_1 \Rightarrow_{\mathcal{P}}^n G$ . Thus, the induction hypothesis yields a graph  $H$  and a multiplicity  $\mu$  such that  $H_1 \xrightarrow{\ell}^*_{\mathcal{P}} H$  (and thus  $G_0 \xrightarrow{\ell}^*_{\mathcal{P}} H$ ) and  $G = H^\mu$ , as required.  $\square$

## 6 Parsing

If adaptive star grammars shall be practically used, algorithms for parsing graphs according to a given adaptive star grammar are needed. In this section, we present such an algorithm. For this purpose, we consider a special case called simple adaptive star grammars first. In the following, let us call an edge  $e$  in a graph  $G$  terminal if it is not incident with a nonterminal node; otherwise,  $e$  is nonterminal.

**Definition 6.1 (Simple Adaptive Star Grammar)** A star rule  $\langle y, R \rangle$  is *empty* if  $R \setminus \{y\}$  is the discrete graph consisting of the border nodes of  $R(y)$ . A set  $\mathcal{P}$  of star rules is *simple* if it does not contain any empty rule. An adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  is simple if  $Z$  consists of an isolated nonterminal node, and  $\mathcal{P}$  is simple.

Thanks to Theorem 5.5, we may restrict our attention to derivations using early cloning. Thus, neither the host graphs nor the rule clones applied in derivation steps contain multiple nodes.

**Lemma 6.2** There is an algorithm that decides whether  $G \xRightarrow{\mathcal{P}}^* G'$  for every finite set  $\mathcal{P}$  of simple star rules and all graphs  $G, G' \in \mathcal{G}_{\mathbf{S} \setminus \mathfrak{S}}$ .

*Proof* Let the *size* of a graph  $H$  be  $\tau(H) = |\dot{H}| + |\{e \in \bar{H} \mid e \text{ terminal}\}|$ . Since a star in a graph  $H$  cannot have more than  $|\dot{H} \setminus \bar{H}|$  edges, there are only finitely many graphs  $H'$  (over a finite alphabet, and up to isomorphism) such that  $\tau(H') \leq \tau(H)$ .

Moreover, since  $\mathcal{P}$  is simple, each derivation step  $H \xRightarrow{\mathcal{P}} H'$  removes a nonterminal node, but adds at least one other node or a terminal edge, i.e.,  $\tau(H) \leq \tau(H')$ . Thus, we can decide whether  $G \xRightarrow{\mathcal{P}}^* G'$  by enumerating the (finitely many) derivations  $G = G_0 \xRightarrow{\mathcal{P}} G_1 \xRightarrow{\mathcal{P}} \cdots \xRightarrow{\mathcal{P}} G_n$ , where  $\tau(G_i) \leq \tau(G')$  and  $G_i \not\cong G_j$  for  $0 \leq i < j \leq n$ , and checking whether  $G_n \cong G'$ .  $\square$

Next, we drop the restriction to simple adaptive star grammars. We will use the following lemma:

**Lemma 6.3** There is an algorithm that decides whether  $G \xRightarrow{\mathcal{P}}^* G \setminus \{x\}$  for every finite set  $\mathcal{P}$  of star rules and all stars  $G \in \mathcal{G}_{\mathbf{S} \setminus \mathfrak{S}}$ , where  $x$  is the center node of  $G$ .

*Proof* The existence of a derivation  $G \xRightarrow{\mathcal{P}}^* G \setminus \{x\}$  requires that no step adds a singular node or a terminal edge. Hence, the number of singular nodes remains constant in the derivation, and its graphs do not contain terminal edges.

Let  $\mathcal{P}_n$  be the (finite) subset of all rules  $\langle y, R \rangle \in \dot{\mathcal{P}}$  such that  $|\dot{R}(y)| \leq n$  and  $R \setminus \{y\}$  contains neither terminal edges nor singular nodes that are not border nodes of  $R(y)$ . Obviously,  $G \xRightarrow{\mathcal{P}}^* G \setminus \{x\}$  iff  $G \Rightarrow_{\mathcal{P}_n}^* G \setminus \{x\}$  where  $n = |\dot{G}|$ .

Now,  $G \Rightarrow_{\mathcal{P}_n}^* G \setminus \{x\}$  is equivalent to the (decidable) question whether an appropriately constructed context-free Chomsky grammar  $G'$  generates the empty string. To see this, construct  $G'$  by using as nonterminal node labels the set of all isomorphism classes  $R(y)$  such that  $R(y)$  is a star occurring in one of the rules in  $\mathcal{P}_n$ . Now, let  $G'$  contain the rule  $[X] \rightarrow [X_1] \cdots [X_k]$  if  $\mathcal{P}_n$  contains a rule  $\langle y, R \rangle$ , where  $X_1, \dots, X_k$  are the stars occurring in  $R$ . It should be clear that  $G'$  generates the empty string if and only if there is a derivation  $G \Rightarrow_{\mathcal{P}_n}^* G \setminus \{x\}$ .  $\square$

Using the two preceding results, we can now show that adaptive star grammars are parseable.

**Theorem 6.4 (Membership is Decidable)** The (uniform) membership problem is decidable for adaptive star grammars, i.e., there is an algorithm that, given an adaptive star grammar  $\Gamma$  and a graph  $G$  as input, decides whether  $G \in \mathbf{L}(\Gamma)$ .

*Proof* For  $\Gamma = \langle S, \mathcal{P}, Z \rangle$ , we construct a new set  $\mathcal{P}'$  of star rules by the following iterative procedure. Initially,  $\mathcal{P}'$  is the (finite) set of all clones  $\langle y, R \rangle$  of rules in  $\mathcal{P}$  such that  $R \in \mathcal{G}_{\dot{S} \setminus \dot{S}}$  and  $|\dot{R}(y)| \leq |\dot{G}| + 1$ . Now, if  $\mathcal{P}'$  contains a rule  $\langle y, R \rangle$  such that there occurs a star with center node  $x$  in  $R$ , then the rule  $\langle y, R \setminus \{x\} \rangle$  is added to  $\mathcal{P}'$  provided that  $R(x) \xrightarrow{e}_{\mathcal{P}}^* R(x) \setminus \{x\}$  (which can be decided by Lemma 6.3). This process is repeated until no new rule can be added to  $\mathcal{P}'$ . Finally, all empty rules are removed from  $\mathcal{P}'$ . Obviously, for every nonempty graph  $G$ , we have  $G \in \mathbf{L}(\Gamma)$  iff  $Z \Rightarrow_{\mathcal{P}'}^* G$ . The result follows by Lemma 6.2 if  $G$  is not empty (as  $\mathcal{P}'$  is simple), and from Lemma 6.3 otherwise.  $\square$

## 7 Associativity and Confluence

From an intuitive point of view, adaptive star replacement has much of the characteristics of a context-free notion of rewriting. In this section, we discuss some of these characteristics in a formal manner. In the literature, there are at least two ways in which one can try to capture the essence of context freeness. To describe them, suppose we are interested in objects belonging to some domain  $D$ .

