The Caucal Hierarchy: Interpretations in the (W)MSO+U Logic*

Paweł Parys

Institute of Informatics, University of Warsaw, ul. Banacha 2, 02-097 Warszawa, Poland

Abstract

The Caucal hierarchy contains graphs that can be obtained from finite directed graphs by alternately applying the unfolding operation and inverse rational mappings. The goal of this work is to check whether the hierarchy is closed under interpretations in logics extending the monadic second-order logic by the unbounding quantifier, U (saying that a subformula holds for arbitrarily large finite sets). We prove that by applying interpretations described in the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic (hence also in its fragment WMSO+U) to graphs of the Caucal hierarchy we can only obtain graphs on the same level of the hierarchy. Conversely, interpretations described in the more powerful MSO+U logic can give us graphs with an undecidable MSO theory, hence outside of the Caucal hierarchy.

Keywords: Caucal hierarchy, Boundedness, MSO+U

1. Introduction

This paper concerns the class of finitely describable infinite graphs introduced by Caucal [1], called the Caucal hierarchy. Graphs on consecutive levels of this hierarchy are obtained from finite graphs by alternately applying the unfolding operation [2] and inverse rational mappings [3]. Since both these operations preserve decidability of the monadic second-order (MSO) theory, graphs in the Caucal hierarchy have a decidable MSO theory. It turns out that this class of graphs has also other definitions. It was shown [4, 5] that the Caucal hierarchy contains exactly ε-closures of configuration graphs of higher-order pushdown automata [6]; while generating trees, these automata are in turn equivalent to a subclass of higher-order recursion schemes called safe recursion schemes [7]. Moreover, Carayol and Wöhrle [5] prove that the defined classes of graphs do not change if we replace the unfolding operation by another transformation of graphs called a treegraph operation (a.k.a. Muchnik's iteration) [8], and similarly, if we replace inverse rational mappings by the more powerful operation of

*Work supported by the National Science Centre, Poland (grant no. 2016/22/E/ST6/00041).

Email address: parys@mimuw.edu.pl (Paweł Parys)

MSO-interpretations, or even MSO-transductions [9, 10]. One can also replace inverse rational mappings by the operation of FO-interpretations, assuming that the FO formulae have access to the descendant relation [11].

In this paper we try to replace inverse rational mappings or MSO-interpretations in the definition of the Caucal hierarchy by interpretations in some extensions of the MSO logic. Namely, we investigate logics obtained from MSO by adding the unbounding quantifier U introduced by Bojańczyk [12]. The meaning of a formula $\mathsf{U}X.\,\varphi$ is that φ holds for arbitrarily large finite sets X. In the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic we can write $\mathsf{U}X.\,\varphi$ only for formulae φ whose free set variables are restricted to range only on finite sets (this fragment subsumes the more known WMSO+U logic in which all set variables are restricted to range only on finite sets [13, 14]). We prove (in Theorem 4.1) that the Caucal hierarchy does not change if we use MSO+$\mathsf{U}^{\mathsf{fin}}$-interpretations in its definition. In other words, by applying MSO+$\mathsf{U}^{\mathsf{fin}}$-interpretations to graphs in the Caucal hierarchy, we only obtain graphs on the same level of the hierarchy.

This result shows robustness of the Caucal hierarchy. On the other hand, it is a bit disappointing (but rather not surprising): it would be nice to find a class of graphs with decidable properties, larger than the Caucal hierarchy. We remark that the class of trees generated by all (i.e., not necessarily safe) higher-order recursion schemes (equivalently, by collapsible pushdown automata [15]) is such a class: these trees have a decidable MSO theory [16], and some of them are not contained in the Caucal hierarchy [17]. This class lacks a nice machine-independent definition (using logics, like for the Caucal hierarchy), though. For some other classes of graphs we only have decidability of first-order logics [18, 19].

Going further, we examine the full MSO+U logic, where the use of the U quantifier is unrestricted. We prove (in Theorem 5.1) that by applying interpretations written in the MSO+U logic, we can obtain graphs outside of the Caucal hierarchy; among them there are graphs with an undecidable MSO theory. This is very expected, since the model-checking problem for this logic is undecidable, already over the unlabeled infinite word [20].

We remark, however, that our result is not a direct consequence of undecidability of the model-checking problem for MSO+U. It is rather related to another interesting theorem, which we prove (in Theorem 5.2) as a side effect: there exists a fixed formula φ of MSO+U such that it is undecidable, given a regular tree T, whether φ holds in T.

This is a journal version of a conference paper [21]. Comparing to the conference paper, we have added Theorems 4.2 and 5.2, as well as several examples.

The paper is structured as follows. In Section 2 we introduce necessary definitions. In Section 3 we recall basic knowledge on the Caucal hierarchy, and we establish a relation between graphs in this hierarchy and trees generated by recursion schemes. In Section 4 we prove that the Caucal hierarchy is closed under MSO+$\mathsf{U}^{\mathsf{fin}}$-interpretations. Finally, in Section 5 we prove that MSO+U-interpretations lead to graphs with an undecidable MSO theory.

2. Preliminaries

2.1. Logics

A *signature* Ξ (of a relational structure) is a list of relation names, $R_1, \ldots, R_n$, together with a number called an *arity* assigned to each of the names. A *(relational) structure* $\mathcal{S} = (U^{\mathcal{S}}, R_1^{\mathcal{S}}, \ldots, R_n^{\mathcal{S}})$ over such a signature Ξ is a set $U^{\mathcal{S}}$, called the *universe*, together with *relations* $R_i^{\mathcal{S}}$ over $U^{\mathcal{S}}$, for all relation names in the signature; the arity of the relations is as specified in the signature. Following the literature on the Caucal hierarchy [1, 3–5, 22] we forbid the universe to have isolated elements: every element of $U^{\mathcal{S}}$ has to appear in at least one of the relations $R_1^{\mathcal{S}}, \ldots, R_n^{\mathcal{S}}$.

