EPiC Series in Computing

Volume 100, 2024, Pages 165–180

Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning

# Tree-Verifiable Graph Grammars

Mark Chimes[1] [*], Radu Iosif[2], and Florian Zuleger[1]

[1] Technische Universtität Wien, Vienna, Austria
{mark.chimes,florian.zuleger}@tuwien.ac.at
[2] Université Grenoble Alpes, Grenoble, France
Radu.Iosif@univ-grenoble-alpes.fr

**Abstract**

Hyperedge-Replacement grammars (HR) have been introduced by Courcelle in order to extend the notion of context-free sets from words and trees to graphs of bounded tree-width. While for words and trees the syntactic restrictions that guarantee that the associated languages of words resp. trees are regular - and hence, MSO-definable - are known, the situation is far more complicated for graphs. Here, Courcelle proposed the notion of regular graph grammars, a syntactic restriction of HR grammars that guarantees the definability of the associated languages of graphs in Counting Monadic Second Order Logic (CMSO). However, these grammars are not complete in the sense that not every CMSO-definable set of graphs of bounded tree-width can be generated by a regular graph grammar. In this paper, we introduce a new syntactic restriction of HR grammars, called tree-verifiable graph grammars, and a new notion of bounded tree-width, called embeddable bounded tree-width, where the later restricts the trees of a tree-decomposition to be a subgraph of the analyzed graph. The main property of tree-verifiable graph grammars is that their associated languages are CMSO-definable and that they have bounded embeddable tree-width. We show further that they strictly generalize the regular graph grammars of Courcelle. Finally, we establish a completeness result, showing that every language of graphs that is CMSO-definable and of bounded embeddable tree-width can be generated by a tree-verifiable graph grammar.

## 1 Introduction

The notion of *regular* word and tree languages is one of the pillars of the theory of formal languages because many equivalent representations of regular languages are known. In particular, we have that Monadic Second Order Logic (MSO) over words defines exactly the class of regular word languages. On the other hand, regular word languages are a subset of the context-free

[*]The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101034440. The second author wishes to acknowledge the support of the French National Research Agency project Non-Aggregative Resource COmpositions (NARCO) under grant number ANR-21-CE48-0011.

word languages, and regular word grammars can be seen as syntactic restrictions of context-free grammars that ensure the regularity of the generated languages. Regular languages enjoy many desirable properties, in particular, the following problems are known to be decidable: (1) *emptiness*: is the language empty? (2) *membership*: does a given word resp. tree belong to the language? (3) *inclusion*: given two languages, is the first included in the second?

Most expressivity (i.e., equivalence between regular and MSO-definable languages) and decidability (i.e., emptiness, membership, inclusion) results can be transferred from words to trees. Because of the importance of regular languages and their decidability properties, it is highly desirable to find similar results on graphs, which are important for many branches of computing, such as e.g., static analysis [19], databases and knowledge representation [1], concurrency [7] and aspects of machine learning, such as neural networks [20].

In this paper, we study graph languages that lie at the intersection of HR-*context-free* and CMSO-*definable* classes. The sets of graphs in this class are, at the same time (i) least solutions of a system of recursive equations using operations that substitute a hyper-edge of a graph by another graph, and (ii) models of a formula that uses (quantification over) variables denoting combined sets of vertices and edges and relation symbols (from a fixed finite signature) describing the incidence relation of a graph. In particular, CMSO is the extension of MSO with atomic propositions that express constraints on the cardinality of a set, modulo a constant.

This intersection class has nice algorithmic properties, because emptiness, membership and inclusion are all decidable. In particular, the decidability of inclusion follows from a celebrated theorem proved by Courcelle, stating that the satisfiability of a CMSO formula is decidable over graphs of bounded tree-width, a parameter that, intuitively speaking, measures the difference between a graph and a tree encoding of that graph [5]. We recall that the MSO-definability of a word language defined by a context-free grammar is, in general, undecidable [10]. Hence, as for words and trees, we will put syntactic restrictions on HR-grammars in order to ensure CMSO-definability.

The *hyperedge replacement* (HR) algebra describe the operations used to build graphs, using zero or more distinguished vertices, called *sources*, labeled by numbers $1, \ldots, n$. A *substitution operation* replaces a *nonterminal edge* attached to vertices $v_1, \ldots, v_n$ in a graph $H$ by a disjoint graph $K$ with sources $1, \ldots, n$, such that the $i$-th source of $K$ is glued with $v_i$, for each $i = 1, \ldots, n$ and this is the only overlapping between the two graphs. A *parallel composition* is a special substitution that glues the $i$-th sources of two graphs, for $i = 1, \ldots, n$. We note that HR-grammars generalize context-free word and tree grammars to graphs of bounded tree-width.

The work in this paper is inspired from the class of *regular graph grammars* introduced by Courcelle [6] as a graph counterpart to the left- or right-recursive word grammars, which define regular, and, hence, MSO-definable [2] word languages. The rules of a regular graph grammar use two disjoint sets of nonterminals, called *recursive* and *non-recursive*. Intuitively, a recursive nonterminal can be composed in parallel with any number of occurrences of the same non-recursive terminal in a derivation, but a substitution may only generate non-recursive terminals. The rules of a regular graph grammar are defined by two syntactic conditions: (i) each two vertices of a graph $H$ in a substitution operation are connected by a path that crosses only terminal edges and internal (i.e., non-source) vertices, except for endpoints, and (ii) the parallel composition involves at most one recursive nonterminal.

The graphs generated by regular graph grammars have the property that each parse tree from the derivation of a graph lies embedded in the structure of the graph and can be extracted by an MSO-definable relation between the relational structures that represent graphs. Then, for each set of graphs defined by the grammar there exists an MSO formula that defines it. This formula is obtained from (i) the MSO definition of the set of parse trees, the existence of which is guaranteed by the alternation between recursive and non-recursive nonterminals in the derivation, and (ii) the definition (i.e., tuple of MSO formulæ) of the relation that extracts the parse tree from the graph. A natural question that arises is whether each set of graphs that is both MSO-definable and HR-context-free can be defined by a regular graph grammar.

*Our contributions* In this paper, we give this question a negative answer and suggest an alternative class that lies at the intersection of CMSO-definable and HR-context-free classes, by introducing *tree-verifiable grammars*. We demonstrate that tree-verifiable grammars strictly generalize the regular graph grammars of Courcelle, and establish a characterization result. We show that a language of graphs is CMSO-definable and of bounded embeddable tree-width, a notion we introduce below, if and only if this language can be generated by a tree-verifiable graph grammar.

The syntactic condition that defines tree-verifiable grammars is that each maps a nonterminal $u$ to a graph operation, which is either (A) a substitution described by a graph with a single terminal hyper-edge attached to an internal *root* node and several *future roots* that are distributed among the nonterminal edges (in this case $u$ is a non-recursive nonterminal), (B) a parallel composition of $u$ with several occurrences of the same non-recursive nonterminal, or (C) a parallel composition of non-recursive nonterminals ($u$ is a recursive nonterminal in the (B) and (C) cases).

We define the *embeddable tree-width* of a graph to be the minimal width of a tree decomposition whose backbone is isomorphic to some spanning tree of the graph. The embeddable tree-width of a graph is, in general, greater than its tree-width, i.e., there are graphs for which the optimal tree decomposition is not isomorphic to a spanning tree. We show that this stronger notion allows us to precisely capture the class of graphs generated by tree-verifiable grammars. We establish that a language of graphs is CMSO-definable and of bounded embeddable tree-width if and only if this language can be generated by a tree-verifiable graph grammar.

In the special case of trees (i.e., graphs obtained by restricting the parallel composition to graphs with one root source) tree-verifiable grammars define precisely the CMSO-definable unranked sets of trees. Moreover, if all the repetition counts from the rules (B) equal one, tree-verifiable grammars are the same as the MSO-definable unranked sets of trees. This settles positively another question relative to the definition of associative-commutative (AC) tree grammars of Courcelle [6, Definition 3.7], namely whether AC grammars define all the MSO-definable sets of unranked trees.

For space reasons, the proofs of the results stated in this paper are given in the technical report [4].

*Related work* The intersection of the CMSO-definable and HR-context-free sets of graphs has been investigated in the seminal paper of Courcelle [6] that introduced regular graph grammars [6, §5] and proves that the set of series-parallel (i.e., graphs with sources 1 and 2 that can be either serialized or composed in parallel) belongs to this class graphs [6, §6]. More recently, Doumane refined this result by developing a language of regular expressions that describes the class of CMSO-definable graphs that can be built using at most two sources [8]. An interesting question that remains open, for the time being, is whether the sets of graphs defined by these regular expressions have bounded embeddable tree-width. It is known that their optimal tree decompositions can be extracted from the structure of the graphs in a canonical fashion, but these canonical decompositions do not correspond to spanning trees.

