Acyclic Contextual Hyperedge Replacement: Decidability of Acyclicity and Generative Power

Frank Drewes[1(✉)], Berthold Hoffmann[2(✉)], and Mark Minas[3(✉)]

[1] Umeå Universitet, Umeå, Sweden
drewes@cs.umu.se
[2] Universität Bremen, Bremen, Germany
hof@uni-bremen.de
[3] Universität der Bundeswehr München, Neubiberg, Germany
mark.minas@unibw.de

Abstract. Graph grammars based on contextual hyperedge replacement (CHR) extend the generative power of the well-known hyperedge replacement (HR) grammars to an extent that makes them useful for practical modeling. Recent work has shown that acyclicity is a key condition for parsing CHR grammars efficiently. In this paper we show that acyclicity of CHR grammars is decidable and that the generative power of acyclic CHR grammars lies strictly between that of HR grammars and unrestricted CHR grammars.

Keywords: Graph grammar · Hyperedge replacement · Contextual hyperedge replacement · Acyclicity · Decidability · Generative power

1 Introduction

Contextual hyperedge replacement (CHR, [3,4]) strengthens the generative power of hyperedge replacement (HR, [11]) significantly, e.g., to languages of unbounded treewidth. This is achieved by a moderate extension: productions may glue a graph not only to nodes attached to the nonterminal hyperedge being replaced, but also to nodes in the context. The applicability of such productions thus depends on the presence of context nodes created by other derivation steps.

In previous work, we have devised efficient parsing algorithms for subclasses of HR grammars, which rely on canonical orders for replacing nonterminals [5,6]. When these algorithms are extended to CHR, the canonical orders may be in conflict with dependencies arising from the creation and use of context nodes. Recently [9] we have shown that a CHR grammar Γ can be turned into an HR grammar generating graphs where the context nodes of Γ are "borrowed", i.e., generated like ordinary nodes. From these graphs, those generated by Γ can be obtained by "contraction", i.e., merging borrowed nodes with other nodes. This is correct provided that contractions cannot create cyclic chains of dependencies.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
N. Behr and D. Strüber (Eds.): ICGT 2022, LNCS 13349, pp. 3–19, 2022.
https://doi.org/10.1007/978-3-031-09843-7_1

In this paper, we establish two important properties of acyclic CHR grammars that have been left open in [9]. (1) It is decidable whether a CHR grammar is acyclic or not (Sect. 3). (2) Acyclicity reduces the generative power of CHR grammars by limiting the possibility to exploit node dependencies between productions (Sect. 4).

We start by recapitulating CHR grammars, borrowing grammars and contractions as well as acyclicity taken from [9] before we present the results of this paper in Sect. 3 and Sect. 4. Finally, we conclude the paper by discussing related and future work in Sect. 5.

2 Contextual Hyperedge Replacement

We let $\mathbb{N}$ denote the set of non-negative integers, and $[n]$ the set $\{1, \dots, n\}$ for all $n \in \mathbb{N}$. A^* denotes the set of all finite sequences over a set A; the empty sequence is denoted by ε, and the length of a sequence α by $|\alpha|$. For a function $f\colon A \to B$, its extension $f^*\colon A^* \to B^*$ to sequences is defined by $f^*(a_1 \cdots a_n) = f(a_1) \cdots f(a_n)$, for all $n \in \mathbb{N}$ and $a_1, \dots, a_n \in A$. As usual, $\to^+$ and $\to^*$ denote the transitive and the transitive reflexive closure of a binary relation $\to$.

Graphs. Let $\Sigma = \dot{\Sigma} \uplus \bar{\Sigma}$ be an alphabet of *labels* for nodes and edges respectively, where edge labels come with a *rank function* $rank\colon \bar{\Sigma} \to \mathbb{N}$.

Then a (*hyper-*) *graph* over Σ is a tuple $G = (\dot{G}, \bar{G}, att_G, lab_G)$, where $\dot{G}$ and $\bar{G}$ are disjoint finite sets of *nodes* and (*hyper-*) *edges*, respectively, the function $att_G\colon \bar{G} \to \dot{G}^*$ attaches sequences of nodes to edges, and the *labeling function* $lab_G\colon \dot{G} \cup \bar{G} \to \Sigma$ maps $\dot{G}$ to $\dot{\Sigma}$ and $\bar{G}$ to $\bar{\Sigma}$ in such a way that $|att_G(e)| = rank(lab_G(e))$ for every edge $e \in \bar{G}$. We assume that the attachment sequences are free of repetitions. $\mathcal{G}_\Sigma$ denotes the class of graphs over Σ. An edge carrying a label $\sigma \in \bar{\Sigma}$ is called *σ-edge*. G° denotes the discrete subgraph of a graph G, which is obtained by removing all edges.

A graph $G \in \mathcal{G}_\Sigma$ is called a *σ-handle* (or just a *handle*) if G has a single σ-edge e with $\sigma \in \bar{\Sigma}$, and each node of G is attached to e. $\mathcal{H}_\Sigma$ shall denote the set of *handles* of Σ. If $rank(\sigma) = 0$, a σ-handle is unique (up to isomorphism); we denote such a handle by $\sigma^\bullet$.

$G - x$ shall denote the graph G without the edge $x \in \bar{G}$. A set of edges $E \subseteq \bar{G}$ *induces* the subgraph consisting of these edges and their attached nodes. Given graphs $G_1, G_2 \in \mathcal{G}_\Sigma$ with disjoint edge sets, a graph $G = G_1 \cup G_2$ is called the *union* of G_1 and G_2 if G_1 and G_2 are subgraphs of G, $\dot{G} = \dot{G}_1 \cup \dot{G}_2$, and $\bar{G} = \bar{G}_1 \cup \bar{G}_2$. Note that $G_1 \cup G_2$ exists only if common nodes are consistently labeled, i.e., $lab_{G_1}(v) = lab_{G_2}(v)$ for $v \in \dot{G}_1 \cap \dot{G}_2$.

For graphs G and H, a *morphism* $m\colon G \to H$ is a pair $m = (\dot{m}, \bar{m})$ of functions $\dot{m}\colon \dot{G} \to \dot{H}$ and $\bar{m}\colon \bar{G} \to \bar{H}$ that preserve attachments and labels, i.e., $att_H(\bar{m}(v)) = \dot{m}^*(att_G(v))$, $lab_H(\dot{m}(v)) = lab_G(v)$, and $lab_H(\bar{m}(e)) = lab_G(e)$ for all $v \in \dot{G}$ and $e \in \bar{G}$.

The morphism is *injective* or *surjective* if both $\dot{m}$ and $\bar{m}$ have this property, and a *subgraph inclusion* of G in H if $m(x) = x$ for every node or edge x in G; then we write $G \subseteq H$. If m is surjective and injective, we say that G and H are *isomorphic*, written as $G \cong H$.

Contextual Hyperedge Replacement (CHR). The set $\bar{\Sigma}$ of edge labels is assumed to contain a subset $\mathcal{N}$ of *nonterminal labels*; edges with labels in $\mathcal{N}$ are *nonterminal* while all others are *terminal*.

A *production* $p = (L, R)$ consists of graphs L and R over Σ such that (1) the *left-hand side* L contains exactly one edge x, which is a nonterminal, and (2) the *right-hand side* R is an arbitrary supergraph of $L - x$. Nodes in L that are not attached to x are the *context nodes* of L (and of p); p is called *context-free* if it has no context nodes, and *contextual* otherwise.