We assume three countable sets of variables: $\mathcal{V}^{FO}$ of first-order variables, $\mathcal{V}^{fin}$ of set variables representing finite sets, and $\mathcal{V}^{inf}$ of set variables representing arbitrary sets. First-order variables are denoted using lowercase letters $x, y, \ldots$, and set variables (of both kinds) are denoted using capital letters $X, Y, \ldots$. The atomic formulae are

- $R(x_1, \ldots, x_n)$, where R is a relation name of arity n (coming from a fixed signature Ξ), and $x_1, \ldots, x_n$ are first-order variables;
- $x = y$, where x, y are first-order variables;
- $x \in X$, where x is a first-order variable, and X a set variable.

Formulae of the *monadic second-order logic with the unbounding quantifier*, MSO+U, are built out of atomic formulae using the Boolean connectives $\vee, \wedge, \neg$, the first-order quantifiers $\exists x$ and $\forall x$, the set quantifiers $\mathsf{U}X$, $\exists_{\mathsf{fin}} X$, and $\forall_{\mathsf{fin}} X$ for $X \in \mathcal{V}^{fin}$, and the set quantifiers $\exists X$ and $\forall X$ for $X \in \mathcal{V}^{inf}$. We use the standard notion of *free variables*. In this paper, we also consider three syntactic fragments of MSO+U. Namely, in the *monadic second-order logic*, MSO, we are not allowed to use variables from $\mathcal{V}^{fin}$, and thus the quantifiers using them: $\mathsf{U}X$ (most importantly), $\exists_{\mathsf{fin}} X$, and $\forall_{\mathsf{fin}} X$. In the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic, the use of the unbounding quantifier is syntactically restricted: we can write $\mathsf{U}X.\, \varphi$ only when all free variables of φ are from $\mathcal{V}^{FO} \cup \mathcal{V}^{fin}$ (i.e., φ has no free variables ranging over infinite sets). In the *weak fragment*, WMSO+U, we cannot use variables from $\mathcal{V}^{inf}$, together with the quantifiers $\exists X$ and $\forall X$.

In order to evaluate an MSO+U formula φ over a signature Ξ in a relational structure $\mathcal{S}$ over the same signature, we also need a *valuation* ν, which is a partial function that maps

- variables $x \in \mathcal{V}^{FO}$ to elements of the universe of $\mathcal{S}$;
- variables $X \in \mathcal{V}^{fin}$ to finite subsets of the universe of $\mathcal{S}$;
- variables $X \in \mathcal{V}^{inf}$ to arbitrary subsets of the universe of $\mathcal{S}$.

The valuation should be defined at least for all free variables of φ. We write $\mathcal{S}, \nu \models \varphi$ when φ is *satisfied* in $\mathcal{S}$ with respect to the valuation ν; this is defined by induction on the structure of φ. For most constructs the definition is as expected, thus we made it explicit only for φ starting with a quantifier $\exists_{\mathsf{fin}}$, $\forall_{\mathsf{fin}}$, or U:

- we have $\mathcal{S}, \nu \models \exists_{\mathsf{fin}} X.\, \psi$ if there exists a finite subset $X^{\mathcal{S}}$ of the universe of $\mathcal{S}$ such that $\mathcal{S}, \nu[X \mapsto X^{\mathcal{S}}] \models \psi$,

- we have $\mathcal{S}, \nu \models \forall_{\mathsf{fin}} X.\, \psi$ if for every finite subset $X^{\mathcal{S}}$ of the universe of $\mathcal{S}$ it holds $\mathcal{S}, \nu[X \mapsto X^{\mathcal{S}}] \models \psi$, and
- we have $\mathcal{S}, \nu \models \mathsf{U} X.\, \psi$ if for every $n \in \mathbb{N}$ there exists a finite subset $X^{\mathcal{S}}$ of the universe of $\mathcal{S}$ having cardinality at least n such that $\mathcal{S}, \nu[X \mapsto X^{\mathcal{S}}] \models \psi$.

In other words, $\exists_{\mathsf{fin}} X.\, \psi$, $\forall_{\mathsf{fin}} X.\, \psi$, and $\mathsf{U} X.\, \psi$ say that ψ is satisfied for some finite set X, for all finite sets X, and for some arbitrarily large finite sets X, respectively.

We write $\varphi(x_1, \ldots, x_n)$ to denote that the free variables of φ are among $x_1, \ldots, x_n$. Then, given elements $u_1, \ldots, u_n$ in the universe of a structure $\mathcal{S}$, we say that $\varphi(u_1, \ldots, u_n)$ is satisfied in $\mathcal{S}$ if φ is satisfied in $\mathcal{S}$ under the valuation mapping x_i to u_i for all $i \in \{1, \ldots, n\}$.

For a logic $\mathcal{L}$, an *$\mathcal{L}$-interpretation* from Ξ_1 to Ξ_2 is a family I of $\mathcal{L}$-formulae $\varphi_R(x_1, \ldots, x_n)$ over Ξ_1, for every relation name R of Ξ_2, where n is the arity of R. Having such an $\mathcal{L}$-interpretation, we can apply it to a structure $\mathcal{S}$ over Ξ_1; we obtain a structure $I(\mathcal{S})$ over Ξ_2, where every relation $R^{I(\mathcal{S})}$ is given by the tuples $(v_1, \ldots, v_n)$ of elements of the universe of $\mathcal{S}$ for which $\varphi_R(v_1, \ldots, v_n)$ is satisfied in $\mathcal{S}$. The universe of $I(\mathcal{S})$ is given implicitly as the set of all elements occurring in the relations $R^{I(\mathcal{S})}$ (because isolated elements are disallowed by the definition of a structure, there is no need to have a separate formula defining the universe).

While writing formulae in this paper, we use the usual notational conventions: we write $\varphi \Rightarrow \psi$ for $\neg\varphi \vee \psi$, we write $\varphi \Leftrightarrow \psi$ for $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, we write $\forall x \in X.\, \varphi$ for $\forall x.\, (x \in X \Rightarrow \varphi)$, we write $\exists x \in X.\, \varphi$ for $\exists x.\, (x \in X \wedge \varphi)$, and we write $x \neq y$ for $\neg(x = y)$.