Substructural logics [17] are first-order logics extended with a non-idempotent *separating conjunction* operator that decomposes logical structures. A distinguished substructural logic is Separation Logic (SL) [12, 18], that is interpreted over finite partial functions of fixed arity, called *heaps*. With heaps one can model certain graphs of bounded degree, i.e., where the number of edges attached to a vertex is bounded by a constant. Another model of composition, used in Graph Logic (GL) of Cardelli et al [3] or the Separation Logic of Relations [15], splits graphs into subgraphs with disjoint sets of edges. These logics are more general than SL, because they describe classes of graphs of unbounded degree (e.g., stars and unranked sets of trees). Used in combination with recursive definitions, the separating conjunction becomes a powerful means to describe shape-related graph properties (lists, trees, stars, etc.).

One of the first fragments of SL with recursive definitions having a decidable entailment problem used an ad-hoc translation into equivalent MSO formulæ, together with a syntactic guarantee of tree-width boundedness [11]. In subsequent work, a decision procedure, not based on MSO, for the entailment problem in this SL fragment was proposed and implemented [14, 13] and the complexity of this entailment problem was established [9, 16]. The ideas used in the definition of the SL fragment motivated the definition of tree-verifiable grammars in this paper. In particular, considering substitution operations described by graphs with a single edge that connects the root to the future roots is similar to the *connectivity* condition from [11]. Moreover, distributing the future roots among the nonterminals guarantees that each future root will eventually become the root of a subgraph, which corresponds to the *establishment* condition of [11]. These conditions guarantee that the SL-defined heaps of [11] have bounded embeddable tree-width. We note that the fragment of [11] is restricted to degree-bounded heaps and that [11] did not attempt to achieve any completeness resp. characterization result that corresponds to the proposed syntactic restriction.

## 2 Preliminaries

The set of natural numbers is denoted by $\mathbb{N}$. Given $i, j \in \mathbb{N}$, we write $[i, j]$ for the set $\{i, i+1, \ldots, j\}$, assumed to be empty if $i > j$. The cardinality of a finite set $A$ is denoted by $\mathrm{card}(A)$. For a set $A$, we denote by $\mathrm{pow}(A)$ its powerset, $A^0 \stackrel{\text{def}}{=} \{\epsilon\}$, $A^{i+1} \stackrel{\text{def}}{=} A^i \times A$, for all $i \geq 0$, $A^* \stackrel{\text{def}}{=} \bigcup_{i \geq 0} A^i$ and $A^+ \stackrel{\text{def}}{=} \bigcup_{i \geq 1} A^i$, where $\times$ is the Cartesian product and $\varepsilon$ denotes the empty sequence. Intuitively, $A^*$ (resp. $A^+$) denotes the set of possibly empty (resp. nonempty) sequences of elements from $A$. The length of a sequence $\alpha \in A^*$ is denoted by $\mathrm{len}(\alpha)$ and its $i$-th element by $\alpha_i$, for $i \in [1, \mathrm{len}(\alpha)]$. For a function $f : A \to B$, we denote its *domain* by $\mathrm{dom}(f) = A$ and its image by $\mathrm{img}(f) \stackrel{\text{def}}{=} \{b \in B \mid f(a) = b \text{ for some } a \in A\}$. The domain-restriction of a function $f$ to a set $C$ is denoted as $f{\downarrow}_C \stackrel{\text{def}}{=} \{(a, f(a)) \mid a \in C\}$.

*Graphs* in this paper are multigraphs consisting of edges of arity one or more. Let $\mathbb{A}$ be a finite alphabet of edge labels, ranged over by $a$ and equipped with arities $\#a \geq 1$.

**Definition 1.** *A* concrete graph (c-graph) of type $n \geq 0$ *is a tuple* $G = \langle V_G, E_G, \lambda_G, \upsilon_G, \xi_G \rangle$, *where:*

- $V_G$ *is a finite set of* vertices*,*
- $E_G$ *is a finite set of* edges*, such that* $V_G \cap E_G = \emptyset$*,*
- $\lambda_G : E_G \to \mathbb{A}$ *is a mapping that defines the labels of the edges,*
- $\upsilon_G : E_G \to V_G^+$ *is a mapping that associates each edge a nonempty sequence of vertices attached to the edge, such that* $\#(\lambda_G(e)) = \mathrm{len}(\upsilon_G(e))$*, for each* $e \in E_G$*,*
- $\xi_G : [1, n] \to V_G$ *is an injective mapping that designates* $n$ *distinct* sources*. The vertex* $\xi_G(i)$ *is the* $i$-th source *of* $G$*. A vertex* $v \in V_G$ *is* internal *if* $v \notin \mathrm{img}(\xi_G)$*. By* $\mathsf{intern}(G) = V_G \setminus \{\xi_G(i) \mid i \in [1, n]\}$ *we denote the internal nodes of* $G$*. For simplicity, we omit the sources for graphs of type* $0$*.*

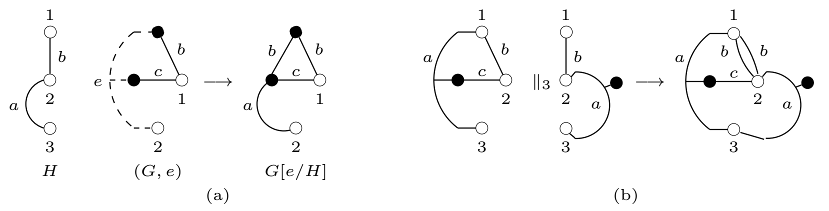

Figure 1: Graph operations: substitution (a) parallel composition (b)

*We denote by $\mathcal{G}^{\mathbb{A}}$ the set of c-graphs with edge labels from $\mathbb{A}$.*

For example, Fig. 1 (a,b) shows several c-graphs with (numbered) sources depicted in black and internal vertices in white.

Two c-graphs are *disjoint* iff their sets of vertices and edges are disjoint, respectively. Two c-graphs are *isomorphic* iff there exists a bijection between their vertices and edges that preserves the edge labels, the sequences of vertices attached to edges and the sources. A *graph* is the equivalence class of a c-graph for isomorphism. For any graph $G$ we denote by $\widehat{G}$ the graph of type 0 that is obtained from $G$ by removing its sources. A language $\mathcal{L}$ of graphs is some set of graphs. We say $\mathcal{L}$ is of type $n$ if all graphs $G \in \mathcal{L}$ are of type $n$. By $\widehat{\mathcal{L}} = \{\widehat{G} \mid G \in \mathcal{L}\}$ we denote the language of type 0 that is obtained by removing the sources of the graphs of $\mathcal{L}$.

*Graph Operations* If $G$ is a c-graph, $e \in E_G$ is an edge such that $\#\lambda_G(e) = m$ and $H$ is a c-graph of type $m$ disjoint from $G$, then the *substitution* $G[e/H]$ deletes the edge $e$ from $G$, adds $H$ to $G$ and joins the $i$-th vertex from $\upsilon_G(e)$ with the $i$-th source of $H$, for all $i \in [1,m]$. The sources and type of $G[e/H]$ are the same as for $G$. Fig. 1 (a) shows an example of substitution, formally defined below:

**Definition 2** (Substitution)**.** *Let $G$ be a c-graph of type $n$, $e \in E_G$ is an edge such that $\upsilon_G(e) = \langle v_1, \ldots, v_m\rangle$ and $H$ is a c-graph of type $m$ disjoint from $G$. Let $\sim\, \subseteq (V_G \cup V_H)^2$ be the least equivalence relation such that $v_i \sim \xi_H(i)$ for all $i \in [1,m]$. We denote by $[u]_\sim$ the $\sim$-equivalence class of $u \in V_G \cup V_H$. Then $G' = G[e/H]$ is defined by setting:*

- $V_{G'} = \{[u]_\sim \mid u \in V_G \cup V_H\}$,
- $E_{G'} = E_G \setminus \{e\} \cup E_H$,
- $\lambda_{G'} \stackrel{\text{def}}{=} (\lambda_G)\downarrow_{(E_G \setminus \{e\})} \cup \lambda_H$,
- $\upsilon_{G'}(e') \stackrel{\text{def}}{=} \langle [u_1]_\sim, \ldots, [u_k]_\sim\rangle$, *for all* $e' \in E_G \setminus \{e\}$ *with* $\upsilon_G(e') = \langle u_1, \ldots, u_k\rangle$, *resp. all* $e' \in E_H$ *with* $\upsilon_H(e') = \langle u_1, \ldots, u_k\rangle$,
- $\xi_{G'}(i) \stackrel{\text{def}}{=} [\xi_G(i)]_\sim$, *for all* $i \in [1,n]$.