We use a special form of standard double-pushout graph transformation [10] for applying productions: Let p be a production as above, and consider some graph G. An injective morphism $m\colon L \to G$ is called a *matching* for p in G. If such a matching exists, we say that p is *applicable* to the nonterminal $m(x) \in \bar{G}$. The *replacement* of $m(x)$ by R (via m) is then given as the graph H obtained from the disjoint union of $G - m(x)$ and R by identifying every node $v \in \dot{L}$ with $m(v)$. We write this as $G \Rightarrow_{m,p} H$, but omit m if it is irrelevant, and write $G \Rightarrow_{\mathcal{P}} H$ if $G \Rightarrow_p H$ for some p taken from a set $\mathcal{P}$ of productions.

This leads to the notion of a CHR grammar [3,4].

Definition 1 (CHR grammar). A *contextual hyperedge replacement grammar* $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ (*CHR grammar*) consists of alphabets Σ and $\mathcal{N}$ as above, a finite set $\mathcal{P}$ of productions over Σ, and a *start symbol* $S \in \mathcal{N}$ such that $\mathit{rank}(S) = 0$. The *language* generated by Γ is given as $\mathcal{L}(\Gamma) = \{G \in \mathcal{G}_{\Sigma\setminus\mathcal{N}} \mid S^\bullet \Rightarrow_{\mathcal{P}}^* G\}$. Γ is a (context-free) *hyperedge replacement grammar* (*HR grammar* [11]) if all productions in $\mathcal{P}$ are context-free.

CHR grammars can generate languages that cannot be generated by HR grammars. In particular, this includes languages of unbounded treewidth, like the language $\mathcal{G}_\Sigma$ of all graphs and our running example introduced next.

Example 1 (CHR grammar for dags). Figure 1 shows our running example, and introduces our conventions for drawing graphs and productions. Nodes are circles, nonterminal edges are rectangular boxes containing the corresponding labels, and terminal edges are shapes like ▷. (In this example, all nodes are labeled with the "invisible" label ␣, i.e., they are effectively unlabeled.) Edges

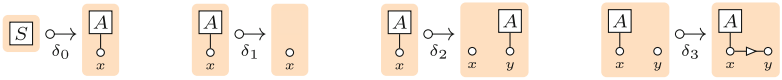


Fig. 1. Productions for generating dags (Example 1)

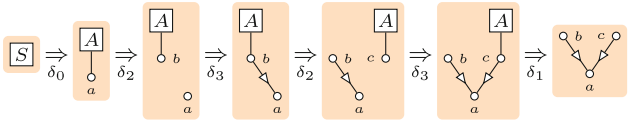


Fig. 2. A derivation with Δ

are connected to their attached nodes by lines ordered counter-clockwise around the edge, starting at noon. For productions (L, R), we draw L and R, and specify the inclusion of $\dot{L}$ in $\dot{R}$ by ascribing the same identifier to them, like x and y in our example.

Figure 1 defines the productions δ_0 to δ_3 of the CHR grammar Δ. S and A are nonterminal labels of rank 0 and 1, respectively, and $\triangleright$ is a binary terminal label. A derivation with this grammar is shown in Fig. 2.

It is easy to see that Δ derives only non-empty unlabeled acyclic graphs (*dags*, for short): In every derivation, the A-edge is attached to a node with indegree 0 so that no cycles may be introduced by production δ_3. Vice versa, every non-empty dag D can be generated with Δ: The nodes of D can be sorted topologically, e.g., as $v_1, \ldots, v_n$. Then every v_i can be generated with production δ_2, and its outgoing edges can be generated with production δ_3 since the targets of these edges must be nodes v_j with $j < i$. So $\mathcal{L}(\Delta)$ is indeed the set of all non-empty dags.

In previous work, we have devised efficient parsing algorithms for HR grammars [5,8]. These algorithms apply productions in canonical order (analogous to leftmost and rightmost derivations in string grammars). When extending these algorithms to CHR grammars, a production may only be applied when its context nodes have been created in previous steps. This may be in conflict with the canonical application orders. In [9], we have shown that there is a close relationship between a CHR grammar Γ and its so-called borrowing (HR) grammar $\hat{\Gamma}$: every graph $H \in \mathcal{L}(\Gamma)$ is a "contraction" of a graph $G \in \mathcal{L}(\hat{\Gamma})$. Moreover, the converse is also true as long as Γ is acyclic, a notion to be recalled later.

In the following, we assume that $\Sigma \backslash \mathcal{N}$ contains two auxiliary edge labels that are not used elsewhere in Γ: edges carrying the unary label $\circledcirc$ will mark borrowed nodes, and binary edges labeled $\neq$ will connect borrowed nodes with other nodes in the same right-hand side, to signify that they must not be identified with each other by contraction later on.

Definition 2 (Borrowing grammar). Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a CHR grammar. For $p = (L, R) \in \mathcal{P}$, its *borrowing production* $\hat{p} = (\hat{L}, \hat{R})$ is obtained by (1) removing every context node from $\hat{L}$ and (2) constructing $\hat{R}$ from R as follows: for every context node v of p, attach a new $\circledcirc$-edge to v, and add $\neq$-edges from v to every other node with the label $lab_L(v)$. The *borrowing grammar* $\hat{\Gamma} = \langle \Sigma, \mathcal{N}, \hat{\mathcal{P}}, S \rangle$ of Γ is given with $\hat{\mathcal{P}} = \{\hat{p} \mid p \in \mathcal{P}\}$.

Note that $\hat{p} = p$ if p is context-free.

Definition 3 (Contraction). For a graph G let

$$\dot{G}_{\circledcirc} = \{v \in \dot{G} \mid v = att_G(e) \text{ for a } \circledcirc\text{-edge } e \in \bar{G}\} \text{ and}$$
$$\neq_G = \{(u, v) \in \dot{G} \times \dot{G} \mid uv = att_G(e) \text{ for a } \neq\text{-edge } e \in \bar{G}\}.$$

A morphism $\mu\colon G \to H$ is called a *joining morphism* for G if $\dot{H} = \dot{G} \setminus \dot{G}_{\circledcirc}$, $\bar{H} = \bar{G}$, $\bar{\mu}$ and the restriction of $\dot{\mu}$ to $\dot{G} \setminus \dot{G}_{\circledcirc}$ are inclusions, and $(v, \dot{\mu}(v)) \notin \neq_G$ for every $v \in \dot{G}_{\circledcirc}$. The graph $core(H)$ obtained from H by removing all edges with labels $\circledcirc$ and $\neq$ is called the *μ-contraction* of G or just *a contraction* of G.

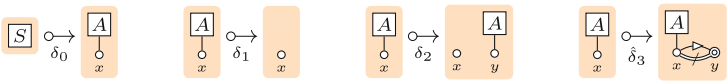


Fig. 3. Borrowing productions for generating dags

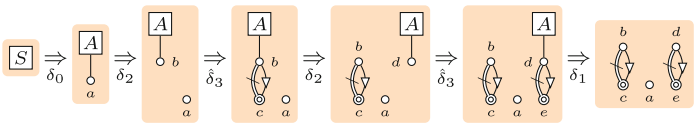


Fig. 4. A derivation with $\hat{\Delta}$

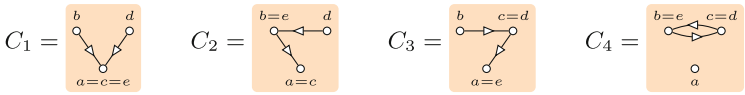


Fig. 5. The four contractions of the graph derived in Fig. 4

Example 2 (Borrowing grammar). Figure 3 shows the borrowing productions for the productions of the CHR grammar of dags in Fig. 1, where the contextual production δ_3 is replaced by the borrowing production $\hat{\delta}_3$.