Context-freeness can be defined relative to a set  $OP$  of operations on  $D$ , each such operation being a function  $\omega: D^n \rightarrow D$  for some  $n \in \mathbb{N}$ .<sup>6</sup> To explain

---

<sup>6</sup>By convention,  $\omega$  is a constant if  $n = 0$ .

this notion of context-freeness, we need to recall some standard terminology. A *ranked alphabet* is a finite set  $\Sigma$  of symbols  $\sigma$ , each having a *rank*  $rk(\sigma) \in \mathbb{N}$ . A *tree over*  $\Sigma$  is a formal expression  $\sigma(t_1, \dots, t_n)$ , where  $\sigma \in \Sigma$  is a symbol of rank  $n$  and  $t_1, \dots, t_n$  are, recursively, trees.<sup>7</sup> The set of all these trees is denoted by  $T_\Sigma$ . A  $\Sigma$ -*algebra*  $\mathcal{A}$  over  $OP$  associates with every symbol  $\sigma \in \Sigma$  an operation  $\sigma_{\mathcal{A}} \in OP$  of arity  $rk(\sigma)$ . The *evaluation* of a tree  $t = \sigma(t_1, \dots, t_k)$  in  $T_\Sigma$  with respect to  $\mathcal{A}$  yields  $eval_{\mathcal{A}}(t) = \sigma_{\mathcal{A}}(eval_{\mathcal{A}}(t_1), \dots, eval_{\mathcal{A}}(t_k))$ .

A *regular tree grammar* is a system  $g = \langle N, \Sigma, P, A_0 \rangle$  consisting of disjoint ranked alphabets  $N$  (the nonterminals) and  $\Sigma$  (the terminals), a set  $P$  of rules, and an initial nonterminal  $A_0 \in N$ . The symbols in  $\Sigma$  can have arbitrary ranks, whereas nonterminals are required to have the rank 0. All rules in  $P$  are of the form  $A \rightarrow t$ , where  $A \in N$  and  $t \in T_\Sigma$ ; the language  $L(g)$  generated by  $g$  is  $L(g) = \{t \in T_\Sigma \mid A_0 \rightarrow_P^* t\}$ , where  $\rightarrow_P$  denotes the term rewrite relation given by  $P$  (i.e., we repeatedly replace some occurrence of a nonterminal  $A$  by a tree  $t$  such that  $A \rightarrow t$  is in  $P$ ). Now,  $L \subseteq D$  is said to be *OP-context-free* if  $L = eval_{\mathcal{A}}(L(g)) = \{eval_{\mathcal{A}}(t) \mid t \in L(g)\}$  for some regular tree grammar  $g$  and some  $\Sigma$ -algebra  $\mathcal{A}$  over  $OP$ .

This view goes back to the seminal paper by Mezei and Wright [15] on context-free sets in arbitrary algebras. To see how, e.g., context-free string grammars fit into it, let  $OP$  be the set *CONC* containing the  $n$ -ary concatenation of strings, for all  $n \in \mathbb{N}$ , and, furthermore, all strings of length 1 as constants. Now, turn a context-free grammar  $G = \langle N, T, P, A_0 \rangle$  into the regular tree grammar  $g = \langle N, \Sigma, P', A_0 \rangle$  given as follows. The set  $P'$  of rules consists of all  $A \rightarrow \circ_n(a_1, \dots, a_n)$  such that  $A \rightarrow a_1 \cdots a_n$  is in  $P$ , where  $a_1, \dots, a_n \in N \cup T$ .  $\Sigma$  contains all elements of  $T$ , considered as symbols of rank 0, and all  $\circ_n$  (of rank  $n$ ) occurring in the rules. Now, let  $\mathcal{A}$  be the  $\Sigma$ -algebra over *CONC* such that  $a_{\mathcal{A}} = a$  for all  $a \in T$ , and  $(\circ_n)_{\mathcal{A}}$  is  $n$ -ary string concatenation. Clearly,  $eval_{\mathcal{A}}(L(g)) = L(G)$ , which means that every context-free string language is *CONC*-context-free.

In [2], Courcelle develops an axiomatic view of context-freeness. In contrast to the setting discussed above, it starts from an abstract notion of sequential rewriting by means of a substitution operator, thus matching the intuitive notion of a grammatical formalism. For this, each object  $O$  is assumed to come with a finite sequence  $\alpha_O = A_1 \cdots A_n$  of nonterminal labels. The number  $n$  is the *rank* of  $O$ , denoted by  $rk(O)$ ; an object of rank 0 is terminal. One may think of  $\alpha_O$  as the multiset of labels of the nonterminals in  $O$ ,

---

<sup>7</sup>Note that the recursion stops when  $n = 0$ .

which is ordered as a sequence of length  $rk(O)$  in order to be able to address individual nonterminals. (For a context-free Chomsky grammar, this could be the sequence of nonterminals in a string, read from left to right.) Given another object  $O'$  and some  $i \in [n]$ , the substitution operator is required to yield an object  $O'' = O[i \leftarrow O']$  satisfying  $\alpha_{O''} = A_1 \cdots A_{i-1} \alpha_{O'} A_{i+1} \cdots A_{rk(O)}$ . In particular, substitution does not duplicate or delete nonterminals (except the substituted one). This is called the *preservation axiom* in [2].

A grammar based on such a substitution operator would have rules of the form  $A \rightarrow O'$ , where  $A$  is a nonterminal label and  $O'$  is an object. If  $O$  is an object such that  $\alpha_O = A_1 \cdots A_n$ , where  $A_i = A$  for some  $i \in [n]$ , then the rule may be applied to this nonterminal, yielding  $O[i \leftarrow O']$ . Derivations start with an initial nonterminal<sup>8</sup> and terminate if an object of rank 0 is reached.

Roughly speaking, the axioms in [2] state that a context-free notion of rewriting is given by a confluent and associative substitution operator. Here, confluence<sup>9</sup> means that the order in which substitutions of distinct nonterminals in an object are performed is immaterial: given objects  $O, O', O''$  and  $i, j \in [rk(O)]$  with  $j < i$ , we have

$$O[i \leftarrow O'][j \leftarrow O''] = O[j \leftarrow O''][i' \leftarrow O'],$$

where  $i' = i - 1 + rk(O'')$ . Associativity means that the equality

$$O[i \leftarrow O'[j \leftarrow O'']] = O[i \leftarrow O'][j' \leftarrow O'']$$

holds, where  $i \in [rk(O)]$ ,  $j \in [rk(O')]$ , and  $j' = i - 1 + j$ .

Corollary 2.17 and Remark 2.18 of [2] provide a strong link between the axiomatic notion of context-freeness and the algebraic one described at the beginning of this section: we may view an object  $O$  as an operation of arity  $n = rk(O)$  by defining  $O(O_1, \dots, O_n) = O[n \leftarrow O_n] \cdots [1 \leftarrow O_1]$ . If  $OP$  is the set of all these operations, and the substitution operator is both confluent and associative, then a language is context-free in the axiomatic sense if and only if it is  $OP$ -context-free in the algebraic sense.<sup>10</sup>

---

<sup>8</sup>By definition, a nonterminal  $A$  is considered as an object  $O$  with  $\alpha_O = A$ , and the application of a rule  $A \rightarrow O'$  to such an object is defined to yield  $O'$ .