*The substitution $G[e/H]$ is lifted from c-graphs to graphs (one can always chose c-graphs that are isomorphic to $G$ and $H$ and are disjoint). Then, the graph $G[e/H]$ is defined as the isomorphism class of the c-graph that results from the substitution involving these c-graphs.*

The graph operation $(G, e_1, \ldots, e_k)$, where $e_1, \ldots, e_k \in E_G$ are pairwise distinct edges, takes pairwise distinct c-graphs $H_1, \ldots, H_k$ of types $\#\lambda_G(e_1), \ldots, \#\lambda_G(e_k)$, respectively, and returns the c-graph $G[e_1/H_1, \ldots, e_k/H_k]$, where the substitutions can be done in any order. Again this operation can be lifted from concrete graphs to graphs.

We now introduce an important special case of a graph operation: The *parallel composition* $\|_n$ denotes the operation $(G, e_1, e_2)$, where $G$ has exactly $n$ vertices, all of which are sources, and $e_1$ and $e_2$ are the only edges of $G$, both of arity $n$ and attached to all $n$ sources. Intuitively, $G_1 \|_n G_2$ denotes the graphs that can be obtained from two graphs $G_1$ and $G_2$ of type $n$ by

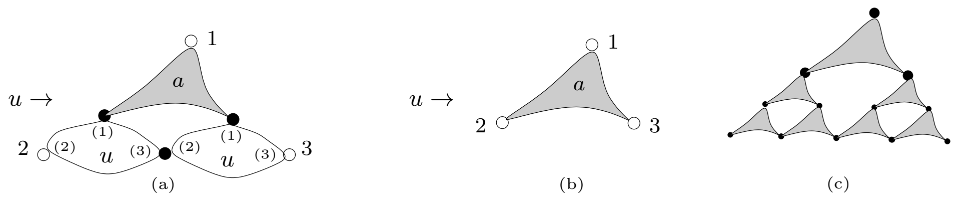

Figure 2: Graph Grammar for Trees with Linked Leaves

merging their sources. Fig. 1 (b) shows an example of parallel composition. Note that the composition operation $\|_n$ is both commutative and associative.

*Graph grammars* are pairs $\Gamma = (\mathcal{U}, \mathcal{R})$, where $\mathcal{U}$ is a set of *nonterminals* distinct from $\mathbb{A}$, ranged over $u, v, w$ with associated arities $\#u \geq 1$, and $\mathcal{R}$ is a finite set of rules of one of the following forms, with $u, v, w \in \mathcal{U}$:

1. $u \rightarrow v_1 \|_n \ldots \|_n v_k$, where $u, v_1, \ldots, v_k \in \mathcal{U}$ are nonterminals of arity $n$,
2. $u \rightarrow G[e_1(v_1), \ldots, e_k(v_k)]$, with $u, v_1, \ldots, v_k \in \mathcal{U}$ nonterminals and $G[e_1(v_1), \ldots, e_k(v_k)]$ denoting the graph operation $(G, e_1, \ldots, e_k)$, for some c-graph $G$ of type $\#u$, such that $\mathrm{img}(\lambda_G) \subseteq \mathbb{A} \cup \{v_1, \ldots, v_k\}$ and $\lambda_G(e) = v_i \iff e = e_i$, for all $e \in E_G$; if $\lambda_G(e) \in \mathcal{U}$ (resp. $\lambda_G(e) \in \mathbb{A}$), we say that $e$ is a nonterminal (resp. terminal) edge of $G$,
3. $\rightarrow u$, for some nonterminal $u \in \mathcal{U}$, is called an *axiom*.

We note that in rules of shape 1., we explicitly allow $k = 0$. With this we denote the rule $u \rightarrow \overline{\mathbf{1}}_n$, where $\overline{\mathbf{1}}_n$ denotes the graph of type $n$ that consists of $n$ vertices (all of which are sources), and that has no edges. The nonterminal on the left-hand side of $\rightarrow$ in a non-axiom rule is called the *head* of that rule. The *language of* $\Gamma$ *associated with nonterminal* $u$ is the set $\mathcal{L}_\Gamma(u)$ of graphs corresponding to $u$ in the unique least solution[1] of the system of recursive equations defined by the grammar rules, with nonterminals interpreted as sets, operations lifted to sets and $\rightarrow$ interpreted as the right-to-left set inclusion $\supseteq$. Note that each $\mathcal{L}_\Gamma(u)$ is a language of type $\#u$. The *language of* $\Gamma$, denoted by $\mathcal{L}(\Gamma)$, is the union of the sets $\widehat{\mathcal{L}_\Gamma(u)}$ for all axioms $\rightarrow u$ of $\Gamma$. Note that $\mathcal{L}(\Gamma)$ is a language of type 0. A set $\mathcal{L}$ of graphs is *hyperedge-replacement* (HR) iff $\mathcal{L} = \mathcal{L}(\Gamma)$ for some graph grammar $\Gamma$.

**Example 1.** *Fig. 2 shows a graph grammar with a single nonterminal $u$, of arity 3, and two rules. The first rule (a) is of the form $u \rightarrow G[e_1(u), e_2(u)]$, where $G$ is a c-graph with a (grey-filled) terminal edge labeled by $a$, where $\#a = 3$, and two nonterminal (unfilled) edges $e_1$ and $e_2$, both labeled by $u$. The order of the vertices attached to these edges is (1) middle, (2) left and (3) right. The second rule (b) is of the form $u \rightarrow G$, where $G$ consists of a single terminal edge labeled by $a$. The language of this grammar consists of all graphs of the form depicted in Fig. 2 (c).*

*Monadic Second Order Logic with Counting* (CMSO) is the set of formulæ written in the syntax from Fig. 3 (a) using a set $\mathcal{V} = \{x, y, \ldots\}$ of *first-order variables*, a set $\mathcal{X} = \{X, Y, \ldots\}$ of *second-order variables* and relation symbols $\mathsf{edg}_a$ of arity $\#a + 1$, for all $a \in \mathbb{A}$. By MSO we denote the subset of CMSO formulæ consisting of formulæ that do not contain atomic subformulæ of the form $\mathsf{card}_{q,p}(X)$, also called *cardinality constraints*. A variable is *free* if it occurs outside the scope of any quantifier. The set of free variables of a formula $\psi$ is denoted by $\mathrm{fv}(\psi)$. A *sentence* is a formula without free variables.

[1]By Tarski's fixpoint theorem, since the graph operations are monotonic (w.r.t. to inclusion of sets) and continuous (w.r.t. infinite unions of sets), the system of recursive equations defined by the grammar has a unique least solution.

$$\psi := x = y \mid \mathsf{edg}_a(x_1, \ldots, x_{\#a+1}) \mid \mathsf{card}_{q,p}(X) \mid X(x) \mid \neg\psi \mid \psi \wedge \psi \mid \exists x \,.\, \psi \mid \exists X \,.\, \psi$$

(a) $\mathsf{CMSO}$ Syntax

$$\begin{array}{lllcl}
G & \models^{\mathfrak{s}} & x = y & \iff & \mathfrak{s}(x) = \mathfrak{s}(y) \\
G & \models^{\mathfrak{s}} & \mathsf{edg}_a(x_1, \ldots, x_{\#a+1}) & \iff & \mathfrak{s}(x_1) \in E_G,\ \lambda_G(\mathfrak{s}(x_1)) = a \text{ and} \\
 & & & & \upsilon_G(\mathfrak{s}(x_1)) = \langle \mathfrak{s}(x_2), \ldots, \mathfrak{s}(x_{\#a+1}) \rangle \\
G & \models^{\mathfrak{s}} & \mathsf{card}_{q,p}(X) & \iff & \mathrm{card}(\mathfrak{s}(X)) = kq + p \text{ for some } k \in \mathbb{N} \\
G & \models^{\mathfrak{s}} & X(x) & \iff & \mathfrak{s}(x) \in \mathfrak{s}(X) \\
G & \models^{\mathfrak{s}} & \neg\phi & \iff & G \not\models^{\mathfrak{s}} \phi \\
G & \models^{\mathfrak{s}} & \phi_1 \wedge \phi_2 & \iff & G \models^{\mathfrak{s}} \phi_1 \text{ and } G \models^{\mathfrak{s}} \phi_2 \\
G & \models^{\mathfrak{s}} & \exists x \,.\, \psi & \iff & G \models^{\mathfrak{s}[x \leftarrow u]} \psi, \text{ for an element } u \in V_G \cup E_G \\
G & \models^{\mathfrak{s}} & \exists X \,.\, \psi & \iff & G \models^{\mathfrak{s}[X \leftarrow U]} \psi, \text{ for a set } U \subseteq V_G \cup E_G
\end{array}$$

(b) $\mathsf{CMSO}$ Semantics

Figure 3: Monadic Second Order Logic with Counting ($\mathsf{CMSO}$)

We define the semantics of $\mathsf{CMSO}$, for graphs of type 0, by a satisfaction relation $G \models^{\mathfrak{s}} \psi$ between graphs and formulæ, where the valuation $\mathfrak{s} : \mathcal{V} \cup \mathcal{X} \to V_G \cup E_G \cup \mathrm{pow}(V_G \cup E_G)$ maps each variable $x \in \mathcal{V}$ to a vertex or and edge, i.e., $\mathfrak{s}(x) \in V_G \cup E_G$, and each variable $X \in \mathcal{X}$ to a finite subset of vertices and edges, i.e., $\mathfrak{s}(X) \subseteq V_G \cup E_G$. The satisfaction relation is defined inductively on the structure of formulæ in Fig. 3 (b). If $\psi$ is a sentence, the satisfaction relation does not depend on the variable mapping and we write $G \models \psi$ instead of $G \models^{\mathfrak{s}} \psi$. A language $\mathcal{L}$ of graphs of type 0 is $\mathsf{CMSO}$-*definable* iff there exists a $\mathsf{CMSO}$ sentence $\psi$ such that $\mathcal{G} = \{G \mid G \models \psi\}$.