A derivation with the borrowing grammar $\hat{\Delta}$ is shown in Fig. 4; the resulting terminal graph can be contracted in four possible ways, to the graphs C_1 to C_4 shown in Fig. 5. The contraction C_4 yields a cyclic graph; it is the only one of these four that cannot be generated with the productions of the CHR grammar Δ in Example 1.

Definition 4 (Borrowing version of a derivation). Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a CHR grammar and $\hat{\Gamma}$ its borrowing grammar. A derivation

$$S^\bullet \Rightarrow^{\hat{m}_1}_{\hat{p}_1} H_1 \Rightarrow^{\hat{m}_2}_{\hat{p}_2} H_2 \Rightarrow^{\hat{m}_3}_{\hat{p}_3} \cdots \Rightarrow^{\hat{m}_n}_{\hat{p}_n} H_n$$

in $\hat{\Gamma}$ is a *borrowing version* of a derivation

$$S^\bullet \Rightarrow^{m_1}_{p_1} G_1 \Rightarrow^{m_2}_{p_2} G_2 \Rightarrow^{m_3}_{p_3} \cdots \Rightarrow^{m_n}_{p_n} G_n$$

in Γ if the following hold, for $i = 1, 2, \ldots, n$ and $p_i = (L, R)$:

1. $\hat{p}_i$ is the borrowing production of p_i,
2. if $\bar{L} = \{e\}$ then $\hat{m}_i(e) = m_i(e)$, and
3. for every $x \in \bar{R} \cup (\dot{R} \setminus \dot{L})$, the images of x in G_i and H_i are the same.

By a straightforward induction, it follows that every derivation in Γ has a borrowing version in $\hat{\Gamma}$, and G_i is the μ_i-contraction of H_i for $i \in [n]$, where the joining morphism μ_i is uniquely determined by $\bar{\mu}_i(e) = e$ for all $e \in \bar{H}_i$.

Theorem 1 will show that the converse is also true, i.e., that every contraction of a graph in $\mathcal{L}(\hat{\Gamma})$ can also be derived in Γ, provided that Γ is acyclic. Informally,

Γ is cyclic if there is a derivation of a graph G in $\hat{\Gamma}$ and a contraction H of G so that there is a cyclic dependency between derivation steps that create nodes and derivation steps that use them as context nodes. These cyclic dependencies then result in derivations of graphs in $\hat{\Gamma}$ having a contraction that cannot be derived in Γ because there is no reordering of the derivation steps that yields a valid derivation in Γ.

For the definition of acyclicity in Definition 6, we need the well-known notion of derivation trees, which reflect the context-freeness of HR grammars [2, Definition 3.3]. Here we use the slightly modified version introduced in [9].

Definition 5 (Derivation tree). Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a HR grammar. The set $\mathbb{T}_\Gamma$ of *derivation trees* over Γ and the mappings $root\colon \mathbb{T}_\Gamma \to \mathcal{H}_\Sigma$ as well as $result\colon \mathbb{T}_\Gamma \to \mathcal{G}_\Sigma$ are inductively defined as follows:

- Each handle $G \in \mathcal{H}_\Sigma$ is in $\mathbb{T}_\Gamma$, and $root(G) = result(G) = G$.
- A triple $t = \langle G, p, c \rangle$ consisting of a nonterminal handle $G \in \mathcal{H}_\mathcal{N}$, a production $p \in \mathcal{P}$, and a sequence $c = t_1 t_2 \cdots t_n \in \mathbb{T}_\Gamma^*$ is in $\mathbb{T}_\Gamma$ if the union graphs $G' = G^\circ \cup \bigcup_{i=1}^n root(t_i)$ and $G'' = G^\circ \cup \bigcup_{i=1}^n result(t_i)$ exist, $G \Rightarrow_p G'$, and $nodes(result(t_i)) \cap nodes(result(t_j)) = nodes(root(t_i)) \cap nodes(root(t_j))$ for all distinct $i, j \in [n]$, where $nodes(H)$ denotes the node set of a graph H. Furthermore, we let $root(t) = G$ and $result(t) = G''$.

We assume the ordering of the subtrees in $c = t_1 t_2 \cdots t_n$ within a derivation tree $t = \langle G, p, c \rangle$ to be chosen arbitrarily, but kept fixed.

Let t, t' be any derivation trees. We call t a parent tree of t', written $t \succ t'$, if $t = \langle G, p, t_1 t_2 \cdots t_n \rangle$ and $t' = t_i$ for some i, and we call t' a subtree of t if $t' = t$ or $t = \langle G, p, t_1 t_2 \cdots t_n \rangle$ and t' is a subtree of t_i for some i. A derivation tree t *introduces* a node u (at its root) if $t = \langle G, p, t_1 t_2 \cdots t_n \rangle$ and $u \in nodes(root(t_i)) \setminus \dot{G}$ for some i. The set of all these nodes is denoted by $intro(t)$.

The following theorem is equivalent to Theorem 3.4 in [2]:

Lemma 1 (See [9, Theorem 1]). *Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a HR grammar, $H \in \mathcal{H}_\Sigma$ a handle and $G \in \mathcal{G}_\Sigma$ a graph. There is a derivation tree $t \in \mathbb{T}_\Gamma$ with $root(t) = H$ and $result(t) = G$ iff $H \Rightarrow_\mathcal{P}^* G$.*

Note that derivation trees are defined only for HR grammars. In the contextual case, any properly labeled node can be used as a context node as long as it has been created earlier in a derivation. This fact produces dependencies between derivation steps which do not exist in HR derivations.

In order to describe these additional dependencies, let us define the relation $\sqsubset_\mu$ on subtrees of a derivation tree $t \in \mathbb{T}_{\hat{\Gamma}}$ described by a joining morphism μ for $result(t)$. For any two subtrees t', t'' of t, we let $t' \sqsubset_\mu t''$ iff there is a node $u \in intro(t'')$ so that $\dot{\mu}(u) \neq u$ and $\dot{\mu}(u) \in intro(t')$.

Informally, $t' \sqsubset_\mu t''$ means that t' describes a derivation step (the topmost one that transforms the root handle of t'), which creates a node used as a contextual node in the corresponding topmost contextual derivation step described by t''. This restricts the set of all borrowing versions of derivations characterized by t:

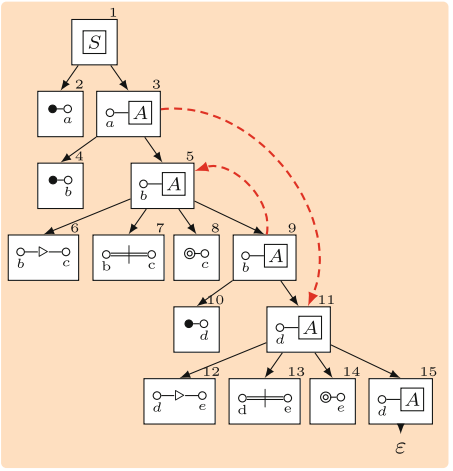


Fig. 6. Derivation tree of the derivation in Fig. 4

the derivation step described by t'' must occur after the one described by t'. However, the order of derivation steps must obey the parent tree relation $\succ$ as well. This motivates the following:

Definition 6 (Acyclic CHR grammar). A CHR grammar Γ is *acyclic* if $(\succ \cup \sqsubset_\mu)^+$ is irreflexive for all derivation trees $t \in \mathbb{T}_{\hat{\Gamma}}$ over $\hat{\Gamma}$ and all joining morphisms μ for $\mathit{result}(t)$. Otherwise, Γ is *cyclic*.