2.2. Graphs and the Caucal hierarchy

The Caucal hierarchy consists of directed, edge-labeled graphs (later on, we also consider node-labeled structures). Thus, for a finite set Σ, a *Σ-labeled graph* G is a relational structure over the signature Ξ_Σ containing binary relation names E_a for all $a \in \Sigma$. In other words, $G = (V^G, (E_a^G)_{a \in \Sigma})$, where V^G is a set of nodes, and $E_a^G \subseteq V^G \times V^G$ is a set of a-labeled edges, for every $a \in \Sigma$ (and where we assume that there are no isolated nodes, i.e., for every $v \in V^G$ there is an edge (v, w) or (w, v) in E_a^G for some $w \in V^G$ and $a \in \Sigma$). A graph is *deterministic* if for every $v \in V^G$ and $a \in \Sigma$ there is at most one node $w \in V^G$ such that $(v, w) \in E_a^G$.

A *path* from a node u to a node v labeled by $w = a_1 \ldots a_n$ is a sequence $v_0 a_1 v_1 \ldots a_n v_n \in V^G(\Sigma V^G)^*$, where $v_0 = u$, and $v_n = v$, and $(v_{i-1}, v_i) \in E_{a_i}^G$ for all $i \in \{1, \ldots, n\}$. A graph is called an (edge-labeled) *tree* when it contains a node r, called the *root*, such that for every node $v \in V^G$ there exists a unique path from r to v. The *unfolding* $Unf(G, r)$ of a graph $G = (V^G, (E_a^G)_{a \in \Sigma})$ from a node $r \in V^G$ is the tree $T = (V^T, (E_a^T)_{a \in \Sigma})$, where V^T is the set of all paths in G starting from r, and E_a^T (for every $a \in \Sigma$) contains pairs (w, w') such that $w' = w \cdot a \cdot v$ for some $v \in V^G$.

The Caucal hierarchy is a sequence of classes of graphs and trees; we use here the characterization from Carayol and Wöhrle [5] as a definition. We define

$\mathit{Graph}(0)$ to be the class containing all finite Σ-labeled graphs, for all finite sets of labels Σ. For all $n \geq 0$, we let

$$\mathit{Tree}(n+1) = \{\, \mathit{Unf}(G, r) \mid G \in \mathit{Graph}(n), r \in V^G \,\}\,, \qquad \text{and}$$
$$\mathit{Graph}(n+1) = \{\, I(T) \mid T \in \mathit{Tree}(n+1), I \text{ an MSO-interpretation} \,\}\,.$$

We do not distinguish between isomorphic graphs.

Example 2.1. Examples of graphs are presented on Figures 1-5. Graphs T_1 and T_2 are trees. We see that T_1 is the unfolding of G_0 from the node depicted on the top. Likewise, T_2 is the unfolding of G_1 from the node depicted on top left.

Let us now observe that G_1 can be MSO-interpreted in T_1. The a-labeled edges remain unchanged, thus we take $\varphi_{E_a}(x, y) \equiv E_a(x, y)$. In order to define b-labeled edges, we need a formula saying that y is reachable from x via a-labeled edges (i.e., that every set containing x and closed under following a-labeled edges contains y):

$$\mathit{conn}_a(x, y) \equiv \forall X.\ (x \in X \wedge \forall z \in X. \forall z'.\ (E_a(z, z') \Rightarrow z' \in X) \Rightarrow y \in X)\ .$$

Using this formula, we say when x and y should be connected by a b-labeled edge in G_1:

$$\varphi_{E_b}(x, y) \equiv x \neq y \wedge \exists x'. \exists y'.\ ((x' = x \vee E_b(x', x)) \wedge E_b(y', y) \wedge \mathit{conn}_a(y', x'))\ .$$

This formula says that there is an edge from y' to y (thus y' is a node depicted in the top line on Figure 2, and y is in the bottom line), that x' is reachable from y' via a-labeled edges, and that x either equals x' or is the node below x; additionally $x \neq y$. This results in creating the edges as in G_1.

Next, we observe that G_2 can be MSO-interpreted in T_2. To this end we define $\mathit{conn}_b(x, y)$ like $\mathit{conn}_a(x, y)$ (with a replaced by b). We then take

$$\varphi_{E_a}(x, y) \equiv \exists z.\ (\mathit{conn}_b(z, x) \wedge z \neq x \wedge \mathit{conn}_b(z, y) \wedge z \neq y)\ , \qquad \text{and}$$
$$\varphi_{E_b}(x, y) \equiv \exists z. \exists z'.\ (\mathit{conn}_b(z, y) \wedge z \neq y \wedge E_a(z, z') \wedge \mathit{conn}_b(z', x) \wedge z' \neq x)\ .$$

Notice that the nodes of T_2 depicted in the top row are not taken into G_2.

Because G_0 is finite, we have $G_i \in \mathit{Graph}(i)$ and $T_j \in \mathit{Tree}(j)$ for $i \in \{0, 1, 2\}$ and $j \in \{1, 2\}$. We remark that G_0 and T_1 are deterministic, while G_1, T_2, G_2 are not (because of multiple b-labeled edges leaving a single node). □

2.3. Higher-Order Recursion Schemes

The set of *(simple) types* is constructed from a unique ground type o using a binary operation $\to$; namely o is a type, and if α and β are types, so is $\alpha \to \beta$. By convention, $\to$ associates to the right, that is, $\alpha \to \beta \to \gamma$ is understood as $\alpha \to (\beta \to \gamma)$. The *order* of a type α, denoted $\mathit{ord}(\alpha)$ is defined by induction: $\mathit{ord}(\mathsf{o}) = 0$ and $\mathit{ord}(\alpha_1 \to \cdots \to \alpha_k \to \mathsf{o}) = \max_i(\mathit{ord}(\alpha_i)) + 1$ for $k \geq 1$.