We define the formula $e \in E \overset{\mathsf{def}}{\iff} \bigvee_{a \in \mathbb{A}} \exists x_1, \ldots, x_{\#a}.\ \mathsf{edg}_a(e, x_1, \ldots, x_{\#a})$, which denotes that $e$ is an edge, and the formula $v \in V \overset{\mathsf{def}}{\iff} \bigvee_{a \in \mathbb{A}} \exists e, x_1, \ldots, x_{\#a}.\ \mathsf{edg}_a(e, x_1, \ldots, x_{\#a}) \wedge \bigvee_{1 \leq i \leq \#a} x_i = v$, which denotes that $v$ is a vertex. As usual we define $\phi \to \psi \overset{\mathsf{def}}{\iff} \neg\phi \vee \psi$ and $\phi \leftrightarrow \psi \overset{\mathsf{def}}{\iff} \phi \to \psi \wedge \psi \to \phi$. For $\Delta = E, V$, we further define $\forall x \in \Delta \,.\, \psi(x) \overset{\mathsf{def}}{\iff} \forall x \,.\, x \in \Delta \to \psi(x)$, $\exists x \in \Delta \,.\, \psi(x) \overset{\mathsf{def}}{\iff} \exists x \,.\, x \in \Delta \wedge \psi(x)$, $\exists! x \in \Delta \,.\, \psi(x) \overset{\mathsf{def}}{\iff} \exists x \in \Delta \,.\, \psi(x) \wedge \forall y \in \Delta \,.\, \psi(y) \to y = x$, and $\exists X \subseteq \Delta \,.\, \psi \overset{\mathsf{def}}{\iff} \exists X \,.\, (\forall x \,.\, X(x) \to x \in \Delta) \wedge \psi$.

*Recognizable Sets of Graphs.* A congruence relation on graphs is an equivalence relation $\cong$ such that (1) any two equivalent graphs are of the same type, and (2) for every graph $H$, for every edge $e$ of $H$, for every graph $G$ of type $\#(\lambda_H(e))$, and every $G' \cong G$, one has $H[e/G] \cong H[e/G']$. Such a congruence is called *locally-finite* iff it has finitely many equivalence classes of each type (by the definition of congruences, each equivalence class consists of graphs of the same type). A set of graphs $\mathcal{L}$ is *recognizable* iff there exists a locally-finite congruence $\cong$ such that $G \cong G'$ only if $G \in \mathcal{L}$ iff $G' \in \mathcal{L}$.

## 3 Tree Grammars

This section is concerned with unranked trees, whose nodes may have unbounded numbers of children, the order of which is not important. We introduce trees as a subset of the set of graphs of type 1, whose only source denotes the root. We consider the following restricted graph operations. Let $\overline{\mathbf{1}}$ be the graph that consists of a singleton vertex, which is also its only source, and that has no edges. For an edge label $a \in \mathbb{A}$ and $1 \leq i \leq \#a$ denote by $\overline{\mathbf{a}}^i$ the

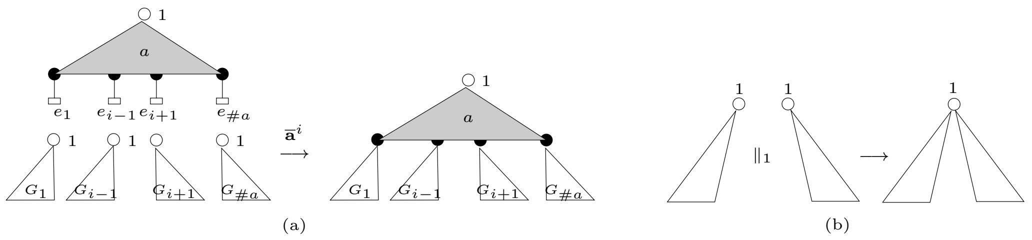

Figure 4: Tree operations: substitution (a) parallel composition (b)

operation $(G, e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{\#a})$, where $G$ is a c-graph that consists of a singleton edge $e$ labelled by $a$ attached to pairwise distinct vertices $v_1, \ldots, v_{\#a}$, such that $\xi_G(1) = v_i$ and the other vertices $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{\#a}$ are attached to distinct edges $e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{\#a}$ labeled with unary symbols, respectively. Intuitively, the graph $\overline{\mathbf{a}}^i(G_1, \ldots, G_{i-1}, G_{i+1}, \ldots, G_{\#a})$ is obtained by taking the disjoint union of the graphs $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots, G_{\#a}$ and adding a new edge labelled by $a$ with sequence of vertices $(\xi_{G_1}(1), \ldots, \xi_{G_{i-1}}(1), v, \xi_{G_{i+1}}(1), \ldots, \xi_{G_{\#a}}(1))$, where $v$ is a new vertex that becomes the only source of the resulting graph. This operation is depicted in Fig. 4 (a). The parallel composition operation for trees is $\|_1$, depicted in Fig. 4 (b).

The set of trees over $\mathbb{A}$ is defined as the least set closed under the operations $\overline{\mathbf{1}}$, $\|_1$ and $\overline{\mathbf{a}}^i$, for all $a \in \mathbb{A}$ and $1 \leq i \leq \#a$. We retrieve the standard terminology for trees from this definition. The vertices of a tree $T$ are called *nodes*. For an edge $e \in E_T$ introduced by an operation $\overline{\mathbf{a}}^i$, we say that $\upsilon_T(i)$ is the *parent* of $\upsilon_T(1), \ldots, \upsilon_T(i-1), \upsilon_T(i+1), \ldots, \upsilon_T(\#a)$ and the latter nodes are the *children* of the former. Note that the trees we consider in this paper are *unordered*, i.e., the relative order of the edges attached to a node is not important, and *unranked*, i.e., there is no bound on the number of children of a node.

We are interested in HR sets of trees, generated by *tree grammars*, which are graph grammars, whose rules use only tree operations. In general, HR sets of trees are not CMSO-definable. For instance, the grammar with rules $u \to \overline{\mathbf{a}}^1 \|_1 \overline{\mathbf{b}}^1 \|_1 u$ and $u \to \overline{\mathbf{1}}$, where $a$ and $b$ are binary edge labels, defines the set of trees of height one, whose root has the same number of $a$- and $b$-labeled edges. Since CMSO cannot describe sets of equal cardinality (see, e.g., [5, Proposition 3.9]), this set is not CMSO-definable.

This motivates the quest for trees grammars defined by easy-to-check syntactic restrictions, that guarantee the CMSO-definability of their languages. The following definition introduces such a class of tree grammars. In the rest of this section, we prove that the languages of the grammars that conform with the following definition coincide with the CMSO-definable tree languages.