<sup>9</sup>With abstract reduction systems, this property is called “commutativity”.

<sup>10</sup>In the terminology of [15, 2], the  $OP$ -context-free sets are the equational ones. Strictly speaking, the *if* direction of the equivalence result, which is Remark 2.18 of [2], requires a small additional assumption, namely that the set of objects is closed under relabelling of nonterminals.

We now discuss to which extent adaptive star grammars have similar properties. As it turns out, the major difference between the notions considered by Courcelle and our adaptive star grammars lies in the fact that (adaptive) star replacement is nondeterministic, and does, therefore, not qualify as a substitution operator in the sense described above. Whereas, in Courcelle's setting, substitution is required to be a deterministic process, the result of a star replacement  $G[x \ /_g r]$  depends on the choice of  $g$ . Thus, specifying only  $x$  and  $r$  leaves us with a set of possible results. In the case of ordinary star replacement, this does not make a big difference, because star grammars can easily be normalized in such a way that substitution becomes deterministic; see Theorem 2.8. However, in the case of adaptive star replacement,  $G[x \ /_\mu r]$  depends on  $\mu$  as well. Thus, it can give rise to any number of results: no result at all, a unique result, finitely many, or infinitely many. We may restrict ourselves to late cloning, in which case we will not get infinitely many results but, still, there is no finite global bound on the number of results we can get. As this type of nondeterminism is crucial for the generative power of adaptive star grammars, it cannot be removed by normalization.

In other words, adaptive star grammars are not context-free in the axiomatic sense, because the resulting substitution operator would not be a function. For very much the same reason, they are not context-free in the algebraic sense either: adaptive star replacement does not give rise to the required set  $OP$  of operations on graphs, as these operations should be functions mapping (tuples of) graphs to graphs. To show that adaptive star grammars enjoy, nevertheless, properties quite similar to context-freeness, we prove nondeterministic variants of confluence and associativity. From this, we obtain Theorem 7.6—a result similar to the equivalence of axiomatic and algebraic context-freeness mentioned above, but using nondeterministic operations instead of the usual deterministic ones.

We first establish some convenient conventions and terminology similar to the abstract notions described in connection with Courcelle's axiomatic approach.

To be able to define substitution in a reasonable manner, we choose star rules rather than graphs to be the objects substitution works on. For every star rule  $r = \langle x, R \rangle$ , we assume throughout this section that the set  $\mathring{R} \setminus \{x\}$  of nonterminals in the right-hand side of  $r$  is ordered as a sequence  $\alpha_r$ . Thus,  $\alpha_r = x_1 \cdots x_n$ , where  $x_1, \dots, x_n$  are pairwise distinct and  $\{x_1, \dots, x_n\} =$

$\overset{\circ}{R} \setminus \{x\}$ . As in the general setting, we let  $rk(r) = n$ .

**Remark 1** To be entirely formal and, at the same time, avoid the slight deviation from the general setting, where  $\alpha_r$  is a sequence of nonterminal labels, one could consider extended star rules of the form  $r = \langle x, R, x_1 \cdots x_n \rangle$ , where  $\langle x, R \rangle$  is a star rule,  $x_1, \dots, x_n$  are the pairwise distinct elements of  $\overset{\circ}{R} \setminus \{x\}$ , and  $\alpha_r = \dot{\ell}_R(x_1) \cdots \dot{\ell}_R(x_n)$ . However, this would only result in inconvenient notational overhead.

If  $r$  is as above and  $r' = \langle y, R' \rangle$  is another star rule, then the substitution of  $x_i$  by  $r'$  in  $r$  yields

$$r[i \leftarrow r'] = \{ \langle x', R[x_i /_g r'] \rangle \mid R(x_i) \cong_g R'(y), \\ x' \text{ is the image of } x \text{ in } R[x_i /_g r'] \}.$$

The adaptive generalization of this type of substitution is defined as follows:

$$r[i \Leftarrow r'] = \{ \langle x', R[x_i \overset{\mu}{//}_g r'] \rangle \mid \mu \text{ a multiplicity,} \\ R(x_i)^\mu \cong_g R'(x)^\mu, \\ x' \text{ is the image of } x \text{ in } R[x_i \overset{\mu}{//}_g r'] \}.$$

Note that  $r[i \leftarrow r'] = r[i \Leftarrow r'] = \emptyset$  unless  $\dot{\ell}_{R'}(y) = \dot{\ell}_R(x_i)$ . For almost everything that follows, this means that it is unnecessary to restrict substitution explicitly to the case where  $\dot{\ell}_{R'}(y) = \dot{\ell}_R(x_i)$ , because the opposite case has no influence on the reasonings.

The notions of substitution just defined are assumed to be compatible with the order of nonterminal nodes: for  $r'' \in r[i \leftarrow r']$  or  $r'' \in r[i \Leftarrow r']$ , it holds that  $\alpha_{r''} = x_1 \cdots x_{i-1} \alpha_{r'} x_{i+1} \cdots x_n$  (assuming for simplicity that star replacement does not rename the nonterminal nodes). In other words, we arrange everything in such a way that the preservation axiom—adapted to the nondeterministic case—holds.

**Remark 2** An appealing variant of the adaptive substitution  $r[i \Leftarrow r']$  is obtained by restricting it to adaptive star replacements that use late cloning. We may denote this variant by  $r[i \overset{\ell}{\Leftarrow} r']$ . The advantage of  $r[i \overset{\ell}{\Leftarrow} r']$  is that it is always finite. By the adequacy of late cloning, all results of this section remain valid if  $r[i \Leftarrow r']$  is replaced with  $r[i \overset{\ell}{\Leftarrow} r']$ . However, some of the proofs would become much more technical, which is the reason for using  $r[i \Leftarrow r']$  in the following.

Finally, we extend substitution to sets of star rules. For this, we say that a set  $\mathcal{R}$  of star rules is *coherent* if all elements of  $\mathcal{R}$  have the same rank, denoted by  $rk(\mathcal{R})$ . If  $\mathcal{R}$  and  $\mathcal{R}'$  are coherent sets of star rules, we define  $\mathcal{R}[i \leftarrow \mathcal{R}']$  and  $\mathcal{R}[i \Leftarrow \mathcal{R}']$  to be the (coherent) sets of star rules given by

$$\mathcal{R}[i \leftarrow \mathcal{R}'] = \bigcup_{\substack{r \in \mathcal{R} \\ r' \in \mathcal{R}'}} r[i \leftarrow r'] \quad \text{and} \quad \mathcal{R}[i \Leftarrow \mathcal{R}'] = \bigcup_{\substack{r \in \mathcal{R} \\ r' \in \mathcal{R}'}} r[i \Leftarrow r'].$$

It should be noted that  $\{r\}[i \leftarrow \{r'\}] = r[i \leftarrow r']$  and  $\{r\}[i \Leftarrow \{r'\}] = r[i \Leftarrow r']$ . In the following, we will therefore identify a star rule  $r$  with  $\{r\}$  whenever this is appropriate.

We can now state in which sense star replacement is confluent.