Having a finite set of letters Σ (an alphabet), a finite set of typed nonterminals $\mathcal{N}$, and a finite set of typed variables V, *(applicative) terms* over $(\Sigma, \mathcal{N}, V)$ are defined by induction:

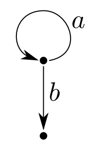


Figure 1: A graph G_0

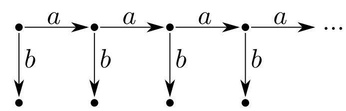


Figure 2: A tree T_1

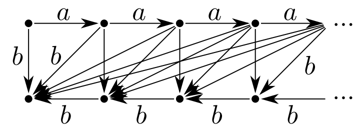


Figure 3: A graph G_1 (arrows without labels denote b-labeled edges)

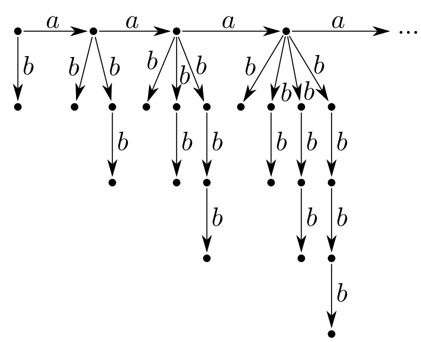


Figure 4: A tree T_2

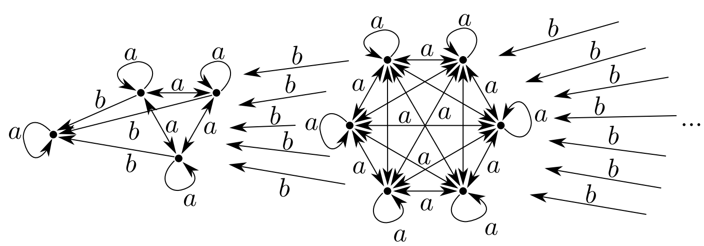


Figure 5: A graph G_2 (the k-th clique consists of $k(k+1)/2$ nodes, connected by a-labeled edges; every node of a clique is connected to every node of the previous clique by a b-labeled edge)

- every nonterminal $X \in \mathcal{N}$ of type α is a term of type α;
- every variable $x \in V$ of type α is a term of type α;
- if $K_1, \ldots, K_k$ are terms of type o and $a \in \Sigma$ is a letter, then $a\langle K_1, \ldots, K_k\rangle$ is a term of type o;
- if K is a term of type $\alpha \to \beta$, and L is a term of type α, then $K\,L$ is a term of type β.

The order of a term K, written $ord(K)$, is defined as the order of its type.

A *(higher-order) recursion scheme* is a tuple $\mathcal{G} = (\Sigma, \mathcal{N}, S, \mathcal{R})$, where Σ is a finite set of letters, $\mathcal{N}$ a finite set of typed nonterminals, $S \in \mathcal{N}$ is a starting nonterminal of type o, and $\mathcal{R}$ a function assigning to every nonterminal $X \in \mathcal{N}$ of type $\alpha_1 \to \cdots \to \alpha_k \to \mathsf{o}$ a rule of the form $X\,x_1\,\ldots\,x_k \to K$, where types of variables $x_1, \ldots, x_k$ are $\alpha_1, \ldots, \alpha_k$, respectively, and K is a term of type o over $(\Sigma, \mathcal{N}, \{x_1, \ldots, x_k\})$. The order of a recursion scheme is defined as the maximum of orders of its nonterminals.

Unlike trees in the Caucal hierarchy, trees generated by recursion schemes are node-labeled; actually, these are infinite terms. They are defined by coinduction (meaning that they can be infinite; for an introduction to coinductive definitions and proofs see, e.g., Czajka [23]): for a finite set Σ and for $k_{\max} \in \mathbb{N}$, a Σ-*node-labeled tree of maximal arity* $k_{\max}$ is of the form $a\langle T_1, \ldots, T_k\rangle$, where $a \in \Sigma$, and $k \leq k_{\max}$, and $T_1, \ldots, T_k$ are again Σ-node-labeled trees of maximal arity $k_{\max}$. For a tree $T = a\langle T_1, \ldots, T_k\rangle$, its set of nodes is defined as the smallest set such that

- ε is a node of T, labeled by a, and
- if u is a node of T_i for some $i \in \{1, \ldots, k\}$, labeled by b, then iu is a node of T, also labeled by b.

Such a tree can be seen as a relational structure over signature $\Xi^{nlt}_{\Sigma,k_{\max}}$ containing unary relation names L_a for all $a \in \Sigma$, and binary relation names Ch_i for all $i \in \{1, \ldots, k_{\max}\}$. Its universe is the set of nodes of T; for $a \in \Sigma$ the relation L^T_a contains all nodes labeled by a; for $i \in \{1, \ldots, k_{\max}\}$ the i-th child relation Ch^T_i contains pairs (u, ui) such that both u and ui are nodes of T.

Having a recursion scheme $\mathcal{G}$, we define a rewriting relation $\to_{\mathcal{G}}$ among terms of type o over $(\Sigma, \mathcal{N}, \emptyset)$: we have $X\,L_1\,\ldots\,L_k \to_{\mathcal{G}} K[L_1/x_1, \ldots, L_k/x_k]$, where X is a nonterminal such that the rule $\mathcal{R}(X)$ is $X\,x_1\,\ldots\,x_k \to K$ (and where $K[L_1/x_1, \ldots, L_k/x_k]$ is the term obtained from K by substituting L_1 for x_1, L_2 for x_2, and so on). We then define a tree *generated* by $\mathcal{G}$ from a term K of type o over $(\Sigma, \mathcal{N}, \emptyset)$, by coinduction:

- if there is a reduction sequence from K to a term of the form $a\langle L_1, \ldots, L_k\rangle$, then the tree equals $a\langle T_1, \ldots, T_k\rangle$, where $T_1, \ldots, T_k$ are the trees generated by $\mathcal{G}$ from $L_1, \ldots, L_k$, respectively;
- otherwise, the tree equals $\omega\langle\rangle$ (where ω is a distinguished letter).

This tree is over an extended alphabet $\Sigma \uplus \{\omega\}$. A tree generated by $\mathcal{G}$ (without a term specified) is the tree generated by $\mathcal{G}$ from the starting nonterminal S.