**Definition 3.** *A tree grammar* $\Gamma = (\mathcal{U}, \mathcal{R})$ *is* regular *iff there exists a set* $\mathcal{W} \subseteq \mathcal{U}$ *of nonterminals such that the rules in* $\mathcal{R}$ *are of one of the following forms, either:*

*A.* $w \to \overline{\mathbf{a}}^i(u_1, \ldots, u_{\#a-1})$*, for some* $a \in \mathbb{A}$ *and* $1 \leq i \leq \#a$*, where* $w \in \mathcal{W}$ *and* $u_1, \ldots, u_{\#a-1} \in \mathcal{U}$*,*

*B.* $u \to u \|_1 w^q$*, for some* $q \in \mathbb{N}$*, where* $u \in \mathcal{U} \setminus \mathcal{W}$ *and* $w \in \mathcal{W}$*,*

*C.* $u \to w_1 \|_1 \cdots \|_1 w_k$*, where* $u \in \mathcal{U} \setminus \mathcal{W}$ *and* $w_1, \ldots, w_k \in \mathcal{W}$*; this rule becomes* $u \to \overline{\mathbf{1}}$*, in case* $k = 0$*.*

*We further require that the axioms are of the form* $\to u$*, for some* $u \in \mathcal{U} \setminus \mathcal{W}$*. We call the grammar* $\Gamma$ aperiodic-regular *iff* $q = 1$ *in all rules of the forms B.*

We call a set $\mathcal{L}$ of trees *regular* resp. *aperiodic-regular* iff there is a regular resp. aperiodic-regular tree-grammar $\Gamma$ such that $\mathcal{L} = \mathcal{L}(\Gamma)$. The shape of regular tree-grammars as given in Definition 3 is inspired by the AC-tree grammars introduced by Courcelle [6, Section 3]. However, Courcelle's definition of AC-grammars only considers grammars with $q = 1$ in all rules of the form rules B in Definition 3 (which correspond to the aperiodic-regular grammars in our terminology). In this section, we show that the aperiodic-regular tree languages coincide with the MSO-definable tree languages. Hence, the AC-grammars considered by Courcelle are not complete wrt CMSO-definability, because there are tree languages that are CMSO-definable but not MSO-definable, see e.g., [5, Corollary 6.6].

First, we prove that the CMSO-definable tree languages are exactly the regular tree languages. For this we will make use of the following lemma:

**Lemma 1.** *Given a regular tree grammar $\Gamma = (\mathcal{U}, \mathcal{R})$ with a set $\mathcal{W}$ as in Definition 3, we can build a regular tree grammar $\Gamma' = (\mathcal{U}', \mathcal{R}')$ with $\mathcal{L}(\Gamma) = \mathcal{L}(\Gamma')$, such that, for each pair of non-terminals $u \in \mathcal{U} \setminus \mathcal{W}$ and $w \in \mathcal{W}$, there is at most one rule $u \rightarrow u \parallel_1 w^q$ of shape B in $\mathcal{R}'$ .*

The first direction of the equivalence between regular and CMSO-definable tree languages is given by the following theorem:

**Theorem 1.** *Let $\mathcal{L}$ be a regular tree language. Then, $\mathcal{L}$ is* CMSO*-definable.*

In order to establish the completeness of regular tree grammars, we use the already known equivalence between CMSO-definable and recognizable languages of unranked trees [5, Theorem 5.3]:

**Theorem 2** ([5])**.** *Let $\mathcal{L}$ be a language of trees. Then, $\mathcal{L}$ is* CMSO*-definable iff $\mathcal{L}$ is recognizable.*

The dual direction of the equivalence between regular tree languages and CMSO-definable tree languages is given by the following result:

**Theorem 3.** *Let $\mathcal{L}$ be a* CMSO*-definable tree language. Then, $\mathcal{L}$ is a regular tree language.*

In the rest of this section, we characterize the class of MSO-definable tree languages by tree grammars. We show that the MSO-definable tree languages are exactly the languages produced by aperiodic-regular tree grammars.

**Theorem 4.** *Let $\mathcal{L}$ be an aperiodic-regular tree language. Then, $\mathcal{L}$ is MSO-definable.*

**Theorem 5.** *Let $\mathcal{L}$ be a MSO-definable tree language. Then, there exists an aperiodic-regular tree grammar $\Gamma$ with $\mathcal{L} = \mathcal{L}(\Gamma)$.*

We summarize our results on trees in the following corollary:

**Corollary 1.** *Let $\mathcal{L}$ be a tree language. Then, the following are equivalent:*
- *$\mathcal{L}$ is* CMSO*-definable.*
- *$\mathcal{L}$ is regular, i.e., there is some regular tree grammar $\Gamma$ with $\mathcal{L} = \mathcal{L}(\Gamma)$.*

*Moreover, the following are equivalent:*
- *$\mathcal{L}$ is MSO-definable.*
- *$\mathcal{L}$ is aperiodic-regular, i.e., there is some aperiodic-regular tree grammar $\Gamma$ with $\mathcal{L} = \mathcal{L}(\Gamma)$.*

## 4 Tree Verifiable Graph-Grammars

In this section, we introduce a syntactic restriction on HR graph grammars, that ensures the CMSO-definability of the generated graph languages. The restriction is given in terms of easy-to-check syntactic conditions on the rules of the HR grammar. Like in the case of regular tree grammars, we partition the set of nonterminals in two alternating subsets. The crucial restriction is then on the shape of the graphs $G$ in the rules of the form $w \leftarrow G[e_1(v_1), \ldots, e_k(v_k)]$, where we require $G$ to consist of a single terminal edge $e$, which needs to contain at least some source and some vertex of each $e_i$. Some additional conditions ensure that these edges and vertices give rise to a spanning tree of the generated graph that witnesses the bounded embeddable tree-width of the graph (see Definition 6 and Proposition 1 below). The spanning tree property also underlies our proof of CMSO-definability, which motivates the term *tree-verifiable*. As we shall argue next (§5), tree-verifiable grammars define an expressive class of graph languages, as they strictly subsume the regular graph grammars of Courcelle [6, §5].

**Definition 4.** *A graph grammar $\Gamma = (\mathcal{U}, \mathcal{R})$ is* tree-verifiable *iff there exist two functions:*

- $\mathsf{rt} : \mathcal{U} \rightarrow \mathbb{N}$ *mapping each non-terminal $u$ to an index $\mathsf{rt}(u) \in [1, \#u]$, called the* root *of $u$,*
- $\mathsf{fut} : \mathcal{U} \rightarrow \mathrm{pow}(\mathbb{N})$ *mapping each non-terminal $u$ to a subset $\mathsf{fut}(u) \subseteq [1, \#u] \setminus \{\mathsf{rt}(u)\}$, called the* future roots *of $u$,*

*and a set $\mathcal{W} \subseteq \mathcal{U}$ of nonterminals, such that the rules in $\mathcal{R}$ are of one of the following forms, either:*

*A. $w \rightarrow G[e_1(u_1), \ldots, e_k(u_k)]$, for some $w \in \mathcal{W}$ and $u_1, \ldots, u_k \in \mathcal{U} \setminus \mathcal{W}$, where $G$ has exactly one terminal edge $e$, the vertices $\{\xi_G(\mathsf{rt}(w))\} \cup \{\upsilon_G(e_i)_{\mathsf{rt}(u_i)}\}_{i \in [1,k]}$ are pairwise distinct and attached to $e$, and*

$$\mathsf{intern}(G) \cup \{\xi_G(i) \mid i \in \mathsf{fut}(w)\} = \biguplus_{i=1..k} \{\upsilon_G(e_i)_j \mid j \in \{\mathsf{rt}(u_i)\} \cup \mathsf{fut}(u_i)\}$$

*i.e., the internal nodes of $G$ plus the future roots associated to $w$ can be partitioned into the roots and future roots associated to all the non-terminals $u_i$.*

*B. $u \rightarrow u \parallel_n w^q$, for some $q \in \mathbb{N}$, where $u \in \mathcal{U} \setminus \mathcal{W}$, $w \in \mathcal{W}$ with $\#u = \#w = n$, such that $\mathsf{rt}(w) = \mathsf{rt}(u)$ and $\mathsf{fut}(w) = \emptyset$, i.e., the roots of $u$ and $w$ agree and, moreover, $w$ has no future roots,*

*C. $u \rightarrow w_1 \parallel_n \cdots \parallel_n w_k$, where $u \in \mathcal{U} \setminus \mathcal{W}$, $w_1, \ldots, w_k \in \mathcal{W}$ with $\#u = \#w_i = n$, such that $\mathsf{rt}(u) = \mathsf{rt}(w_i)$ and $\mathsf{fut}(u) = \biguplus_{i=1..k} \mathsf{fut}(w_i)$, i.e., the roots associated to the non-terminals of the right-hand side agree with the root of the left-hand side and that future roots of the left-hand side are partitioned into the future roots of the non-terminals of the right-hand side.*

*Furthermore, the axioms are of the shape $\rightarrow u$ for some $u \in \mathcal{U} \setminus \mathcal{W}$ with $[1, \#u] = \{\mathsf{rt}(u)\} \cup \mathsf{fut}(u)$, i.e., for every start non-terminal $u$ every source must be a root or future root.*

A set of graphs $\mathcal{L}$ is said to be *tree-verifiable* iff there is a tree-verifiable graph-grammar $\Gamma$ such that $\mathcal{L} = \mathcal{L}(\Gamma)$. The following example states a tree-verifiable grammar for the trees with linked leaves from Example 1 plus an additional binary edge from the bottom-right node of the tree to the root of the tree (see the left of Fig. 6 for the depiction of such a graph)[2]:

**Example 2.** *Let us consider the tree-verifiable grammar from Fig. 5, with the set of non-terminals $\mathcal{U} = \{u_{axiom}, v_{axiom}, z_{axiom}, w_{axiom}, u, v, w\}$ and distinguished non-terminals $\mathcal{W} = \{v_{axiom}, w_{axiom}, w\}$, with arities, roots and future roots, as follows:*

- $\#u_{axiom} = \#v_{axiom} = 1$, $\mathsf{rt}(u_{axiom}) = \mathsf{rt}(v_{axiom}) = 1$, $\mathsf{fut}(u_{axiom}) = \mathsf{fut}(v_{axiom}) = \emptyset$,
- $\#z_{axiom} = \#w_{axiom} = 2$, $\mathsf{rt}(z_{axiom}) = \mathsf{rt}(w_{axiom}) = 1$, $\mathsf{fut}(z_{axiom}) = \mathsf{fut}(w_{axiom}) = \emptyset$,
- $\#u = \#w = 3$, $\mathsf{rt}(u) = \mathsf{rt}(w) = 1$, $\mathsf{fut}(u) = \mathsf{fut}(w) = \{2\}$,

[2]We note that we chose to add the binary edge as the easiest way of setting up a tree-verifiable grammar (with axioms) that meets the requirements of Def. 4. However, one can also state a slightly more complicated tree-verifiable grammar that exactly produces the trees with linked leaves from Example 1.