**Lemma 7.1** If  $\mathcal{R}, \mathcal{R}', \mathcal{R}''$  are coherent sets of star rules and  $i, j \in [rk(\mathcal{R})]$  with  $i > j$ , then

$$\mathcal{R}[i \leftarrow \mathcal{R}'] [j \leftarrow \mathcal{R}''] = \mathcal{R}[j \leftarrow \mathcal{R}''] [i' \leftarrow \mathcal{R}'],$$

with  $i' = i - 1 + rk(\mathcal{R}'')$ .

*Proof* Clearly, it suffices to prove the statement for star rules instead of sets of star rules. Thus, let  $r = \langle x, R \rangle, r' = \langle y, R' \rangle, r'' = \langle z, R'' \rangle$  be star rules with  $\alpha_r = x_1 \cdots x_n$ , and assume that  $\dot{\ell}_R(x_i) = \dot{\ell}_{R'}(y)$  and  $\dot{\ell}_R(x_j) = \dot{\ell}_{R''}(z)$  since, otherwise, the statement is trivially true. By the definitions of star replacement and the operator ' $\leftarrow$ ', the second component of  $r[i \leftarrow r'] [j \leftarrow r'']$  is the set of all graphs obtained from the disjoint union of  $R, R',$  and  $R''$  by

- identifying  $R(x_i)$  with  $R'(y)$  and  $R(x_j)$  with  $R''(z)$  according to some isomorphisms  $g$  and  $h$ , and
- deleting (the images of)  $y$  and  $z$  together with their incident edges.

Again by the definitions of star replacement and ' $\leftarrow$ ' (and taking the choice of  $i'$  into account), the same holds for  $r[j \leftarrow r''] [i' \leftarrow r']$ , which proves that the sets are equal.  $\square$

As a consequence, we obtain a similar confluence property for adaptive star replacement.

**Theorem 7.2 (Confluence of Adaptive Star Replacement)** If  $\mathcal{R}, \mathcal{R}', \mathcal{R}''$  are coherent sets of star rules and  $i, j \in [rk(\mathcal{R})]$  with  $i > j$ , then

$$\mathcal{R}[i \Leftarrow \mathcal{R}'] [j \Leftarrow \mathcal{R}''] = \mathcal{R}[j \Leftarrow \mathcal{R}''] [i' \Leftarrow \mathcal{R}'],$$

with  $i' = i - 1 + rk(\mathcal{R}'')$ .

*Proof* By the definition of adaptive star replacement, if  $\mathcal{R}_1, \mathcal{R}_2$  are coherent sets of star rules, then  $\mathcal{R}_1[i \leftarrow \mathcal{R}_2] = cl \mathcal{R}_1[i \leftarrow cl \mathcal{R}_2]$ . (Here, we use the convention that  $cl$  takes precedence over the substitution operator.) Consequently, Lemma 7.1 yields

$$\begin{aligned} \mathcal{R}[i \leftarrow \mathcal{R}'][j \leftarrow \mathcal{R}'' ] &= cl \mathcal{R}[i \leftarrow cl \mathcal{R}'][j \leftarrow cl \mathcal{R}'' ] \\ &= cl \mathcal{R}[j \leftarrow cl \mathcal{R}'' ][i' \leftarrow cl \mathcal{R}'] \\ &= \mathcal{R}[j \leftarrow \mathcal{R}'' ][i' \leftarrow \mathcal{R}'], \end{aligned}$$

as claimed.  $\square$

With respect to associativity, the situation is similar.

**Lemma 7.3** If  $\mathcal{R}, \mathcal{R}', \mathcal{R}''$  are coherent sets of star rules and  $i \in [rk(\mathcal{R})]$ ,  $j \in [rk(\mathcal{R}')]$ , then

$$\mathcal{R}[i \leftarrow \mathcal{R}'[j \leftarrow \mathcal{R}'' ]] = \mathcal{R}[i \leftarrow \mathcal{R}'][j' \leftarrow \mathcal{R}'' ],$$

with  $j' = i - 1 + j$ .

*Proof* Again, it suffices to prove the statement for star rules instead of sets of star rules. Thus, let  $r = \langle x, R \rangle, r' = \langle y, R' \rangle, r'' = \langle z, R'' \rangle$  be star rules with  $\alpha_r = x_1 \cdots x_m$  and  $\alpha_{r'} = y_1 \cdots y_n$ , and assume that  $\ell_R(x_i) = \ell_{R'}(y)$  and  $\ell_{R'}(y_j) = \ell_{R''}(z)$  (because, again, the statement is trivially true, otherwise). Then the second component of  $r[i \leftarrow r'[j \leftarrow r'']]$  is the set of all graphs obtained from the disjoint union of  $R, R'$ , and  $R''$  by

- identifying  $R(x_i)$  with  $R'(y)$  and  $R'(y_j)$  with  $R''(z)$  according to some isomorphisms  $g$  and  $h$ , and
- deleting (the images of)  $y$  and  $z$  together with their incident edges.

The same holds for  $r[i \leftarrow r'][j' \leftarrow r'']$ , which proves the lemma.  $\square$

**Theorem 7.4 (Associativity of Adaptive Star Replacement)** If  $\mathcal{R}, \mathcal{R}', \mathcal{R}''$  are coherent sets of star rules and  $i \in [rk(\mathcal{R})]$ ,  $j \in [rk(\mathcal{R}')]$ , then

$$\mathcal{R}[i \leftarrow \mathcal{R}'[j \leftarrow \mathcal{R}'' ]] = \mathcal{R}[i \leftarrow \mathcal{R}'][j' \leftarrow \mathcal{R}'' ],$$

with  $j' = j + i - 1$ .

*Proof* This follows from Lemma 7.3 in the same way as Theorem 7.2 follows from Lemma 7.1.  $\square$

Using Theorems 7.2 and 7.4, we can now show in which sense the language generated by an adaptive star grammar can be obtained by evaluating regular tree languages. For this, we extend the notion of algebras

to the nondeterministic case. Consider a set  $OP$  of nondeterministic operations on  $D$ , i.e., each element of  $OP$  is a function  $\omega: D^n \rightarrow \wp(D)$  for some  $n \in \mathbb{N}$ . We extend such an operation to sets of arguments by defining  $\omega(D_1, \dots, D_n) = \bigcup \{\omega(O_1, \dots, O_n) \mid O_1 \subseteq D_1, \dots, O_n \subseteq D_n\}$ , for all  $D_1, \dots, D_n \subseteq D$ .

Given a ranked alphabet  $\Sigma$ , a *nondeterministic  $\Sigma$ -algebra  $\mathcal{A}$  over  $OP$*  associates a nondeterministic operation  $\sigma_{\mathcal{A}} \in OP$  of arity  $n$  with each  $\sigma \in \Sigma$  of rank  $n$ . For  $t = \sigma(t_1, \dots, t_n) \in T_{\Sigma}$ , we let

$$eval_{\mathcal{A}}(t) = \sigma_{\mathcal{A}}(eval_{\mathcal{A}}(t_1), \dots, eval_{\mathcal{A}}(t_n)).$$

For a set  $T \subseteq T_{\Sigma}$ ,  $eval_{\mathcal{A}}(T) = \bigcup_{t \in T} eval_{\mathcal{A}}(t)$ .