Notice that for every term K of type o over $(\Sigma, \mathcal{N}, \emptyset)$ there is at most one term K' such that $K \to_{\mathcal{G}} K'$; thus the tree generated by $\mathcal{G}$ is unique. Indeed, every such term is either of the form $X\,L_1\,\ldots\,L_k$ (for a nonterminal X) or $a\langle L_1, \ldots, L_k\rangle$. In the former case, the term has a successor in the $\to_{\mathcal{G}}$ relation;

in the latter case, we create a node of the generated tree.

Example 2.2. Consider a recursion scheme $\mathcal{G}_1 = (\{d, e, f\}, \mathcal{N}_1, S, \mathcal{R}_1)$, where $\mathcal{N}_1$ contains nonterminals S and X of type o, and Y of type $\mathsf{o} \to \mathsf{o} \to \mathsf{o}$. The rules in $\mathcal{R}_1$ are

$$S \to Y\, e\langle X, f\langle X\rangle\rangle\, f\langle f\langle X\rangle\rangle\,, \qquad X \to X\,,$$
$$Y\, x\, y \to d\langle x, Y\, e\langle x, y\rangle\, f\langle y\rangle\rangle\,.$$

The bracketing convention here is that $Y\, e\langle X, f\langle X\rangle\rangle$ denotes $Y\,(e\langle X, f\langle X\rangle\rangle)$, etc. The type of both variables x, y is o, respectively. The order of the nonterminal Y is 1, and thus also the order of $\mathcal{G}_1$ is 1.

Because $X \to_{\mathcal{G}_1} X$, the tree generated by $\mathcal{G}_1$ from X is $\omega\langle\rangle$. Denote

$$M_0 = N_0 = X\,, \qquad M_i = f\langle M_{i-1}\rangle\,, \qquad N_i = e\langle N_{i-1}, M_i\rangle \qquad \text{for } i \geq 1\,,$$

and likewise

$$T_0 = U_0 = \omega\langle\rangle\,, \qquad T_i = f\langle T_{i-1}\rangle\,, \qquad U_i = e\langle U_{i-1}, T_i\rangle \qquad \text{for } i \geq 1\,.$$

The tree generated by $\mathcal{G}_1$ from M_i (N_i) is T_i (U_i, respectively). We have

$$S \to_{\mathcal{G}_1} Y\, N_1\, M_2 \to_{\mathcal{G}_1} d\langle N_1, Y\, e\langle N_1, M_2\rangle\, f\langle M_2\rangle\rangle = d\langle N_1, Y\, N_2\, M_3\rangle\,.$$

Thus the tree generated by $\mathcal{G}_1$ (from the starting nonterminal S) has d-labeled root, below which we have the tree generated from N_1, that is U_1, as the first subtree, and the (infinite) tree generated from $Y\, N_2\, M_3$ as the second subtree. Going further, we have

$$Y\, N_i\, M_{i+1} \to_{\mathcal{G}_1} d\langle N_i, Y\, e\langle N_i, M_{i+1}\rangle\, f\langle M_{i+1}\rangle\rangle = d\langle N_i, Y\, N_{i+1}\, M_{i+2}\rangle\,,$$

and thus the tree generated by $\mathcal{G}_1$ from $Y\, N_i\, M_{i+1}$ has d-labeled root, U_i as the first subtree, and the tree generated from $Y\, N_{i+1}\, M_{i+2}$ as the second subtree (for every $i \in \mathbb{N}$). Overall, the tree T_{n1} generated by $\mathcal{G}_1$ is depicted on Figure 6. □

Example 2.3. Let us now define a more complicated recursion scheme, namely $\mathcal{G}_3 = (\{a, b, c\}, \mathcal{N}_3, S, \mathcal{R}_3)$ with the following rules in $\mathcal{R}_3$:

$$S \to X\, D\, B\,, \qquad X\, x\, y \to a\langle y\, c\langle\rangle, X\,(Y\, x)\,(x\, y)\rangle\,, \qquad B\, z \to b\langle z\rangle\,,$$
$$Y\, x\, y\, z \to y\,(x\, y\, z)\,, \qquad D\, y\, z \to y\,(y\, z)\,.$$

Types of the variables are, respectively,

$$x\colon (\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}\,, \qquad y\colon \mathsf{o} \to \mathsf{o}\,, \qquad z\colon \mathsf{o}\,.$$

In consequence, types of the nonterminals are, respectively,

$$S\colon \mathsf{o}\,, \qquad X\colon ((\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}) \to (\mathsf{o} \to \mathsf{o}) \to \mathsf{o}\,, \qquad B\colon \mathsf{o} \to \mathsf{o}\,,$$
$$Y\colon ((\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}) \to (\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}\,, \qquad D\colon (\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}\,.$$

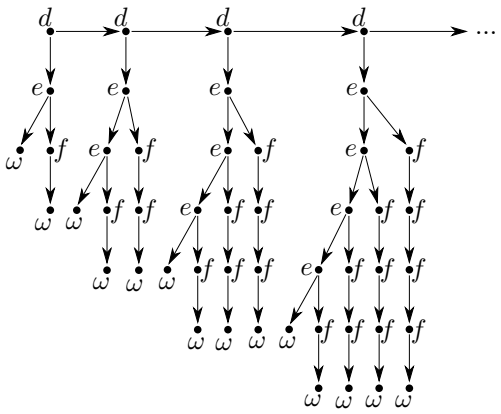


Figure 6: A tree $T_{\mathsf{n}1}$, which is generated by the recursion scheme $\mathcal{G}_1$ from Example 2.2, and by the pushdown automaton $\mathcal{A}_{\mathsf{n}1}$ from Example 2.11. Children are ordered from left to right.

The order of $\mathcal{G}_3$ is 3.