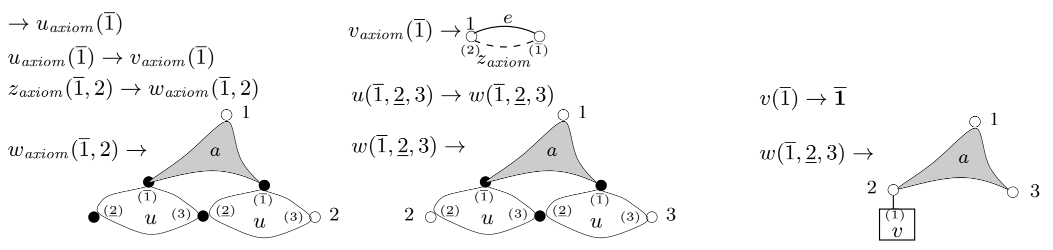
$\to u_{axiom}(\overline{1})$
$u_{axiom}(\overline{1}) \to v_{axiom}(\overline{1})$
$z_{axiom}(\overline{1},2) \to w_{axiom}(\overline{1},2)$
$w_{axiom}(\overline{1},2) \to$
$v_{axiom}(\overline{1}) \to$
$z_{axiom}$
$u(\overline{1},\underline{2},3) \to w(\overline{1},\underline{2},3)$
$w(\overline{1},\underline{2},3) \to$
$v(\overline{1}) \to \mathbf{1}$
$w(\overline{1},\underline{2},3) \to$
a
u
e
v

Figure 5: Tree-Verifiable Grammar for Trees with Linked Leaves

- $\#v = 1$, $\mathsf{rt}(v) = 1$, $\mathsf{fut}(v) = \emptyset$.

*We write the indices $1, \ldots, \#n$ enclosed in parenthesis next to each nonterminal $n \in \mathcal{U}$, where $\mathsf{rt}(n)$ is overlined and $\mathsf{fut}(n)$ are underlined. Internal nodes are drawn as black circles and sources are white numbered circles. Binary terminal edges are drawn with solid lines and binary nonterminal edges are dashed. Ternary terminal edges are filled with grey and ternary nonterminal edges are unfilled.*

The first result is that tree-verifiable graph grammars produce $\mathsf{CMSO}$-definable sets of graphs:

**Theorem 6.** *Let $\mathcal{L}$ be a tree-verifiable graph language. Then, $\mathcal{L}$ is $\mathsf{CMSO}$-definable.*

The rest of this section is concerned with proving a partial completeness result. We define a parameter, called *embeddable tree-width*, which is an over-approximation of the standard tree-width of a graph. We show that the class of $\mathsf{CMSO}$-definable sets of graphs, for which the embeddable tree-width is bounded, coincides with the class of tree-verifiable graph languages.

Before defining embeddable tree-width formally, we recall the standard definitions of *tree decomposition* and *treewidth* of a c-graph, by considering trees over an alphabet of edge labels that consists of a single binary element. A *path* from $v_1$ to $v_{k+1}$ in a c-graph $G$ is a sequence of tuples of the form:

$$\langle v_1, i_1, e_1, v_2, j_1 \rangle, \langle v_2, i_2, e_2, v_3, j_2 \rangle, \ldots, \langle v_k, i_k, e_k, v_{k+1}, j_k \rangle$$

where $v_1, \ldots, v_{k+1} \in V_G$, $e_1, \ldots, e_k \in E_G$ and $v_\ell$ and $v_{\ell+1}$ are the $i_\ell$-th and $j_\ell$-th vertices attached to $e_\ell$, respectively, for all $\ell \in [1, k]$. A set $C \subseteq V_G$ is *connected* in $G$ iff for any $u, v \in C$ there exists a path from $u$ to $v$ in $G$ containing only vertices from $C$.

**Definition 5.** *A* tree decomposition *of a c-graph $G$ of type $n$ is a pair $(T, \beta)$, where $T$ is a tree, called* backbone, *and $\beta : V_T \to \mathrm{pow}(V_G)$ maps the vertices of $T$ into sets of vertices from $G$, such that:*

1. *for each $e \in E_G$, $\upsilon_G(e) = \langle v_1, \ldots, v_k \rangle$, there exists $n \in V_T$ such that $v_1, \ldots, v_k \in \beta(n)$,*
2. *for each $v \in V_G$ the set $\{n \in V_T \mid v \in \beta(n)\}$ is nonempty and connected in $T$.*

*The* width *of the tree decomposition is $\mathrm{wd}(T, \beta) \overset{\mathsf{def}}{=} \max_{n \in V_T} \mathrm{card}(\beta(n)) - 1$. The* tree-width *of $G$ is $\mathrm{tw}(G) \overset{\mathsf{def}}{=} \min\{\mathrm{wd}(T, \beta) \mid (T, \beta)$ tree decomposition of $G\}$. The tree-width of a graph is the tree-width of some c-graph from it[3]. A set $\mathcal{G}$ of graphs has* bounded tree-width *iff $\{\mathrm{tw}(G) \mid G \in \mathcal{G}\}$ is a finite set.*

We refine the above definition by considering only backbones that are spanning trees of the graph. The spanning trees considered cover both the vertices and the edges of the graph:

[3]Isomorphic c-graphs have equal tree-widths.

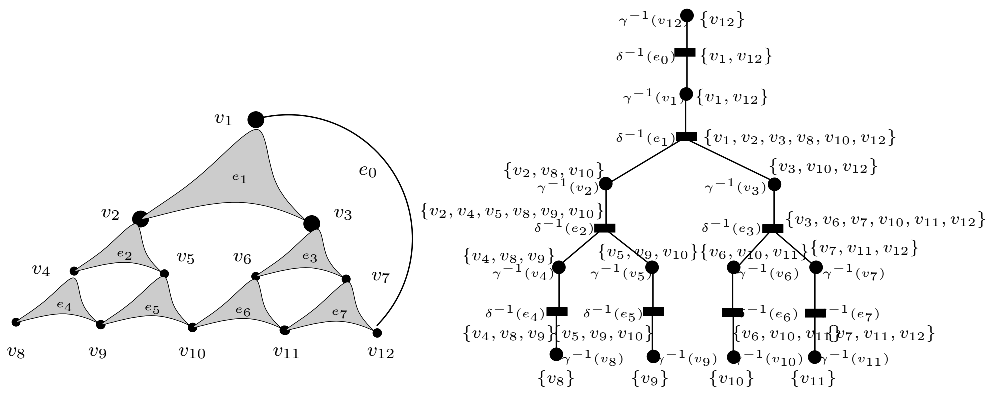
$v_1$
$e_1$
$e_0$
$v_2$
$v_3$
$e_2$
$e_3$
$v_4$
$v_5$
$v_6$
$v_7$
$e_4$
$e_5$
$e_6$
$e_7$
$v_8$
$v_9$
$v_{10}$
$v_{11}$
$v_{12}$
$\gamma^{-1}(v_{12})$ $\{v_{12}\}$
$\delta^{-1}(e_0)$ $\{v_1, v_{12}\}$
$\gamma^{-1}(v_1)$ $\{v_1, v_{12}\}$
$\delta^{-1}(e_1)$ $\{v_1, v_2, v_3, v_8, v_{10}, v_{12}\}$
$\{v_2, v_8, v_{10}\}$ $\gamma^{-1}(v_2)$
$\{v_3, v_{10}, v_{12}\}$ $\gamma^{-1}(v_3)$
$\{v_2, v_4, v_5, v_8, v_9, v_{10}\}$ $\delta^{-1}(e_2)$
$\delta^{-1}(e_3)$ $\{v_3, v_6, v_7, v_{10}, v_{11}, v_{12}\}$
$\{v_4, v_8, v_9\}$ $\gamma^{-1}(v_4)$
$\{v_5, v_9, v_{10}\}$ $\gamma^{-1}(v_5)$
$\{v_6, v_{10}, v_{11}\}$ $\gamma^{-1}(v_6)$
$\{v_7, v_{11}, v_{12}\}$ $\gamma^{-1}(v_7)$
$\delta^{-1}(e_4)$ $\delta^{-1}(e_5)$ $\delta^{-1}(e_6)$ $\delta^{-1}(e_7)$
$\{v_4, v_8, v_9\}$ $\{v_5, v_9, v_{10}\}$ $\{v_6, v_{10}, v_{11}\}$ $\{v_7, v_{11}, v_{12}\}$
$\gamma^{-1}(v_8)$ $\gamma^{-1}(v_9)$ $\gamma^{-1}(v_{10})$ $\gamma^{-1}(v_{11})$
$\{v_8\}$ $\{v_9\}$ $\{v_{10}\}$ $\{v_{11}\}$