Example 3 (Derivation tree for dags). The derivation tree of the borrowing derivation in Fig. 4 is shown in Fig. 6 in black. Edges between derivation tree nodes represent the parent tree relation $\succ$. The thick dashed arrows drawn in red represent the relation $\sqsubset_\mu$ for the joining morphism μ defined by $\mu(e) = b$ and $\mu(c) = d$, yielding contraction C_4 in Fig. 5. This implies $t_3 \sqsubset_\mu t_{11}$ and $t_9 \sqsubset_\mu t_5$, respectively, when derivation subtrees are referred to by the numbers at their root nodes. Relations $\succ$ and $\sqsubset_\mu$ thus introduce a cycle affecting t_5 and t_9.

Theorem 1 (See [9, Theorem 2]). *Let Γ be a CHR grammar and $\hat{\Gamma}$ its borrowing grammar. For every graph $H \in \mathcal{L}(\Gamma)$, there is a graph $G \in \mathcal{L}(\hat{\Gamma})$ so that H is a contraction of G. Moreover, every contraction of a graph in $\mathcal{L}(\hat{\Gamma})$ is in $\mathcal{L}(\Gamma)$ if Γ is acyclic.*

3 Acyclicity of CHR Grammars is Decidable

Definition 6 does not provide effective means to check whether a CHR grammar is acyclic or not. In [9], we have devised a decidable sufficient criterion for acyclicity based on the so-called *grammar graph* $\mathsf{GG}(\Gamma)$ of a CHR grammar Γ (see [9, Definition 11]).

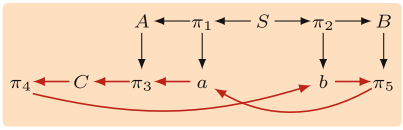
$\mathsf{GG}(\Pi) =$ 

Fig. 7. Grammar graph of the CHR grammar Π

$\mathsf{GG}(\Gamma)$ has nonterminal edge labels, node labels and productions (or production names) as nodes, and binary edges so that (1) every production is the target of an edge from its left-hand side nonterminal, and source of edges to the nonterminals on its right-hand side, (2) every node label ℓ is the source of edges to all productions with ℓ-nodes as context nodes, and the target of edges from all productions introducing ℓ-nodes on their right-hand side.

If $\mathsf{GG}(\Gamma)$ does not have a cycle that contains a node label as node, one can conclude that Γ is acyclic by [9, Lemma 1]. However, this criterion is not necessary, i.e., one cannot be sure that Γ is cyclic if $GG(\Gamma)$ has such a cycle, as the following (pathological) example shows.

Example 4 (Acyclic CHR grammar with cyclic grammar graph). Let Π be the CHR grammar with the following productions over nullary nonterminal labels S (the start symbol), A, B, C, and node labels a, b:

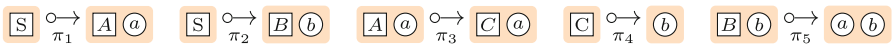
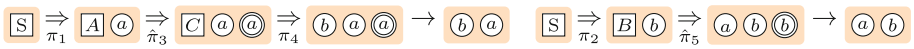


Figure 7 shows the grammar graph $\mathsf{GG}(\Pi)$ that has clearly a cycle containing node labels a and b as nodes. Even so, Π is acyclic as one can see as follows: The borrowing grammar $\hat{\Pi}$ has only the following two derivations, each of them with one possible contraction:



Both of them do not borrow any node before it has been created, i.e., $(\succ \cup \sqsubset_\mu)^+$ is irreflexive. Therefore, Π is acyclic.

We will now provide a decidable criterion for acyclicity of CHR grammars that is both sufficient *and* necessary, by turning a CHR grammar into a HR grammar whose language contains cyclic graphs iff the CHR grammar is cyclic. We start by motivating the construction of the HR grammar.

The dependency grammar Γ^{D} of a CHR grammar Γ is a HR grammar that has as its language graphs that contain derivation trees of the borrowing grammar $\hat{\Gamma}$. To be more precise, if there is a derivation of a graph H in $\hat{\Gamma}$, Γ^{D} derives a graph D that has a subgraph D' whose nodes correspond to the nonterminal derivation tree nodes of H, and its edges from parent to child nodes represent the parent tree relation $\succ$ on derivation trees. D contains additional nodes that represent nodes of H created by derivation steps, and additional edges. Whenever there is a possibility for a joining morphism μ to merge a node n with a borrowed node n' in H, D will contain a path between those nodes t and t' in D' that represent the creation of n and n', respectively. This path represents

$t \sqsubset_\mu t'$, and thus D contains a cycle iff $(\succ \cup \sqsubset_\mu)^+$ is reflexive, characterizing the cyclicity of Γ. This makes acyclicity of Γ decidable because it is decidable whether the language of an HR grammar contains cyclic graphs.

Let us first introduce some auxiliary concepts before we formally define dependency grammars: a production $p = (L, R)$ *borrows* a node label $\ell \in \dot{\Sigma}$ if L has a context node labeled ℓ; p *creates* ℓ if it creates a node labeled ℓ, i.e., $\ell = lab(v)$ for some $v \in \dot{R} \setminus \dot{L}$. We denote the sets of node labels borrowed and created by p by $B(p)$ and $C(p)$, respectively.

Definition 7 (Dependency grammar). Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a CHR grammar, where $\dot{\Sigma} = \{a_1, \ldots, a_m\}$. The *dependency grammar* of Γ is the HR grammar $\Gamma^{\mathsf{D}} = \langle \Sigma^{\mathsf{D}}, \mathcal{N}^{\mathsf{D}}, \mathcal{P}^{\mathsf{D}}, S \rangle$ consisting of the following components:

$$\mathcal{N}^{\mathsf{D}} = \{A^{\mathsf{D}} \mid A \in \mathcal{N}\} \cup \{S\} \qquad \bar{\Sigma}^{\mathsf{D}} = \mathcal{N}^{\mathsf{D}} \cup \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{t}\}$$
$$\dot{\Sigma}^{\mathsf{D}} = \mathcal{N} \cup \{\uparrow a_1, \ldots, \uparrow a_m, \downarrow a_1, \ldots, \downarrow a_m\} \qquad \mathcal{P}^{\mathsf{D}} = \{(S^\bullet, \mathsf{dep}(S))\} \cup \{p^{\mathsf{D}} \mid p \in \mathcal{P}\}$$

where $rank(A^{\mathsf{D}}) = 2m + 1$ for all $A \in \mathcal{N}$, S is nullary, and all other edge labels are binary.

To define the productions, let us denote by $\mathsf{dep}(A)$, for $A \in \mathcal{N}$, the A^{D}-handle consisting of an A^{D}-edge e which is attached to nodes $n_0, \ldots, n_{2m}$ labeled by $A, \uparrow a_1, \ldots, \uparrow a_m, \downarrow a_1, \ldots, \downarrow a_m$, respectively. For each of the labels $\ell \in \{A, \uparrow a_1, \ldots, \uparrow a_m, \downarrow a_1, \ldots, \downarrow a_m\}$, the unique node attached to e which is labeled ℓ will be denoted by $e.\ell$. We call $e.A$ the *main node* of the handle and the nodes $e.\uparrow a_i$ and $e.\downarrow a_i$ its *satellites*. (We also consider $e.A$ to be its own satellite.)