In the following, the objects considered are star rules of rank 0, while the operations on them are arbitrary star rules. For the latter, we identify every star rule  $r$  of rank  $n$  with the (nondeterministic) operation of rank  $n$  which is given by

$$r(r_1, \dots, r_n) = r[1 \Leftarrow r_1] \cdots [n \Leftarrow r_n],$$

for all star rules  $r_1, \dots, r_n$  of rank 0. The set of these operations is denoted by  $SR$ , and the set of all star rules of rank 0 is denoted by  $D_{SR}$ .

A pair  $(g, \mathcal{A})$  consisting of a regular tree grammar  $g = (N, \Sigma, P, A_0)$  and a nondeterministic  $\Sigma$ -algebra  $\mathcal{A}$  over  $SR$  is called a *tree-based adaptive star grammar*. It generates the language  $\mathbf{L}_{\mathcal{A}}(g) = eval_{\mathcal{A}}(L(g))$ .

Before turning to the main result of this section, we prove an easy lemma.

**Lemma 7.5** Let  $\mathcal{A}$  be a nondeterministic  $\Sigma$ -algebra over  $SR$ . For all trees  $t = \sigma(t_1, \dots, t_n) \in T_{\Sigma}$ , it holds that

$$eval_{\mathcal{A}}(t) = \sigma_{\mathcal{A}}[1 \Leftarrow eval_{\mathcal{A}}(t_1)] \cdots [n \Leftarrow eval_{\mathcal{A}}(t_n)].$$

*Proof* By the definition of  $eval_{\mathcal{A}}$ , the equality stated in the lemma holds if, for every star rule  $r$  of rank  $n$  and all  $\mathcal{R}_1, \dots, \mathcal{R}_n \subseteq D_{SR}$ ,

$$r[1 \Leftarrow \mathcal{R}_1] \cdots [n \Leftarrow \mathcal{R}_n] = \bigcup \{r[1 \Leftarrow r_1] \cdots [n \Leftarrow r_n] \mid r_1 \in \mathcal{R}_1, \dots, r_n \in \mathcal{R}_n\}.$$

The latter follows by an easy induction on  $n$ : it is trivial for  $n = 0$ , and the inductive step is given by

$$\begin{aligned} & r[1 \Leftarrow \mathcal{R}_1] \cdots [n+1 \Leftarrow \mathcal{R}_{n+1}] \\ &= (\bigcup \{r[1 \Leftarrow r_1] \cdots [n \Leftarrow r_n] \mid r_1 \in \mathcal{R}_1, \dots, r_n \in \mathcal{R}_n\})[n+1 \Leftarrow \mathcal{R}_{n+1}] \\ &= \bigcup \{r[1 \Leftarrow r_1] \cdots [n+1 \Leftarrow r_{n+1}] \mid r_1 \in \mathcal{R}_1, \dots, r_{n+1} \in \mathcal{R}_{n+1}\}, \end{aligned}$$

as claimed.  $\square$

For the main result of this section, let us call an adaptive star grammar *special* if its initial star is the universal  $A$ -star for some nonterminal node label  $A$ . Special adaptive star grammars have the same generative power than general ones. To see this, consider an adaptive star grammar  $\Gamma$ , whose initial star  $Z$  consists of an isolated node labeled  $B$ . We can turn  $\Gamma$  into a special adaptive star grammar  $\Gamma'$  by adding a new nonterminal node label  $A$ , replacing the initial star by  $univ(A)$ , and adding the star rule  $r = \langle x, R \rangle$ , where  $R$  consists of two isolated nodes  $x, y$  with  $\dot{\ell}_R(x) = A$  and  $\dot{\ell}_R(y) = B$ . In this way, every derivation in  $\Gamma'$  starts with an application of  $r$  to  $univ(A)$ , which yields  $Z$ . Hence,  $\mathbf{L}(\Gamma') = \mathbf{L}(\Gamma)$  and  $\ddot{\mathbf{L}}(\Gamma') = \ddot{\mathbf{L}}(\Gamma)$ .

**Theorem 7.6** For a set  $\mathcal{R}$  of star rules, let  $\mathcal{R}^\ominus = cl\{R \setminus \{x\} \mid \langle x, R \rangle \in \mathcal{R}\}$  be the set of its (cloned) right-hand sides. Then the following are equivalent, for every set  $L$  of graphs:

- (i)  $L = \ddot{\mathbf{L}}(\Gamma)$  for some adaptive star grammar  $\Gamma$ .
- (ii)  $L = \mathbf{L}_{\mathcal{A}}(g)^\ominus$  for a tree-based adaptive star grammar  $(g, \mathcal{A})$ .

*Proof* Consider a special adaptive star grammar  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  on the one hand, and a tree-based adaptive star grammar  $(g, \mathcal{A})$  on the other, where  $g = \langle N, \Sigma, P, A_0 \rangle$ . We say that  $\Gamma$  and  $(g, \mathcal{A})$  *correspond* to each other if the following hold:

- $N = \overset{\circ}{S}$ , where  $A_0$  is the label of the center node of  $Z$ ,
- There is a bijection  $b : \mathcal{P} \rightarrow P$  such that, for every rule  $r = \langle x, R \rangle \in \mathcal{P}$  with  $\alpha_r = x_1 \cdots x_n$ ,  $b(r) = (\dot{\ell}_R(x) \rightarrow \sigma(\dot{\ell}_R(x_1), \dots, \dot{\ell}_R(x_n)))$  for some symbol  $\sigma$  with  $\sigma_{\mathcal{A}} = r$ .

Clearly, for every special adaptive star grammar  $\Gamma$ , a corresponding pair  $(g, \mathcal{A})$  can be constructed, and vice versa. Therefore, it suffices to show that

$$\ddot{\mathbf{L}}(\Gamma) = eval_{\mathcal{A}}(L(g))^\ominus$$

if  $\Gamma$  and  $(g, \mathcal{A})$  correspond to each other.

For this, given a nonterminal label  $A \in \overset{\circ}{S}$ , let  $\mathcal{S}_A$  be the union of all sets  $r[i_1 \Leftarrow r_1] \cdots [i_m \Leftarrow r_m] \subseteq D_{SR}$ , where  $m \in \mathbb{N}$ ,  $r, r_1, \dots, r_m \in \mathcal{P}$ , the label of the left-hand side of  $r$  is  $A$ , and  $i_j \in rk(r[i_1 \Leftarrow r_1] \cdots [i_{j-1} \Leftarrow r_{j-1}])$  for all  $j \in [m]$ .

The reader should easily be able to verify that  $\ddot{\mathbf{L}}(\Gamma) = \mathcal{S}_{A_0}^\ominus$ . Thus, the proof is finished if we can show that  $\mathcal{S}_A = eval_{\mathcal{A}}(L_A(g))$  for all  $A \in \overset{\circ}{S}$ , where  $L_A(g) = \{t \in T_\Sigma \mid A \rightarrow_g^* t\}$ . We consider the inclusions separately.