Denote

$$D_2 = D \qquad \text{and} \qquad D_i = Y\, D_{i-1} \text{ for } i \geq 3\,, \qquad \text{and}$$
$$B_1 = B \qquad \text{and} \qquad B_i = D_i\, B_{i-1} \text{ for } i \geq 2\,.$$

Notice that D_i represents a function of type $(\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}$ that applies i times the function given as the first argument to the value given as the second argument. In consequence, B_i represents a function of type $\mathsf{o} \to \mathsf{o}$ that appends $i!$ times a b-labeled node above the value given as its argument. We have that

$$S \to_{\mathcal{G}_3} X\, D_2\, B_1 \qquad \text{and}$$
$$X\, D_{i+1}\, B_i \to_{\mathcal{G}_3} a\langle B_i\, c\langle\rangle, X\, D_{i+2}\, B_{i+1}\rangle \qquad \text{for } i \geq 2\,.$$

It follows that the tree $T_{\mathsf{n}3}$ generated by $\mathcal{G}_3$ is a comb, where on the rightmost branch we have an infinite sequence of a-labeled nodes, and to the left of the k-th a-labeled node there is attached a branch consisting of $k!$ b-labeled nodes, and finished with a c-labeled node. See Figure 7. □

We define when a term is *safe*, by induction on its structure:

- all nonterminals and variables are safe,
- a term $a\langle K_1, \ldots, K_k\rangle$ is safe if the subterms $K_1, \ldots, K_k$ are safe,
- a term $M = K\, L_1\, \ldots\, L_k$ is safe if the subterms $K, L_1, \ldots, L_k$ are safe and $ord(x) \geq ord(M)$ for all variables x appearing in M.

A recursion scheme is safe if right-hand sides of all its rules are safe. In this paper we only consider safe recursion schemes.

Example 2.4. In order to see that the recursion scheme $\mathcal{G}_1$ from Example 2.2 is safe, we have to check inequalities

$$ord(x) \geq ord(Y\, e\langle x, y\rangle\, f\langle y\rangle) \qquad \text{and} \qquad ord(y) \geq ord(Y\, e\langle x, y\rangle\, f\langle y\rangle)\,.$$

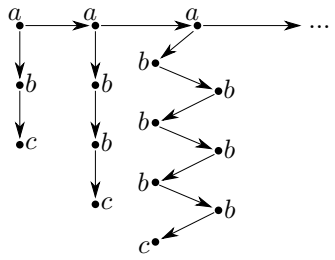


Figure 7: A tree $T_{\mathrm{n}3}$, which is generated by the recursion scheme $\mathcal{G}_3$ from Example 2.3. The number b-labeled nodes on the branch leaving the k-th a-labeled node is $k!$.

They trivially hold: all orders equal 0.

Actually, one can see that every recursion scheme of order 1 (or 0) is safe. Indeed, if M is a subterm of the right-hand side of a rule in such a recursion scheme, and if M is of order 0 (recall that in particular the whole right-hand side is always of order 0), then either

- M is a nonterminal or a variable (and thus M is safe), or
- M is of the form $a\langle K_1, \ldots, K_k\rangle$, where $K_1, \ldots, K_k$ are of order 0, or
- M is of the form $X\, L_1\, \ldots\, L_k$, where X is a nonterminal (thus X is safe), and $L_1, \ldots, L_k$ are of order 0; the inequality $ord(x) \geq ord(M)$ trivially holds in this case, for every variable x, because $ord(M) = 0$.

In consequence, an immediate structural induction shows that right-hand sides of all the rules are safe, that is, that the recursion scheme is safe.

Notice that not all subterms of a safe term need to be safe. For example, in the rule for Y in $\mathcal{G}_1$ there is a subterm $Y\, e\langle x, y\rangle$ of order 1 that has a free variable x of order 0; this subterm is not safe. □

Example 2.5. Let us now see that the recursion scheme $\mathcal{G}_3$ from Example 2.3 is safe. We have to check that

$$2 = ord(x) \geq ord(Y\, x) = 2\,,$$
$$2 = ord(x) \geq ord(x\, y) = 1\,, \qquad 1 = ord(y) \geq ord(x\, y) = 1\,;$$

besides that, there are some trivially satisfied inequalities, in which the right-hand side (i.e., the order of a subterm) is 0. □

Remark 2.6. Clearly there exist recursion schemes that are not safe. Some of them can be rewritten as safe recursion schemes generating the same tree. There exist, however, recursion schemes for which there is no safe recursion scheme generating the same tree [17].

2.4. Higher-Order Pushdown Automata

We actually need to consider two models of higher-order pushdown automata: nondeterministic (non-branching) automata of Carayol and Wöhrle [5],

where letters are read by transitions, and deterministic tree-generating automata of Knapik, Niwiński, and Urzyczyn [7], where there are special commands for creating labeled tree nodes. We use the name *edge-labeled pushdown automata* for the former model, and *node-labeled pushdown automata* for the latter model.

Most parts of the definition are common for these two models. Let Γ be a finite set containing a distinguished initial symbol $\bot \in \Gamma$. For every $n \in \mathbb{N}$ we define a set $PD_n(\Gamma)$ of *stacks of order* n over the stack alphabet Γ: we let $PD_0(\Gamma) = \Gamma$, and for $n \geq 1$ as $PD_n(\Gamma)$ we take the set of nonempty sequences of elements of $PD_{n-1}(\Gamma)$. For such sequences we use the notation $[s_1, s_2, \ldots, s_k]$. The initial stack $\bot_n \in PD_n(\Gamma)$ is also defined by induction: $\bot_0 = \bot$, and $\bot_n = [\bot_{n-1}]$ for $n \geq 1$. The function $top\colon PD_n(\Gamma) \to \Gamma$ returns the topmost symbol of a stack: we have $top(\gamma) = \gamma$ if $\gamma \in PD_0(\Gamma)$, and $top([s_1, s_2, \ldots, s_k]) = top(s_k)$.