Now, for $p = (L, R) \in \mathcal{P}$ the *dependency production* $p^{\mathsf{D}} = (L^{\mathsf{D}}, R^{\mathsf{D}})$ is defined as follows: Suppose that the nonterminal edge of L has the label $A_0 \in \mathcal{N}$ and the nonterminal edges of R, ordered arbitrarily, have the labels $A_1, \ldots, A_k \in \mathcal{N}$. Then $L^{\mathsf{D}} = \mathsf{dep}(A_0)$ and R^{D} is the disjoint union of all handles $\mathsf{dep}(A_i)$ for $i \in [k]$, the set $\dot{L}^{\mathsf{D}}$ of all left-hand side nodes, and additional binary edges as specified below, where we denote the nonterminal edge in L^{D} by e_0 and that of $\mathsf{dep}(A_i)$ in R^{D} by e_i (for $i \in [k]$). Denoting a binary x-edge from $e_i.\ell$ to $e_j.\ell'$ by $e_i.\ell \to_x e_j.\ell'$, the terminal edges in R are:

$$e_0.A_0 \to_{\mathtt{t}} e_i.A_i \qquad \text{for all } i \in [k] \tag{1}$$
$$e_0.A_0 \to_{\mathtt{a}} e_0.\uparrow a \qquad \text{for all } a \in C(p) \tag{2}$$
$$e_i.\uparrow a \to_{\mathtt{b}} e_0.\uparrow a \qquad \text{for all } i \in [k] \tag{3}$$
$$e_i.\uparrow a \to_{\mathtt{c}} e_j.\downarrow a \qquad \text{for all } i, j \in [k]\ ,\ i \neq j \tag{4}$$
$$e_0.\downarrow a \to_{\mathtt{d}} e_i.\downarrow a \qquad \text{for all } i \in [k] \tag{5}$$
$$e_0.\downarrow a \to_{\mathtt{e}} e_0.A_0 \qquad \text{for all } a \in B(p) \tag{6}$$
$$e_i.\uparrow a \to_{\mathtt{f}} e_0.A_0 \qquad \text{for all } i \in [k] \text{ and } a \in B(p) \tag{7}$$

Note that each node in the right-hand side of a production is attached to one and only one nonterminal edge e, i.e., its denotation as $e.\ell$ is unique. Therefore, the derivation of a graph $D \in \mathcal{L}(\Gamma^{\mathsf{D}})$ induces a unique partition of $\dot{D}$ into subsets, each consisting of a main node and its satellites that have at some point during the derivation been attached to the same nonterminal.

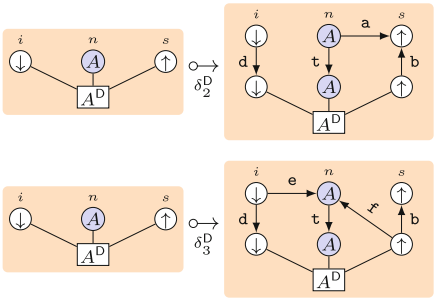


Fig. 8. Dependency productions for productions δ_2 and δ_3 of grammar Δ

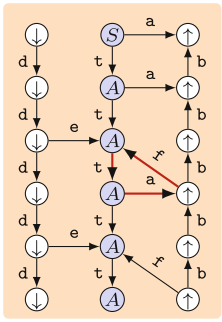


Fig. 9. Dependency graph of the derivation in Fig. 4

Derivations in borrowing grammars and dependency grammars are closely related: Consider a CHR grammar Γ, its borrowing grammar $\hat{\Gamma}$, and its dependency grammar Γ^{D}. The first step of a derivation in Γ^{D} yields $\mathsf{dep}(S)$. For $p \in \mathcal{P}$, the nonterminals in p^{D} (both in the left- and right-hand side) correspond bijectively to those in p, where each label A in p has been replaced by A^{D} in p^{D}. Thus, every derivation tree $t \in \mathbb{T}_{\hat{\Gamma}}$ corresponds to a unique derivation tree $t^{\mathsf{D}} \in \mathbb{T}_{\Gamma^{\mathsf{D}}}$, and vice versa. We call the graph $\mathit{result}(t^{\mathsf{D}})$ the *dependency graph* of t. By the above discussion, each graph derived by Γ^{D} is the dependency graph of some derivation tree $t \in \mathbb{T}_{\hat{\Gamma}}$.

Now, given such a dependency graph $D = \mathit{result}(t^{\mathsf{D}})$, consider the subgraph D' of D induced by its $\mathtt{t}$-edges (defined by (1) in Definition 7). By (1), the nodes of D' are the main nodes of D, i.e., those carrying a label $A \in \mathcal{N}$. As explained above, in the derivation of D each such node was – together with its satellites – attached to a unique nonterminal edge, and this edge carried the corresponding label A^{D}. This means that D' is a tree which is isomorphic to the derivation tree t of which D is the dependency graph, provided that we disregard the leaves of t. The isomorphism relates each node n of D' to a unique subtree $T(n) = \langle H, p, c\rangle$ of t, and the node label of n coincides with the nonterminal label of H.

Example 5. We consider CHR gammar Δ for dags again (see Example 1). Production δ_2 does not borrow any node, but creates the "invisible" node label $\sqcup$, i.e., $C(\delta_2) = \{\sqcup\}$ and $B(\delta_2) = \emptyset$. Production δ_3 does not create any node, but borrows the node label $\sqcup$, i.e., $C(\delta_3) = \emptyset$ and $B(\delta_3) = \{\sqcup\}$. Figure 8 shows the corresponding dependency productions where $\uparrow$ and $\downarrow$ denote $\uparrow_\sqcup$ and $\downarrow_\sqcup$, respectively. The main nodes are drawn with a blueish background.

Figure 9 shows a dependency graph D created by Δ^{D}. It corresponds to the derivation shown in Fig. 4. The main nodes of D are drawn with a blueish background again; they are also the nodes of tree D' induced by the $\mathtt{t}$-edges.

In the following, consider an arbitrary CHR grammar Γ, its dependency grammar Γ^{D}, and any dependency graph $D \in \mathcal{L}(\Gamma^{\mathsf{D}})$.

Lemma 2. *Every cycle in D contains an edge $\to_{\mathtt{e}}$ or $\to_{\mathtt{f}}$.*

Proof. We first show that every cycle in D contains a node labeled A for some $A \in \mathcal{N}$. If all the nodes in a cycle carried labels of the form $\uparrow a$ or $\downarrow a$, then the cycle can only contain edges labeled `b`, `c`, or `d`. But the definition of these edges in (3)–(5) prohibits such a cycle. Since the `t`-edges in D form a tree, the incoming edge of some A-node ($A \in \mathcal{N}$) in the cycle is not a `t`-edge, and must thus be labeled `e` or `f`. □

In the following, we write paths as an alternating sequence of nodes and edges like $n_0 \to_{a_1} n_1 \to_{a_2} \cdots \to_{a_l} n_l$. We may omit nodes between consecutive edges, and use the shorthand notation $n \to_u^* n'$ and $n \to_u^+ n'$ if there is a path from node n to n' consisting exclusively of $\to_u$-edges. $n \to_u^* n'$ also permits the empty path, i.e., $n = n'$, but $n \to_u^+ n'$ requires a path with at least one edge. We define further relations $\rightarrowtail$ and $\rightsquigarrow$ on nodes of D: $n \rightarrowtail n'$ iff there is a path $n \to_{\mathtt{t}}^+ \to_{\mathtt{a}} \to_{\mathtt{b}}^* \to_{\mathtt{f}} n'$, and $n \rightsquigarrow n'$ iff there is a path $n \to_{\mathtt{t}}^* \to_{\mathtt{a}} \to_{\mathtt{b}}^* \to_{\mathtt{c}} \to_{\mathtt{d}}^* \to_{\mathtt{e}} n'$.

Lemma 3. *D is cyclic iff $n \rightarrowtail n$ or $n \rightsquigarrow^k n$ for some $n \in \dot{D}$ and $k > 1$.*

Proof. D is obviously cyclic if it contains such a node n.