( $\subseteq$ ) Consider  $\mathcal{R} = r[i_1 \Leftarrow r_1] \cdots [i_m \Leftarrow r_m] \subseteq D_{SR}$ , as in the definition of  $\mathcal{S}_A$ , where  $\mathcal{R} \neq \emptyset$ . Let  $r = \langle x, R \rangle$  with  $\alpha_r = x_1 \cdots x_n$  and  $\dot{\ell}_R(x) = A$ . We proceed by induction on  $m$  to show that there is a tree  $t \in L_A(g)$  such that  $\mathcal{R} = \text{eval}_{\mathcal{A}}(t)$ . Using Theorems 7.2 and 7.4 (and an obvious induction), the expression defining  $\mathcal{R}$  can be rewritten to obtain

$$\begin{aligned} \mathcal{R} &= r[1 \Leftarrow r^1[i_1^1 \Leftarrow r_1^1] \cdots [i_{m_1}^1 \Leftarrow r_{m_1}^1]] \cdots [n \Leftarrow r^n[i_1^n \Leftarrow r_1^n] \cdots [i_{m_n}^n \Leftarrow r_{m_n}^n]] \\ &= r[1 \Leftarrow \mathcal{R}_1] \cdots [n \Leftarrow \mathcal{R}_n], \end{aligned}$$

where  $m = 1 + \sum_{j=1}^n m_j$ . By the assumption that  $\mathcal{R} \neq \emptyset$ , the label of the left-hand side of each  $r^j$  ( $j \in [n]$ ) must coincide with  $\dot{\ell}_R(x_j)$ . Hence, the induction hypothesis applies and yields trees  $t_1 \in L_{\dot{\ell}_R(x_1)}(g), \dots, t_n \in L_{\dot{\ell}_R(x_n)}(g)$  such that  $\mathcal{R}_j = \text{eval}_{\mathcal{A}}(t_j)$  for all  $j \in [n]$ . Moreover, since  $\Gamma$  and  $(g, \mathcal{A})$  correspond to each other,  $P$  contains a rule  $\dot{\ell}_R(x) \rightarrow \sigma(\dot{\ell}_R(x_1), \dots, \dot{\ell}_R(x_n))$  with  $\sigma_{\mathcal{A}} = r$ . Consequently,  $\dot{\ell}_R(x) \rightarrow_P^* \sigma(t_1, \dots, t_n)$  and

$$\begin{aligned} \mathcal{R} &= r[1 \Leftarrow \mathcal{R}_1] \cdots [n \Leftarrow \mathcal{R}_n] \\ &= r[1 \Leftarrow \text{eval}_{\mathcal{A}}(t_1)] \cdots [n \Leftarrow \text{eval}_{\mathcal{A}}(t_n)] \\ &= \text{eval}_{\mathcal{A}}(\sigma(t_1, \dots, t_n)) \quad (\text{see Lemma 7.5}), \end{aligned}$$

as required.

( $\supseteq$ ) This direction is completely analogous to the preceding one, using the reverse arguments together with structural induction on trees  $t \in T_{\Sigma}$  such that  $A \rightarrow_P^* t$ .  $\square$

In the rest of this section, we discuss a natural special case of adaptive star grammars. In the deterministic setting, the evaluation of every tree yields a unique result. In particular, since the emptiness problem is decidable for regular tree grammars, this implies that emptiness is decidable for  $OP$ -context-free sets  $L$  (specified by a regular tree grammar  $g$  and an algebra  $\mathcal{A}$  over  $OP$ ), as  $L = \emptyset$  if and only if  $L(g) = \emptyset$ . Obviously, this equivalence does not carry over to the nondeterministic case, because the evaluation of a tree may yield an empty set of results.

This phenomenon can also be observed at the level of derivations in adaptive star grammars: given a graph  $G$ , a nonterminal node  $y \in \dot{G}$ , and a rule  $r = \langle x, R \rangle$  with  $\dot{\ell}_R(x) = \dot{\ell}_G(y)$ , it is not necessarily the case that  $r$  can be applied to  $x$ , as we may not be able to find a multiplicity  $\mu$  with  $R^\mu(x) = G^\mu(y)$ . We now formulate a syntactically verifiable condition which ensures that derivations cannot get stuck in this way.

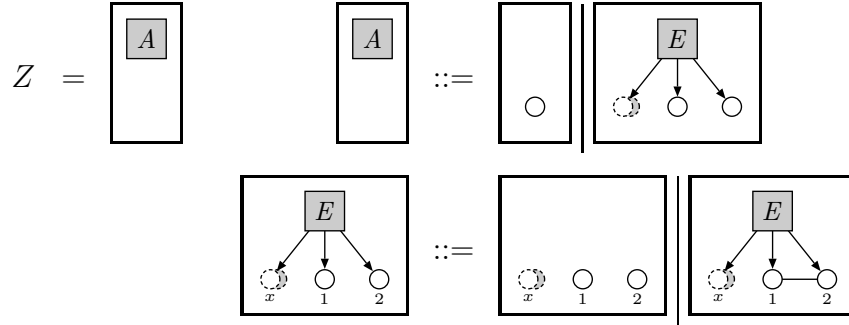


Figure 12: A uniform star grammar deriving all graphs

**Definition 7.7 (Uniformity)**

1. A star  $G$  encompasses another star  $H$ , denoted  $G \succeq H$ , if  $G^\mu \cong H$  for some multiplicity  $\mu$ .
2. Let  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  be an adaptive star grammar, and let  $RHS = \{Z\} \cup \{R \setminus \{x\} \mid \langle x, R \rangle \in \mathcal{P}\}$ .  $\Gamma$  is *uniform* if there are stars  $LHS(l)$ ,  $A \in \mathring{S}$ , such that
  - for all rules  $\langle x, R \rangle \in \mathcal{P}$ ,  $R(x) \cong LHS(\dot{\ell}_R(x))$ ;
  - for all  $R \in RHS$  and  $y \in \mathring{R}$ ,  $LHS(\dot{\ell}_R(y)) \succeq R(y)$ .

Note that  $\succeq$  is transitive. For a graph  $G$ , a star rule  $r = \langle x, R \rangle$  can be applied to  $y \in \mathring{G}$  if  $R(x) \succeq G(y)$ . However, the converse is not true, because the definition of adaptive star replacement allows to clone nodes in both  $R$  and  $G$ .

**Example 7.8** The adaptive star grammar in Example 4.3, generating all complete graphs, is uniform. The one for the set of all unlabelled undirected graphs given in Example 4.4 is not uniform, but the uniform adaptive star grammar shown in Figure 12 generates the same language. Finally, the adaptive star grammar generating all dags from Example 4.6 (see Figure 10) is not uniform either. However, if we replace the rules  $d_i$  by the clones  $d_i [y \cdot \frac{1}{1}]$ , the grammar becomes uniform while still deriving the same language.

We can now show easily that derivations in uniform adaptive star grammars cannot get stuck in the sense mentioned above.