Next, for every $n \in \mathbb{N}$ we define a finite set $Op_n(\Gamma)$ of operations on these stacks, where every $op \in Op_n(\Gamma)$ is a partial function from $PD_n(\Gamma)$ to $PD_n(\Gamma)$. Namely, we take $Op_n(\Gamma) = \{id\} \cup \{rew_\rho \mid \rho \in \Gamma\} \cup \{push^i, pop^i \mid 1 \leq i \leq n\}$, where for $\gamma \in PD_0(\Gamma)$ we define $id(\gamma) = \gamma$ and $rew_\rho(\gamma) = \rho$, and for $s = [s_1, \ldots, s_{k-1}, s_k] \in PD_n(\Gamma)$ we define

- $push^n(s) = [s_1, \ldots, s_{k-1}, s_k, s_k]$;
- $pop^n(s) = [s_1, \ldots, s_{k-1}]$ if $k \geq 2$;
- $op(s) = [s_1, \ldots, s_{k-1}, op(s_k)]$ if $op \in Op_n(\Gamma) \setminus \{push^n, pop^n\}$ and $op(s_k)$ is defined.

Notice that $pop^i(s)$ remains undefined if the topmost order-i stack of s has size 1. We overload here the operation names: the same name (e.g. pop^4) is used for operations over stacks of different orders.

Having the above, we define an *edge-labeled pushdown automaton of order* n as a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_I, \Delta)$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite stack alphabet, $q_I \in Q$ is an initial state, and $\Delta \subseteq Q \times \Gamma \times (\Sigma \uplus \{\varepsilon\}) \times Q \times Op_n(\Gamma)$ is a transition relation. It is assumed that for every pair (q, γ) either all tuples $(q, \gamma, a, q', op) \in \Delta$ have $a = \varepsilon$, or all have $a \in \Sigma$. The automaton is *deterministic* if for every pair (q, γ) there is either exactly one transition (q, γ, a, q', op), where $a = \varepsilon$, or there are $|\Sigma|$ such transitions, one for every $a \in \Sigma$. A *configuration* of $\mathcal{A}$ is a pair $(q, s) \in Q \times PD_n(\Gamma)$, and $(q_I, \bot_n)$ is the initial configuration. For $a \in \Sigma \cup \{\varepsilon\}$, there is an a-labeled *transition* from a configuration (p, s) to a configuration (q, t), written $(p, s) \xrightarrow{a}_{\mathcal{A}} (q, t)$, if in Δ there is a tuple $(p, top(s), a, q, op)$ such that $op(s) = t$. The *configuration graph* of $\mathcal{A}$ is the edge-labeled graph of all configurations of $\mathcal{A}$ reachable from the initial configuration, with an edge labeled by $a \in \Sigma \cup \{\varepsilon\}$ from c to d if there is a transition $c \xrightarrow{a}_{\mathcal{A}} d$.

Let G be the configuration graph of $\mathcal{A}$ (or, in general, any $(\Sigma \uplus \{\varepsilon\})$-labeled graph, in which edges leaving every particular node are either all labeled by elements of Σ, or all labeled by ε). The ε-closure of G is the Σ-labeled graph obtained from G by removing all nodes with only outgoing ε-labeled edges and adding an a-labeled edge between v and w if in G there is a path from v to w labeled by a word in $a\varepsilon^*$. Notice that, in particular,

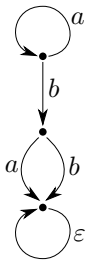


Figure 8: The configuration graph of the automaton $\mathcal{A}_0$ from Example 2.8

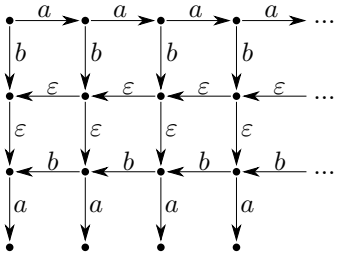


Figure 9: The configuration graph of the automaton $\mathcal{A}_1$ from Example 2.9

- nodes without any outgoing edges in G are removed in the ε-closure;
- in the ε-closure there may exist nodes without any outgoing edges (resulting from nodes whose all potential successors were removed);
- the initial configuration of $\mathcal{A}$ is treated as any other node, hence it may be removed.

The graph *generated* by $\mathcal{A}$ is the ε-closure of the configuration graph of $\mathcal{A}$.

Remark 2.7. One can imagine several nonequivalent definitions of the ε-closure of a graph, following the same idea, but differing in some details. We use the particular definition given above in order order to remain consistent with the literature (in particular with Carayol and Wöhrle [5], and with Carayol's PhD thesis [22]).

Example 2.8. Let us start with a degenerate case, where the order is 0. Namely, consider a (deterministic) edge-labeled pushdown automaton $\mathcal{A}_0 = (\{q_1, q_2, q_3\}, \{a, b\}, \{\bot\}, q_1, \Delta_0)$ of order 0. It has three states (among which q_1 is initial), two input letters, one stack symbol. The transition relation is defined as the following set:

$$\Delta_0 = \{(q_1, \bot, a, q_1, id), (q_1, \bot, b, q_2, id), (q_2, \bot, a, q_3, id), \\ (q_2, \bot, b, q_3, id), (q_3, \bot, \varepsilon, q_3, id)\}.$$

Notice that the stack of $\mathcal{A}_0$ is $\bot$ in every configuration, and thus it can be ignored. Thus, $\mathcal{A}_0$ is just a nondeterministic finite automaton. Actually, this is true for every automaton of order 0, even if the symbol on the stack can change. This is because the stack, taking only one among finitely many values, can be treated as a part of a state.

The configuration graph of $\mathcal{A}_0$ is depicted on Figure 8. Looking from the top, the nodes correspond to configurations $(q_1, \bot)$, $(q_2, \bot)$, and $(q_3, \bot)$.