If D is cyclic, let D' be the tree induced by the `t`-edges of D. We distinguish between two cases: First assume that D contains an `f`-edge, say $n' \to_{\mathtt{f}} n$. Node n' must be labeled $\uparrow a$ for some $a \in \dot{\Sigma}$ and, by (2) and (3), there must be nodes n'', n''' such that $n'' \to_{\mathtt{a}} n''' \to_{\mathtt{b}}^* n' \to_{\mathtt{f}} n$. Note that n and n'' carry nonterminal labels in $\mathcal{N}$ and are thus main nodes of D. Now let m' and m''' be the main nodes of satellites n' and n''', respectively. Then n, m', n'', m''' are nodes of D'. By (2), (3), and (7), $n'' = m'''$, m''' is a (not necessarily proper) descendant of m', and m' is a child of n, i.e., n'' is a proper descendant of n, and therefore $n \to_{\mathtt{t}}^+ n''$. This shows that $n \rightarrowtail n$.

Now consider the case where D does not contain an edge $\to_{\mathtt{f}}$. We select any cycle of D, which must contain a node n with an incoming edge $\to_{\mathtt{e}}$ by Lemma 2. Removing all occurrences of $\to_{\mathtt{e}}$ in the cycle decomposes it into k paths. These paths have the form $\to_{\mathtt{t}}^* \to_{\mathtt{a}} \to_{\mathtt{b}}^* \to_{\mathtt{c}} \to_{\mathtt{d}}^*$, i.e., we have $n \rightsquigarrow^k n$ for some $k \geq 1$. But then we must have $k > 1$, which can be seen as follows. Consider a path $n_1 \to_{\mathtt{t}}^* n_2 \to_{\mathtt{a}} n_3 \to_{\mathtt{b}}^* n_4 \to_{\mathtt{c}} n_5 \to_{\mathtt{d}}^* n_6 \to_{\mathtt{e}} n_7$, and let m_i be the main node of satellite n_i, for $i \in [7]$. Then each m_i is a node of D'. In D', the path from m_1 to m_2 descends down the tree (by (1)), then stays at $m_2 = m_3$ (by (2)), and ascends to an ancestor m_4 of m_3 (by (3)). Now, by (4), m_5 is a proper sibling of m_4, and $m_6 = m_7$ is a descendant of m_5 (by (5) and (6)). Since D' is a tree, this implies that $n_1 \neq n_7$. Therefore, we cannot have $n \rightsquigarrow n$ for any node n. □

Lemma 4. *Let Γ be a CHR grammar and Γ^{D} its dependency grammar. Γ is cyclic iff the language of Γ^{D} contains a cyclic graph.*

Proof. Let $\Gamma = \langle \Sigma, \mathcal{N}, \mathcal{P}, S \rangle$ be a CHR grammar, $\hat{\Gamma}$ its borrowing grammar, and Γ^{D} its dependency grammar.

Let us first assume that the language of Γ^{D} contains a cyclic graph D, which is the dependency graph of a derivation tree $t \in \mathbb{T}_{\hat{\Gamma}}$ over $\hat{\Gamma}$. Mirroring Lemma 3, we distinguish two cases:

Case 1: D contains a node n such that $n \rightsquigarrow^k n$ for some $k > 1$, i.e., there are nodes $n_0, n_1, \ldots, n_k \in \dot{D}$, $n'_i, n''_i, n'''_i \in \dot{D}$, subtrees $\langle H_i, p_i, c_i\rangle$ and $\langle H'_i, p'_i, c'_i\rangle$ of t, and node labels $a_i, \ldots, a_k \in \dot{\Sigma}$ such that $n_0 = n = n_k$ and $lab_D(n''_i) = \uparrow a_i$, $lab_D(n'''_i) = \downarrow a_i$, $T(n_i) = \langle H_i, p_i, c_i\rangle$, $T(n'_i) = \langle H'_i, p'_i, c'_i\rangle$, and $n_{i-1} \to_{\mathtt{t}}^* n'_i \to_{\mathtt{a}} n''_i \to_{\mathtt{b}}^* \to_{\mathtt{c}} \to_{\mathtt{d}}^* n'''_i \to_{\mathtt{e}} n_i$ for $i \in [k]$.

By the condition in (6), $n'''_i \to_{\mathtt{e}} n_i$ implies that $a_i \in B(p_i)$, i.e., p_i has a context node labeled a_i. By the condition in (2), $n'_i \to_{\mathtt{a}} n''_i$ further implies that $a_i \in C(p'_i)$, i.e., p'_i creates a node labeled a_i. Therefore, by merging the corresponding nodes, there is a joining morphism μ_i such that $T(n'_i) \sqsubset_{\mu_i} T(n_i)$. Note that $n'_i \neq n_i$ follows from the fact that the path from n'_i to n_i contains an edge $\to_{\mathtt{c}}$, which implies that there is no $\neq$-edge between those a_i-nodes, and thus that these nodes are not prevented from becoming merged. Finally, $n_{i-1} \to_{\mathtt{t}}^* n'_i$ implies that $T(n'_i)$ is a subtree of $T(n_{i-1})$ in t, and therefore $T(n_{i-1}) \succ^* T(n'_i) \sqsubset_{\mu_i} T(n_i)$ for $i \in [k]$. It should be clear that there is a joining morphism that can act as μ_i for $i \in [k]$, and therefore, $T(n)(\succ \cup \sqsubset_\mu)^+ T(n)$, i.e., Γ is cyclic by Definition 6.

Case 2: D contains a node n such that $n \rightarrowtail n$. i.e., there is a path $n \to_{\mathtt{t}}^+ \to_{\mathtt{a}} \to_{\mathtt{b}}^* \to_{\mathtt{f}} n$ in D. By arguments entirely analogous to Case 1, one can then conclude that there is a joining morphism μ such that $T(n)(\succ \cup \sqsubset_\mu)^+ T(n)$, i.e., Γ is cyclic.

Assume now that Γ is cyclic. By Definition 6, there is a derivation tree $t \in \mathbb{T}_{\hat{\Gamma}}$ and a joining morphism μ for $result(t)$ such that $(\succ \cup \sqsubset_\mu)^+$ is reflexive. Because $\succ^+$ is irreflexive, there must be subtrees $t_0, \ldots, t_k$ and $t'_1, \ldots, t'_k$ of t for some $k \geq 1$ such that $t_0 = t_k$ and $t_{i-1} \succ^* t'_i \sqsubset_\mu t_i$ for $i \in [k]$. Let us choose such subtrees $t_0, \ldots, t_k$ and $t'_1, \ldots, t'_k$ of t, with the additional condition that k is minimal (with respect to the given t), and let D be the dependency graph of t.

If $k = 1$, we have $t_0 \succ^* t'_1 \sqsubset_\mu t_0$, i.e., t'_1 is a (proper) subtree of t_0, and the topmost derivation step of t_0 uses a node with label $a \in \dot{\Sigma}$ as a context node that is created by the topmost derivation step of t'_1, indicated by μ merging the corresponding nodes. Let n and n' be the nodes of D such that $T(n) = t_0$ and $T(n') = t'_1$. Clearly, D contains the cycle $n \to_{\mathtt{t}}^+ n' \to_{\mathtt{a}} n'' \to_{\mathtt{b}}^* n''' \to_{\mathtt{f}} n$ for some nodes n'', n''' with $lab_D(n'') = lab_D(n''') = \uparrow a$.

If $k > 1$, we can conclude that t'_i is not a subtree of t_i for any $i \in [k]$. Otherwise, we would have $t_i \succ^* t'_i \sqsubset_\mu t_i$, contradicting the selection of $k > 1$ being minimal for t.