**Theorem 7.9** Let  $\Gamma = \langle S, \mathcal{P}, Z \rangle$  be a uniform adaptive star grammar. For all derivations  $Z \Rightarrow_{\mathcal{P}}^* G$ , if  $y \in \mathring{G}$  and  $r = \langle x, R \rangle \in \mathcal{P}$  satisfy  $\dot{\ell}_R(x) = \dot{\ell}_G(y)$ , then  $R(x) \succeq G(y)$ .

*Proof* Using the notation from Definition 7.7, it follows by a straightforward induction on the length of derivations that there is a  $R_0 \in RHS$  and  $z \in \dot{R}_0$  such that  $R_0(z) \succeq G(y)$ . Further, by uniformity, we have  $R(x) \succeq R_0(z)$ , and thus  $R(x) \succeq R_0(z) \succeq G(y)$ .  $\square$

Finally, we prove the expected result saying that the emptiness problem is decidable for uniform adaptive star grammars.

**Theorem 7.10** The following problem is decidable:

*Input:* A uniform adaptive star grammar  $\Gamma$ .

*Question:* Is  $\mathbf{L}(\Gamma)$  empty?

*Proof* Transform  $\Gamma$  into a tree-based adaptive star grammar  $(g, \mathcal{A})$  by first turning it into a special adaptive star grammar and then constructing the corresponding tree-based adaptive star grammar as described in the proof of Theorem 7.6. Clearly, this is an algorithmic procedure. The following properties of  $(g, \mathcal{A})$  are easily verified, where  $g = (N, \Sigma, P, A_0)$ :

- (i) By construction, every rule in  $P$  has the form  $A \rightarrow \sigma(A_1, \dots, A_n)$  for a symbol  $\sigma \in \Sigma$  and some  $A, A_1, \dots, A_n \in N$  (where  $n = rk(\sigma)$ ).
- (ii) Let  $A \rightarrow \sigma(A_1, \dots, A_n)$  be a rule in  $P$ . If  $\sigma_{\mathcal{A}} = r = \langle x, R \rangle$  with  $\alpha_r = x_1 \cdots x_n$ , then  $\dot{\ell}_R(x) = A$  and  $\dot{\ell}_R(x_i) = A_i$  for all  $i \in [n]$ . (This is an obvious consequence of the fact that  $(g, \mathcal{A})$  corresponds to an adaptive star grammar in the sense defined in the proof of Theorem 7.6.)
- (iii) For every  $A \in N$ , there is a star  $LHS(A)$  such that the following hold for every  $\sigma \in \Sigma$ , where  $\sigma_{\mathcal{A}} = \langle x, R \rangle$ :
  - $R(x) \cong LHS(\dot{\ell}_R(x))$ ;
  - for every  $y \in \dot{R} \setminus \{x\}$ ,  $LHS(\dot{\ell}_R(y)) \succeq R(y)$ .

(This follows from the fact that the special adaptive star grammar which is first constructed from  $\Gamma$  is special except for the form of its initial star.)

Using these properties, we prove the following claim:

For every  $t \in T_{\Sigma}$  such that  $A \rightarrow_P^* t$  for some  $A \in N$ ,  $eval_{\mathcal{A}}(t)$  contains a rule  $r = \langle x, R \rangle$  such that  $R(x) \cong LHS(A)$ .

The proof is by structural induction on  $t$ . By (i), the derivation of  $t$  has the form  $A \rightarrow_P \sigma(A_1, \dots, A_n) \rightarrow_P^* t$ . Let  $\sigma_{\mathcal{A}} = r_0 = \langle x_0, R_0 \rangle$  and  $\alpha_{r_0} = x_1 \cdots x_n$ .

By (ii),  $\dot{\ell}_{R_0}(x_i) = A_i$  for  $1 \leq i \leq n$ . Hence, by the induction hypothesis and (iii), there are  $r_i = \langle y_i, R_i \rangle \in eval_{\mathcal{A}}(t_i)$  such that  $R_i(y_i) \cong LHS(A_i) =$

$LHS(\ell_{R_0}(x_i)) \succeq R_0(x_i)$ , for all  $i \in [n]$ . In other words, there are multiplicities  $\mu_1, \dots, \mu_n$  and isomorphisms  $g_1, \dots, g_n$  such that  $R_i^{\mu_i}(y_i) \cong_{g_i} R_0(x_i)$ . It follows that  $r_0(r_1, \dots, r_n) = r_0[x_1 \leftarrow r_1] \cdots [x_n \leftarrow r_n]$  contains, in particular, the rule  $r = \langle x, R \rangle$ , where  $R = R_0[x_1 /_{g_1} r_1^{\mu_1}] \cdots [x_n /_{g_n} r_n^{\mu_n}]$  and  $x$  is the image of  $x_0$  in  $R$ . In particular,  $R(x) \cong R_0(x_0) \cong LHS(A)$  (where the second isomorphism comes from (iii)). This completes the proof of the claim since  $r_0(r_1, \dots, r_n) \subseteq eval_{\mathcal{A}}(t)$ .

Choosing  $A = A_0$ , the claim implies that  $\mathbf{L}_{\mathcal{A}}(g) = \emptyset$  if and only if  $L(g) = \emptyset$ . Using the already mentioned fact that the emptiness problem is decidable for regular tree grammars, this finishes the proof of the theorem.  $\square$

## 8 Conclusions

Adaptive star grammars have been devised for specifying software models based on graphs in [5], and have been used in an extensive case study on the refactoring of graphs that represent object-oriented programs [20]. Here we have explored their generative power and have related them to context-free ways of rewriting.

Figure 13 depicts the classes of languages studied in this paper, together with the class RE of recursively enumerable graph languages and the class DEC of decidable graph languages. By definition, DEC consists of all graph languages whose membership problem is decidable. Two subclasses of adaptive star replacement correspond to well-known context-free ways of graph replacement: Star replacement (SR, for short, Def. 2.7) corresponds to hyperedge replacement (HR, [11, 3]) by Theorem 2.8, and NR-like adaptive star replacement (NR-ASR, Def. 4.7) corresponds to boundary edNCE node replacement (B-edNCE, [10]) by Theorem 4.8, while it is well-known that node replacement languages properly include hyperedge replacement languages [9, Section 4.3]. Uniform adaptive star replacement (uASR, Def. 7.7) is more powerful than these classes because it can generate the language of all graphs (see Example 7.8), which cannot be generated by context-free node replacement [9, Theorem 4.17]. Of course, RE properly includes DEC. Theorem 6.4 shows that the adaptive star replacement languages (ASR, Def. 4.2) are in-

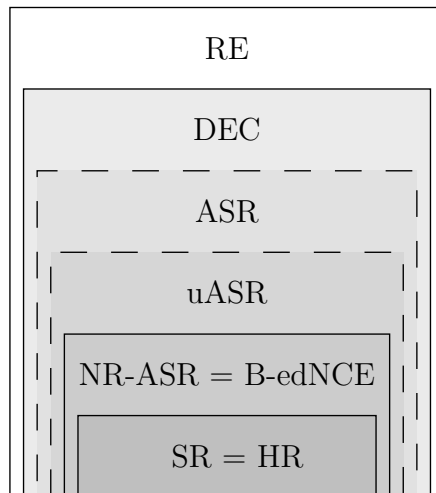


Figure 13: Relation of languages generated by star replacement and adaptive star replacement.

cluded in DEC.<sup>11</sup> It is still open whether the inclusions

$$\text{uASR} \subseteq \text{ASR} \subseteq \text{DEC}$$

are proper or not. This is why the borders between these classes are dashed in Figure 13. The case study [20] has indicated that adaptive star replacement may generate graph languages that represent certain context-sensitive properties, like correspondence of used entities with appropriate declarations, whereas other properties, like the correspondence of formal to actual parameter types, could not (yet) be defined. So these questions will be subject to future research.