Figure 6: Embeddable Tree Decomposition

**Definition 6.** *An* embeddable *tree-decomposition of a c-graph $G$ is a tuple $(T, \beta, \gamma, \delta)$ such that $(T, \beta)$ is a tree-decomposition of $G$, and there is some subset $W \subseteq V_T$ of the nodes of $T$ such that:*

- *the root of $T$ does not belong to $W$,*
- *for every node $u \in V_T$ and every child $w \in V_T$ of $u$, we have $w \in W$ iff $u \notin W$, i.e., the membership in $W$ alternates for the nodes on every path of $T$,*

*Moreover, $\gamma : V_T \setminus W \to V_G$ is a bijective mapping of the nodes from $V_T \setminus W$ to the vertices of $G$, and $\delta : W \to E_G$ is a bijective mapping of the vertices in $W$ to the edges of $G$, satisfying the conditions:*

1. *for all $u \in V_T \setminus W$ we have that $\gamma(u) \in \beta(u)$, and*
2. *for all $w \in W$ and $e = \delta(w)$ we have*
   - *if $w$ is the child of $u$ in $T$ then there is some $i \in [1, \#\lambda_G(e)]$ such that $\upsilon_G(e)_i = \gamma(u)$,*
   - *for every child $v$ of $w$ in $T$ there is a distinct $j \in [1, \#\lambda_G(e)] \setminus \{i\}$ such that $\upsilon_G(e)_j = \gamma(v)$,*
   - *$\upsilon_G(e)_i \in \beta(w)$ for all $1 \leq i \leq \mathrm{len}(\upsilon_G(e))$.*

*The* embeddable tree-width *of $G$ is:*

$$\mathrm{etw}(G) \stackrel{\mathsf{def}}{=} \min\{\mathrm{wd}(T, \beta) \mid (T, \beta, \gamma, \delta) \textit{ is an embeddable tree decomposition of } G\}$$

*The embeddable tree-width of a graph is the embeddable tree-width of some c-graph from it. A set $\mathcal{G}$ of graphs has* bounded embeddable tree-width *iff $\{\mathrm{etw}(G) \mid G \in \mathcal{G}\}$ is a finite set.*

For example, Fig. 6 shows a graph that is generated by the tree-verifiable grammar from Example 2 and an embeddable tree decomposition of this graph (this tree decomposition is the tree decomposition that underlies the proof of Proposition 1). The nodes from the inverse image of $\gamma$ (resp. $\delta$) are depicted by filled circles (resp. rectangles). The sets in the image of $\beta$ (called *bags*) associated with the nodes of the tree decomposition are stated next to the tree nodes.

The following variation of Def. 6 will be useful for working with the notion of an embeddable tree-decomposition in analyzing graph grammars and for conducting inductive proofs about graph grammars:

**Definition 7.** *Let $G$ be c-graph $G$ of type $n$ and let $z \in [1, n]$ and $\pi \subseteq [1, n] \setminus \{z\}$. We consider a tuple $(T, \beta, \gamma, \delta)$ and subset $W \subseteq V_T$ such that the conditions of Def. 6 are satisfied, except for the bijectivity requirement on $\gamma$, and the requirement that the root of $T$ does not belong to*

*$W$. Let $r$ be the root of $T$. We now call $(T, \beta, \gamma, \delta)$ a $(z, \pi)$-*embeddable tree-decomposition *of $G$ iff the following hold:*

1. *$\xi_G(i) \in \beta(r)$ for all $i \in [1, n]$,*
2. *$\gamma : V_T \setminus W \to \mathsf{intern}(G) \cup A$ is a bijective mapping of the vertices in $V_T \setminus W$ to the internal vertices of $G$ plus the set $A = \pi$ in case of $r \in W$ and $A = \{z\} \cup \pi$ otherwise,*
3. *if $r \in W$ then there is some $i \in [1, \#\lambda_G(\delta(r))]$ such that $\upsilon_G(\delta(r))_i = \xi_G(z)$ and $\gamma(u) \neq \xi_G(z)$ for every child $u$ of $r$,*
4. *else ($r \in V_T \setminus W$) we have that $\gamma(r) = \xi_G(z)$.*

*We call the tree-decomposition* pointed*, if $r \in W$, and* non-pointed*, otherwise.*

To begin with, we note that tree-verifiable graph grammars are set up such that the resulting languages always have bounded embeddable tree-width:

**Proposition 1.** *Let $\mathcal{L}$ be a tree-verifiable set of graphs. Then, $\mathcal{L}$ has bounded embeddable tree-width. In more detail, given a tree-verifiable graph-grammar $\Gamma = (\mathcal{U}, \mathcal{R})$, let $k$ be the maximum of $\mathrm{card}(V_G)$ for the graphs $G$ appearing in $\mathcal{R}$ of shape A and the maximum of $\#u$ for all non-terminals $u \in \mathcal{U}$; then we have $\mathrm{etw}(G) \leq k$ for all $G \in \mathcal{L}(\Gamma)$.*

We now prove the existence of a tree-verifiable grammar that generates all graphs with embeddable treewidth $\leq n$ for some given $n \in \mathbb{N}$. While the existence of such a grammar is an interesting result on its own, we will build on this result in our partial completeness proof (Theorem 7), where we pair the non-terminals of this grammar with the equivalence classes of some congruence relation.

**Proposition 2.** *For each finite alphabet $\mathbb{A}$ of edge labels and each $n \in \mathbb{N}$, there exists a tree-verifiable grammar $\Gamma$, such that $\mathcal{L}(\Gamma) = \{G \in \mathcal{G}^{\mathbb{A}} \mid G$ has type $0$ and $\mathrm{etw}(G) \leq n\}$.*

The following result combines ideas from Theorem 3 and Proposition 2:

**Theorem 7.** *Let $\mathcal{L}$ be a $\mathsf{CMSO}$-definable graph language with bounded embeddable tree-width. Then, $\mathcal{L}$ is a tree-verifiable graph-language.*

We are ready to state the main result of this paper:

**Corollary 2.** *Let $\mathcal{L}$ be a graph language of type $0$. Then, the following are equivalent:*

- *$\mathcal{L}$ has bounded embeddable tree-width and is $\mathsf{CMSO}$-definable.*
- *$\mathcal{L}$ is tree-verifiable.*

## 5 Comparison with Regular Graph Grammars

In this section we compare our tree-verifiable grammar to the regular graph-grammars of Courcelle [6]. We will establish that the tree-verifiable grammars strictly generalize regular graph grammars. We will first demonstrate that for every regular graph grammar there is a tree-verifiable grammar that generates the same graph language. We will then show that cycles are tree-verifiable but not regular, see Example 3. We begin by recalling the *regular graph grammars* of Courcelle [6], under a slightly different notation:

**Definition 8** ([6])**.** *A graph operation $(G, e_1, \ldots, e_k)$ is* regular *iff the following hold:*

1. *$G$ has at least one edge, either (a) a single terminal edge attached to sources only, or (b) each of its edges is attached to an internal vertex, and*
2. *between any two vertices of $G$ there is a path that traverses only internal vertices (excepting the endpoints) and terminal edges.*

*A graph grammar $\Gamma = (\mathcal{U}, \mathcal{R})$ is* regular *iff there exists a set $\mathcal{W} \subseteq \mathcal{U}$ of nonterminals and the rules in $\mathcal{R}$ are of one of the following forms, either:*

*A. $w \rightarrow G[e_1(u_1), \ldots, e_k(u_k)]$, for some nonterminals $w \in \mathcal{W}$, $u_1, \ldots, u_k \in \mathcal{U} \setminus \mathcal{W}$ and some regular graph operation $(G, e_1, \ldots, e_k)$,*