Let $n_i \in \dot{D}$ such that $T(n_i) = t_i$ for $i = 0, \ldots, k$ and $n'_i \in \dot{D}$ such that $T(n'_i) = t'_i$ for $i \in [k]$ and assume that t_i uses a node with label $a_i \in \dot{\Sigma}$ as context node that is created by the topmost derivation step of t'_i for $i \in [k]$, indicated by μ merging the corresponding nodes. D thus contains paths $n_{i-1} \to_{\mathtt{t}}^* n'_i \to_{\mathtt{a}} n''_i \to_{\mathtt{b}}^* \to_{\mathtt{c}} \to_{\mathtt{d}}^* n'''_i \to_{\mathtt{e}} n_i$ where $n''_i, n'''_i \in \dot{D}$ such that $lab_D(n''_i) = \uparrow a_i$ and $lab_D(n'''_i) = \downarrow a_i$ for $i \in [k]$. These paths define a cycle in D because $n_0 = n_k$. □

Example 6. The dependency graph shown in Fig. 9 in Example 5 has a cycle drawn in red indicating that the CHR grammar Δ for dags is indeed cyclic.

Let us reconsider the "pathological" CHR grammar Π introduced in Example 4. The criterion devised in [9] does not help to decide whether Π is acyclic

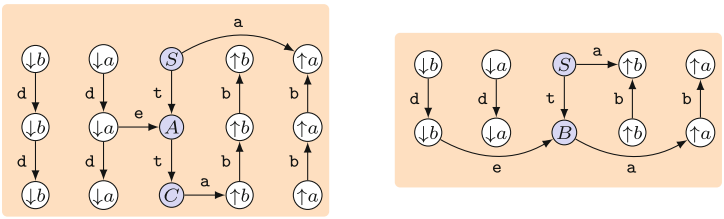


Fig. 10. Dependency graphs for the grammar in Example 4

or cyclic, but Lemma 4 does: Fig. 10 shows the only two dependency graphs of $\mathcal{L}(\Pi^{\mathsf{D}})$. Π is indeed acyclic because they do not contain any cycle.

Theorem 2. *Acyclicity of CHR grammars is decidable.*

Proof. Following Definition 7, the dependency grammar Γ^{D} of a CHR grammar Γ can effectively be constructed. Moreover, it is decidable whether the language of an HR grammar such as Γ^{D} contains cyclic graphs [12]. Therefore, acyclicity of CHR grammars is decidable by Lemma 4. □

The construction of the dependency grammar Γ^{D} of a CHR grammar Γ according to rules (1)–(6) can be made in polynomial time. However, a straightforward cyclicity check would inspect all combinations of the finite set of dependency productions $\mathcal{P}^{\mathsf{D}}$, taking exponential time. We are not aware of any more efficient cyclicity checks for HR grammars. However, this problem closely corresponds to the circularity problem of attribute grammars. Attribute grammars are a well-known formalism for adding semantic information to context-free string grammars, and a proper definition of their semantics requires acyclic dependencies between attributes [14]. Knuth's (corrected) algorithm for checking circularity of an attribute grammar has an exponential worst-case running time [15]; Jazayeri et al. [13] have further proved that any deterministic algorithm solving this problem requires exponential running time, so that an efficient algorithm for checking the cyclicity of CHR grammars is unlikely to exist.

4 Acyclicity Restricts Generative Power

In this section, we answer the second question left open in [9], namely whether the class of graph languages generated by acyclic CHR grammars is a proper subset of the class of all CHR languages. It was conjectured in [9] that the set of all dags (see Example 1) could not be generated by an acyclic CHR grammar, a proof of which would thus answer the question positively. We now show that this is indeed the case.

As usual, we call two nodes v, v' of a graph G adjacent if both v and v' occur in $att_G(e)$, for some edge $e \in \bar{G}$. It is well known that HR languages are graph languages of bounded treewidth. Since the number of (unordered) pairs of adjacent nodes in a graph G of bounded treewidth is linearly bounded by the number $|\dot{G}|$ of its nodes, the following observation is immediate.

Observation 1. For every HR language L there is a constant w such that no graph $G \in L$ contains more than $w|\dot{G}|$ pairs of adjacent nodes.

In a borrowing grammar, each borrowed node is connected via $\neq$-edges to at most s other nodes, where s is the number of nodes of the largest right-hand side of the grammar. Hence, if we are given a joining morphism $\mu\colon G \to H$ for a graph G in the borrowing language generated by this grammar, and $z \in \dot{G} \setminus \dot{H}$ is a borrowed node, then for all but at most s nodes $v \in \dot{H}$, we can instead map z to v to obtain another valid joining morphism.

To express this formally, let us first note that a joining morphism μ is uniquely determined by $\dot{\mu}$. More precisely, given a graph G, consider a function $f\colon \dot{G} \to \dot{G} \setminus \dot{G}_{\circledcirc}$ such that, for all $u, v \in \dot{G}$,

(J1) if $u \notin \dot{G}_{\circledcirc}$ then $f(u) = u$, and
(J2) $f(u) = f(v)$ only if u and v are not connected by a $\neq$-edge.

Then there is a unique joining morphism $\mu\colon G \to H$ with $\dot{\mu} = f$. In particular, H is uniquely determined by f. This also implies that, given a joining morphism μ and a borrowed node $z \in \dot{G}_{\circledcirc}$, we can modify μ so that it, instead of joining z with $\dot{\mu}(z)$, joins it with any other node $v \in \dot{G} \setminus \dot{G}_{\circledcirc}$, provided that condition (J2) is fulfilled. In the following, the resulting joining morphism is denoted by $\mu_{z \mapsto v}$, i.e., $\mu_{z \mapsto v}$ is the unique joining morphism such that, for all $u \in \dot{G}$,

$$\dot{\mu}_{z \mapsto v}(u) = \begin{cases} v & \text{if } u = z \\ \dot{\mu}(u) & \text{otherwise.} \end{cases}$$

Using this notation, the observations above can be stated formally as follows.

Observation 2. For every borrowing language L, there is a constant s such that the following holds. Let $G \in L$, and let $\mu\colon G \to H$ be a joining morphism for G. Then, for every node $z \in \dot{G}_{\circledcirc}$, there are at least $|\dot{H}| - s$ nodes $v \in \dot{H}$ such that $\dot{\mu}_{z \mapsto v}$ determines a valid joining morphism for G.

We can now show that the language $\mathcal{L}(\Delta)$ of Example 1 is a CHR language beyond the generative capacity of acyclic CHR grammars.

Lemma 5. *The language of all (unlabeled) dags can be generated by a CHR grammar but not by an acyclic CHR grammar.*

Proof. Example 1 shows that the language $\mathcal{D}$ of all directed acyclic graphs is indeed a CHR language. It remains to show that $\mathcal{D}$ cannot be generated by an acyclic CHR grammar. To prove this by contradiction, assume that there is an acyclic CHR grammar Γ auch that $L(\Gamma) = \mathcal{D}$. We can assume that $\hat{\Gamma}$ does not generate borrowed nodes that are only incident with $\circledcirc$- and $\neq$-edges, because it is well known that $\hat{\Gamma}$ can otherwise be modified to remove such nodes from its language, and by the definition of contraction, such nodes do not affect the result of a contraction.