In a way, adaptive star replacement is still rather close to context-free ways of replacement: it enjoys nondeterministic variants of commutativity, associativity, and generatability from derivation trees (Theorems 7.2, 7.4, and 7.6). In particular, uniform adaptive star replacement may not get stuck in derivations (Theorem 7.9), i.e., every rule applies to every star occurrence when their nonterminals match.

The closeness to context-free replacement suggests another line of investigation. We may ask ourselves which properties are decidable for adaptive

---

<sup>11</sup>However, adaptive star grammars generate the recursively enumerable string languages (represented as chain graphs) if the stars are generalized so that they are allowed to have parallel edges, see [5, Section 6].

star grammars, as we did for emptiness of the generated language in Theorem 7.10; we may devise transformations for the removal of unproductive and unused nonterminals, factorization etc.; and we may find ways to prove that certain properties of adaptive star rules imply that the graphs generated by them have certain graph-theoretic properties, like connectedness, acyclicity, or planarity.

It will also be important to think about efficient parsing. One way is to restrict the forms of rules. Another one could be to identify “context-free kernels” (without multiple nodes) underlying adaptive star rules that generate “spanning subgraphs” of the generated graphs. Parsing could then start with the kernel rules, and extend to the full rules and the complete graphs at a later stage.

As already mentioned in Section 1, a very concrete use of this work is made in the paper [4], where we use adaptive star grammars to specify languages of graphs that may be substituted for a variable in a graph transformation rule. Commutativity and associativity is essential in this situation. Such transformation rules have turned out to be useful for specifying refactoring operations, which may delete or copy large structures, like the subgraph representing a method body, in a single rule application. All these concepts will be implemented in the graph transformation language and tool **Diaplan** [6].

## References

- [1] Henning Christiansen. A survey of adaptable grammars. *SIGPLAN Notices*, 25(1):35–44, 1990.
- [2] Bruno Courcelle. An axiomatic definition of context-free rewriting and its application to NLC rewriting. *Theoretical Computer Science*, 55:141–181, 1987.
- [3] Frank Drewes, Annegret Habel, and Hans-Jörg Kreowski. Hyperedge replacement graph grammars. In Rozenberg [17], chapter 2, pages 95–162.
- [4] Frank Drewes, Berthold Hoffmann, Dirk Janssens, Mark Minas, and Niels Van Eetvelde. Shaped generic graph transformation. In Andy Schürr, Manfred Nagl, and Albert Zündorf, editors, *Applications of Graph Transformations with Industrial Relevance, Third International Symposium, AGTIVE 2007, Kassel, Germany, October 10-12, 2007, Revised Selected and Invited Papers*, number 5088 in Lecture Notes in Computer Science, pages 201–216. Springer-Verlag, 2008.

- [5] Frank Drewes, Berthold Hoffmann, Dirk Janssens, Mark Minas, and Niels Van Eetvelde. Adaptive star grammars. In Andrea Corradini, Hartmut Ehrig, Ugo Montanari, Leila Ribeiro, and Grzegorz Rozenberg, editors, *3rd Int'l Conference on Graph Transformation (ICGT'06)*, number 4178 in Lecture Notes in Computer Science, pages 77–91. Springer, 2006.
- [6] Frank Drewes, Berthold Hoffmann, Raimund Klein, and Mark Minas. Rule-based programming with **Diaplan**. *Electronic Notes in Theoretical Computer Science*, 127:15–26, 2005.
- [7] Frank Drewes, Berthold Hoffmann, and Mark Minas. Adaptive star grammars for graph models. In Hartmut Ehrig, Reiko Heckel, Grzegorz Rozenberg, and Gabriele Taentzer, editors, *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*, number 5214 in Lecture Notes in Computer Science, pages 442–457. Springer-Verlag, 2008.
- [8] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg, editors, *Graph Grammars and Their Application to Computer Science and Biology*, number 73 in Lecture Notes in Computer Science, pages 1–69. Springer, 1979.
- [9] Joost Engelfriet. Context-free graph grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3: Beyond Words, chapter 3, pages 125–213. Springer, 1999.
- [10] Joost Engelfriet and Grzegorz Rozenberg. Node replacement graph grammars. In Rozenberg [17], chapter 1, pages 1–94.
- [11] Annegret Habel. *Hyperedge Replacement: Grammars and Languages*. Number 643 in Lecture Notes in Computer Science. Springer, 1992.
- [12] Berthold Hoffmann. Conditional adaptive star grammars. In Frank Drewes, Annegret Habel, Berthold Hoffmann, and Detlef Plump, editors, *Manipulation of Graphs, Algebras and Pictures*, pages 171–189. Universität Bremen, 2009.
- [13] Berthold Hoffmann, Dirk Janssens, and Niels Van Eetvelde. Cloning and expanding graph transformation rules for refactoring. *Electronic Notes in Theoretical Computer Science*, 152(4):53–67, 2006. Proc. Graph and Model Transformation Workshop (GRAMoT'05).
- [14] Tom Mens, Serge Demeyer, and Dirk Janssens. Formalising behaviour-preserving transformation. In Andrea Corradini, Hartmut Ehrig, Hans-Jörg

- Kreowski, and Grzegorz Rozenberg, editors, *First International Conference on Graph Transformation (ICGT'02)*, number 2505 in Lecture Notes in Computer Science, pages 286–301. Springer, 2002.
- [15] Jorge Mezei and Jesse B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.
  - [16] Jörg Niere and Albert Zündorf. Using Fujaba for the development of production control systems. In Manfred Nagl, Andy Schürr, and Manfred Münch, editors, *Int'l Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE'99), Selected Papers*, number 1779 in Lecture Notes in Computer Science, pages 181–191. Springer, 2000.
  - [17] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. I: Foundations*. World Scientific, Singapore, 1997.
  - [18] Andy Schürr, Andreas Winter, and Albert Zündorf. The PROGRES approach: Language and environment. In Gregor Engels, Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. II: Applications, Languages, and Tools*, chapter 13, pages 487–550. World Scientific, Singapore, 1999.
  - [19] Shane Sendall. Combining generative and graph transformation techniques for model transformation: An effective alliance? In *Proc. OOPSLA'03-Workshop on Generative Techniques in the Context of MDA*, 2003. URL: [www.softmetaware.com/oopsla2003/mda-workshop.html](http://www.softmetaware.com/oopsla2003/mda-workshop.html).
  - [20] Niels Van Eetvelde. *A Graph Transformation Approach to Refactoring*. Doctoral thesis, Universiteit Antwerpen, May 2007.