*B. $u \rightarrow u \parallel_n w^q$, for some $q \in \mathbb{N}$, where $u \in \mathcal{U} \setminus \mathcal{W}$ and $w \in \mathcal{W}$, such that $\#u = \#w = n$,*

*C. $u \rightarrow w_1 \parallel_n \cdots \parallel_n w_k$, $u \in \mathcal{U} \setminus \mathcal{W}$ and $w_1, \ldots, w_k \in \mathcal{W}$, such that $\#u = \#w_i = n$.*

*We further require that axioms are of the shape $\rightarrow u$ for some $u \in \mathcal{U} \setminus \mathcal{W}$, and that $\#u > 1$ implies that $k \geq 1$ for all rules $u \rightarrow w_1 \parallel_n \cdots \parallel_n w_k$ with head $u$.*

We note that the restriction that $\rightarrow u$ and $\#u > 1$ imply $k \geq 1$ for all rules $u \rightarrow w_1 \parallel_n \cdots \parallel_n w_k$ with head $u$ is not part of the definition by Courcelle [6]. This only excludes a constant number of graphs, namely the graphs of $n$ isolated vertices, for which there is an axiom $u \rightarrow \overline{\mathbf{1}}_n$, with $n = \#u > 1$, and a rule $\rightarrow u$. We make this assumption in order to facilitate the comparison with tree-verifiable graph grammars, which always generate connected graphs. We note, however, that we could alternatively have weakened the definition of tree-verifiable graph grammars in order to allow for the generation of the graphs $\overline{\mathbf{1}}_n$.

Courcelle established the CMSO-definability of regular graph grammars, see [6, Theorems 4.8 and 5.10]. We now show that tree-verifiable graph grammars can simulate regular graph grammars (see Theorem 8 below). We note that the simulation result together with Theorem 6 also establishes the CMSO-definability of regular graph grammars. For the purposes of the upcoming proofs, we introduce the following simplified form of regular graph grammars:

**Lemma 2.** *For every regular graph grammar $\Gamma$ there is a regular graph grammar $\Gamma'$ with $\mathcal{L}(\Gamma) = \mathcal{L}(\Gamma')$ and $\#u = 1$ for all axioms $\rightarrow u$ of $\Gamma'$.*

**Theorem 8.** *For every regular graph grammar $\Gamma$ there is a tree-verifiable graph grammar $\Gamma'$ such that $\mathcal{L}(\Gamma) = \mathcal{L}(\Gamma')$.*

Next, we prove that tree-verifiable grammars strictly subsume regular graph grammars. Given a graph $G$, we recall that a set of vertices $C \subseteq V_G$ is connected in $G$ iff between any two vertices in $C$ there exists a path in $G$ that traverses only vertices from $C$. If $V_G$ is connected in $G$, we say that $G$ is *connected*. A graph is *disconnected* iff it is not connected. A *cut* $C \subseteq V_G$ of a connected graph $G$ is a set of vertices, such that the graph obtained by removing from $G$ the vertices in $C$ and the edges incident to some vertex from $C$ is disconnected.

**Proposition 3.** *Let $\Gamma$ be a regular graph grammar. Then, there is a constant $n \in \mathbb{N}$ such that every graph $G \in \mathcal{L}(\Gamma)$ with $\mathrm{card}(V_G) \geq n$ has a cut $C \subseteq V_G$ that is connected in $G$.*

The following example shows a graph language consisting of simple cycles of arbitrary size. These graphs do not have connected cuts, because by removing any sequence of one or more consecutive vertices in a cycle we always obtain a connected acyclic path.

**Example 3.** *We define cycles by the tree-verifiable grammar from Fig. 7, with the set of non-terminals $\mathcal{U} = \{u_{axiom}, w_{axiom}, u, w\}$ and distinguished non-terminals $\mathcal{W} = \{w_{axiom}, w\}$, with arities, roots and future roots, as follows:*

- *$\#u_{axiom} = \#w_{axiom} = 1$, $\mathsf{rt}(u_{axiom}) = \mathsf{rt}(w_{axiom}) = 1$, $\mathsf{fut}(u_{axiom}) = \mathsf{fut}(w_{axiom}) = \emptyset$,*
- *$\#u = \#w = 2$, $\mathsf{rt}(u) = \mathsf{rt}(w) = 1$, $\mathsf{fut}(u) = \mathsf{fut}(w) = \emptyset$.*

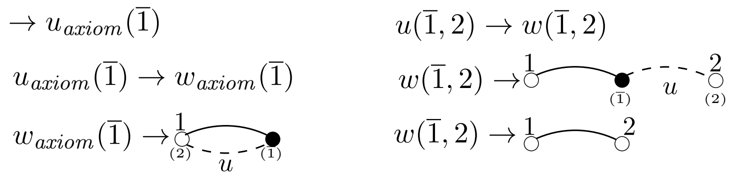
$\to u_{axiom}(\overline{1})$
$u_{axiom}(\overline{1}) \to w_{axiom}(\overline{1})$
$w_{axiom}(\overline{1}) \to$
1
(2)
(1)
$u$
$u(\overline{1},2) \to w(\overline{1},2)$
$w(\overline{1},2) \to$
1
($\overline{1}$)
$u$
2
(2)
$w(\overline{1},2) \to$
1
2

Figure 7: Tree-Verifiable Grammar for Cycles

## 6 Conclusions

We introduced the notion of tree-verifiable grammars, a class of hyperedge-replacement grammars that produce CMSO-definable languages. Since CMSO-definable context-free graph languages come with a decidable inclusion problem, the class of tree-verifiable graph languages is a useful tool for reasoning about and verifying systems that can be modeled using graphs, such as, e.g., distributed networks. We characterize tree-verifiable languages by showing their equivalence with the class of CMSO-definable graph languages that have bounded embeddable tree-width, a novel parameter, which strengthens the standard graph-theoretic tree-width. Finally, we show that the class of tree-verifiable graph languages strictly subsumes the languages produced by the regular graph grammars introduced by Courcelle, providing a more expressive class of languages at the intersection of the hyperedge-replacement context-free and the CMSO-definable sets of graphs.

## References

[1] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 2000.

[2] J. Richard Büchi. *Weak Second-Order Arithmetic and Finite Automata*, pages 398–424. Springer New York, New York, NY, 1990.

[3] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. A Spatial Logic for Querying Graphs. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Proceedings of the 29$^{th}$ International Colloquium on Automata, Languages and Programming (ICALP'02)*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, July 2002.

[4] Mark Chimes, Radu Iosif, and Florian Zuleger. Tree-verifiable graph grammars. *CoRR*, abs/2402.17015, 2024.

[5] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[6] Bruno Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theor. Comput. Sci.*, 80(2):153–202, 1991.

[7] Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors. *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, volume 5065 of *Lecture Notes in Computer Science.* Springer, 2008.

[8] Amina Doumane. Regular expressions for tree-width 2 graphs. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 121:1–121:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[9] Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Entailment checking in separation logic with inductive definitions is 2-exptime hard. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning,*

*Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, pages 191–211. EasyChair, 2020.

[10] S. Greibach. A note on undecidable properties of formal languages. *Math. Systems Theory*, 2:1–6, 1968.

[11] Radu Iosif, Adam Rogalewicz, and Jirí Simácek. The tree width of separation logic with recursive definitions. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, volume 7898 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2013.

[12] Samin S. Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In Chris Hankin and Dave Schmidt, editors, *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 14–26. ACM, 2001.

[13] Jens Katelaan, Christoph Matheja, Thomas Noll, and Florian Zuleger. Harrsh: A tool for unied reasoning about symbolic-heap separation logic. In Gilles Barthe, Konstantin Korovin, Stephan Schulz, Martin Suda, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22 Workshop and Short Paper Proceedings, Awassa, Ethiopia, 16-21 November 2018*, volume 9 of *Kalpa Publications in Computing*, pages 23–36. EasyChair, 2018.

[14] Jens Katelaan, Christoph Matheja, and Florian Zuleger. Effective entailment checking for separation logic with inductive definitions. In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II*, volume 11428 of *Lecture Notes in Computer Science*, pages 319–336. Springer, 2019.

[15] Viktor Kuncak and Martin Rinard. Generalized records and spatial conjunction in role logic. In *Static Analysis*, pages 361–376, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[16] Christoph Matheja, Jens Pagel, and Florian Zuleger. A decision procedure for guarded separation logic complete entailment checking for separation logic with inductive definitions. *ACM Trans. Comput. Log.*, 24(1):1:1–1:76, 2023.

[17] Francesco Paoli. *Substructural Logics: A Primer*. Springer, Dordrecht, Netherland, 2002.

[18] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings*, pages 55–74. IEEE Computer Society, 2002.

[19] Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '99, page 105–118, New York, NY, USA, 1999. Association for Computing Machinery.

[20] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.