For $m, n \in \mathbb{N}$, define H_{mn} to be the unlabeled graph with $\dot{H}_{mn} = \{u_1, \ldots, u_m\} \cup \{v_1, \ldots, v_n\} \cup \{u'_1, \ldots, u'_m\}$ with $u_m = v_1$ and $v_n = u'_1$, such

that there are edges from u_i to u_{i+1} and from u'_i to u'_{i+1} for all $i \in [m-1]$, as well as from v_i to v_j for $1 \leq i < j \leq n$. Thus, H_{mn} consists of a complete graph on n nodes between two chains of m nodes from above and below (if edges point downwards). In the following, we consider pairs of graphs $G_{mn} \in L(\hat{\Gamma})$ and $H_{mn} \in L(\Gamma)$ such that G_{mn} contracts to H_{mn} via a joining morphism μ.

We first fix m in such a way that it is larger than the constant s of Observation 2 applied to $\hat{\Gamma}$. Then G_{mn} can have at most $2(m-1)+2(n-1)$ nodes u such that $\dot{\mu}(u) \in \{u_1, \ldots, u_m, u'_1, \ldots, u'_m\}$ (since each of these nodes has at least one incident edge while H_{mn} contains only $2(m-1)+2(n-1)$ edges incident with them in total). Since we have fixed m, this number is linear in n. However, the total number of edges of H_{mn} – and thus that of G_{mn} – grows quadratically in n, which by Observation 1 means that we can choose n sufficiently large to make sure that G_{mn} contains at least one node $z \in \dot{G}_{mn} \setminus \dot{H}_{mn}$ such that $\dot{\mu}(z) = v_i$ for some $i \in \{2, \ldots, n-1\}$.

Now, consider such a node z. Since $m > s$, where s is the constant of Observation 2, by that same observation there is at least one $j \in [m]$ such that $\mu_{z \mapsto u_j}$ is also a valid joining morphism. The same holds for $u'_1, \ldots, u'_m$ instead of $u_1, \ldots, u_m$.

Let e be an edge incident with z that is neither a $\circledcirc$- nor a $\neq$-edge. If z is the target of e, then its source is a node u such that $\dot{\mu}(u) = v_p$ for some $p < i$. Consider an appropriate $j \in [m]$ such that $\mu_{z \mapsto u_j}$ is a joining morphism (which, by the previous paragraph, exists). Thus, instead of being joined with v_i by μ, we join z with u_j by $\mu_{z \mapsto u_j}$. Since this leaves the edges originating from $u_j, \ldots, u_m$ unaffected, the path from u_j to $u_m = v_1$ still exists in the contraction with respect to $\mu_{z \mapsto u_j}$, and so does the edge from v_1 to v_p. However, e now leads from v_p to u_j, which results in the cycle $u_j, \ldots, u_m, v_p, u_j$.

The case where z is the source of e is symmetric, using $\mu_{z \mapsto u'_j}$ instead of $\mu_{z \mapsto u_j}$. Thus, both cases lead us to the conclusion that $L(\Gamma)$ contains a graph that has a cycle, contradicting the initial assumption that $L(\Gamma) = \mathcal{D}$. □

From Lemma 5, Example 1 and the fact (known from [9]) that acyclic CHR grammars can generate various graph languages that are not HR languages, we get the second main result of this paper as an immediate consequence.

Theorem 3. *The generative power of acyclic CHR grammars lies strictly between the generative powers of HR and CHR grammars.*

5 Conclusions

In this paper, we have established two main results: (1) acyclicity of CHR grammars is decidable and (2) the generative power of acyclic CHR grammars lies strictly between that of HR and CHR grammars. Since acyclicity is one condition for efficient parsing with the predictive top-down and predictive shift-reduce algorithms of [7,9], this is important for the practical use of CHR grammars.

Since this paper is on a very specific topic, related work is rare. We are only aware of Berglund's pumping lemma for CHR grammars [1], which shows their

close relation to context-free hyperedge replacement. (It seems that this pumping lemma cannot be used to prove Lemma 5.)

In future work, we plan to compensate for the restricted generative power of acyclic CHR grammars by conditional contractions, which may require or forbid the existence of certain paths. Then, e.g., the language of all dags can be generated by a conditional acyclic CHR grammar that forbids that there is a path to a borrowed node from its contracted node. The specification of such paths could be based on the "navigational logic" proposed by Orejas et al. [16].

References

1. Berglund, M.: Analyzing and pumping hyperedge replacement formalisms in a common framework. In: Echahed, R., Plump, D. (eds.) Pre-Proceedings Tenth International Workshop on Graph Computation Models, GCM@STAF 2019, Eindhoven, The Netherlands, 17 July 2019, pp. 17–32 (2019)
2. Drewes, F., Habel, A., Kreowski, H.J.: Hyperedge replacement graph grammars. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations, chap. 2, pp. 95–162. World Scientific, Singapore (1997)
3. Drewes, F., Hoffmann, B.: Contextual hyperedge replacement. Acta Informatica **52**(6), 497–524 (2015). https://doi.org/10.1007/s00236-015-0223-4
4. Drewes, F., Hoffmann, B., Minas, M.: Contextual hyperedge replacement. In: Schürr, A., Varró, D., Varró, G. (eds.) AGTIVE 2011. LNCS, vol. 7233, pp. 182–197. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34176-2_16
5. Drewes, F., Hoffmann, B., Minas, M.: Predictive top-down parsing for hyperedge replacement grammars. In: Parisi-Presicce, F., Westfechtel, B. (eds.) ICGT 2015. LNCS, vol. 9151, pp. 19–34. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21145-9_2
6. Drewes, F., Hoffmann, B., Minas, M.: Predictive shift-reduce parsing for hyperedge replacement grammars. In: de Lara, J., Plump, D. (eds.) ICGT 2017. LNCS, vol. 10373, pp. 106–122. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61470-0_7
7. Drewes, F., Hoffmann, B., Minas, M.: Extending predictive shift-reduce parsing to contextual hyperedge replacement grammars. In: Guerra, E., Orejas, F. (eds.) ICGT 2019. LNCS, vol. 11629, pp. 55–72. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23611-3_4
8. Drewes, F., Hoffmann, B., Minas, M.: Formalization and correctness of predictive shift-reduce parsers for graph grammars based on hyperedge replacement. J. Log. Algebr. Methods Program. (JLAMP) **104**, 303–341 (2019). https://doi.org/10.1016/j.jlamp.2018.12.006
9. Drewes, F., Hoffmann, B., Minas, M.: Rule-based top-down parsing for acyclic contextual hyperedge replacement grammars. In: Gadducci, F., Kehrer, T. (eds.) ICGT 2021. LNCS, vol. 12741, pp. 164–184. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78946-6_9
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs, Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31188-2
11. Habel, A.: Hyperedge Replacement: Grammars and Languages. LNCS, vol. 643. Springer, Heidelberg (1992). https://doi.org/10.1007/BFb0013875

12. Habel, A., Kreowski, H., Vogler, W.: Metatheorems for decision problems on hyperedge replacement graph languages. Acta Informatica **26**(7), 657–677 (1989). https://doi.org/10.1007/BF00288976
13. Jazayeri, M., Ogden, W.F., Rounds, W.C.: The intrinsically exponential complexity of the circularity problem for attribute grammars. Commun. ACM **18**(12), 697–706 (1975). https://doi.org/10.1145/361227.361231
14. Knuth, D.E.: Semantics of context-free languages. Math. Sys. Theory **2**(2), 127–145 (1968). https://doi.org/10.1007/BF01692511. Correction: [15]
15. Knuth, D.E.: Semantics of context-free languages: correction. Math. Sys. Theory **5**(2), 95–96 (1971). https://doi.org/10.1007/BF01702865
16. Navarro, M., Orejas, F., Pino, E., Lambers, L.: A navigational logic for reasoning about graph properties. J. Log. Algebraic Methods Program. **118**, 100616 (2021). https://doi.org/10.1016/j.jlamp.2020.100616