In order to obtain the ε-closure of this graph, we remove the node for the configuration $(q_3, \bot)$ (because all edges leaving this node are ε-labeled). Thus, this ε-closure, that is, the graph generated by $\mathcal{A}_0$ is exactly the graph G_0 from Figure 1. □

Example 2.9. As an example of an edge-labeled pushdown automaton of order 1 consider $\mathcal{A}_1 = (\{q_1, q_2, q_3, q_4\}, \{a, b\}, \{\bot\}, q_1, \Delta_1)$. Its transition relation is defined as the following set:

$$\Delta_1 = \{(q_1, \bot, a, q_1, push^1), (q_1, \bot, b, q_2, id), (q_2, \bot, \varepsilon, q_2, pop^1), \\ (q_2, \bot, \varepsilon, q_3, id), (q_3, \bot, b, q_3, pop^1), (q_3, \bot, a, q_4, id)\}.$$

Starting from the initial configuration $(q_1, [\bot])$, the automaton $\mathcal{A}_1$ can push some number of $\bot$ symbols to the stack, reading the letter a in every such step. Then, at some moment, it decides to change the state to q_2 and read the letter b. In state q_2 some number of stack symbols can be popped, without reading any letter (ε-transitions), and then the state changes to q_3. In state q_3 again some number of stack symbols can be popped, but this time the automaton reads b in every step. Finally, it can change its state to q_4 while reading a. In state q_4 the automaton gets stuck. The configuration graph of $\mathcal{A}_1$ is depicted on Figure 9. A node in the i-th row and j-th column denotes a configuration with state q_i and with j symbols on the stack.

We can observe that the ε-closure of the aforementioned graph (i.e, the graph generated by $\mathcal{A}_1$) is exactly the graph G_1 from Figure 3. Indeed, in the ε-closure we should remove all configurations with state q_2 and q_4. Moreover, every path going down from a node in the top row, then left some number of times, and then down again should be contracted to a single b-labeled edge. Notice that a-labeled edges between the third and the fourth row disappear in the ε-closure.

We remark that $\mathcal{A}_1$ is not deterministic, because of the two ε-transitions $(q_2, \bot, \varepsilon, q_2, pop^1)$, $(q_2, \bot, \varepsilon, q_3, id)$, both having $(q_2, \bot)$ on the first two coordinates. □

Example 2.10. Let us now present an automaton $\mathcal{A}_2$ of order 2 that generates the graph G_2 from Figure 5. We take $\mathcal{A}_2 = (\{q_i \mid i \in \{1, \ldots, 8\}\}, \{a, b\}, \{\bot\}, q_1, \Delta_2)$. The transition relation is

$$\Delta_2 = \{(q_1, \bot, \varepsilon, q_1, push^1), (q_1, \bot, \varepsilon, q_2, push^1), (q_2, \bot, \varepsilon, q_3, push^2), \\ (q_3, \bot, \varepsilon, q_3, pop^1), (q_3, \bot, \varepsilon, q_4, push^2), (q_4, \bot, \varepsilon, q_4, pop^1), \\ (q_4, \bot, \varepsilon, q_5, pop^1), (q_5, \bot, a, q_6, pop^2), (q_6, \bot, \varepsilon, q_2, pop^2), \\ (q_5, \bot, b, q_7, pop^2), (q_7, \bot, \varepsilon, q_8, pop^2), (q_8, \bot, \varepsilon, q_2, pop^1)\}.$$

This relation is depicted symbolically on Figure 10.

The configurations of $\mathcal{A}_2$ that remain in the generated graph are of the form $(q_5, [[\bot^{k+1}], [\bot^{j+1}], [\bot^i]])$, where $1 \leq i \leq j \leq k$, and where $[\bot^i]$ denotes the stack of order 1 consisting of i repetitions of the symbol $\bot$. Configurations with the same value of k are grouped in the same clique of G_2; the number of pairs (i, j) such that $1 \leq i \leq j \leq k$ is indeed $k(k+1)/2$.

It is easy to see that configurations with state q_5 reachable from the initial configuration are exactly all configurations of the aforementioned form. Moreover, after reading a the automaton can return to such a configuration with the value of k unchanged, and with arbitrary values of i and j (in the appropriate

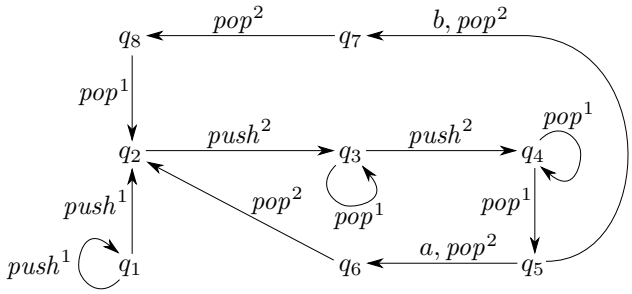


Figure 10: Transitions of the automaton $\mathcal{A}_2$ from Example 2.10

range). Likewise, after reading b the automaton can return to such a configuration with the value of k decreased by 1, and with arbitrary i and j. □

Next, we define a *node-labeled pushdown automaton of order n* as a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_I, \delta)$, where Q, Σ, Γ, q_I (and configurations) are as previously, and $\delta\colon Q \times \Gamma \to (Q \times Op_n(\Gamma)) \uplus (\Sigma \times Q^*)$ is a transition function. This time transitions are not labeled by anything; we have $(p, s) \to_{\mathcal{A}} (q, t)$ when $\delta(p, top(s)) = (q, op)$ and $op(s) = t$. We write $\to_{\mathcal{A}}^*$ for the reflexive transitive closure of $\to_{\mathcal{A}}$. We define when a node-labeled tree over the alphabet $\Sigma \uplus \{\omega\}$ is generated by $\mathcal{A}$ from (p, s), by coinduction:

- if $(p, s) \to_{\mathcal{A}}^* (q, t)$, and $\delta(q, top(t)) = (a, q_1, \ldots, q_k) \in \Sigma \times Q^*$, and trees $T_1, \ldots, T_k$ are generated by $\mathcal{A}$ from $(q_1, t), \ldots, (q_k, t)$, respectively, then the tree $a\langle T_1, \ldots, T_k\rangle$ is generated by $\mathcal{A}$ from (p, s);
- if there is no (q, t) such that $(p, s) \to_{\mathcal{A}}^* (q, t)$ and $\delta(q, top(t)) \in \Sigma \times Q^*$, then $\omega\langle\rangle$ is generated by $\mathcal{A}$ from (p, s).

While talking about the tree generated by $\mathcal{A}$, without referring to a configuration, we mean generating from the initial configuration $(q_I, \bot_n)$.

Example 2.11. Let us generate the tree T_{n1} depicted on Figure 6 using a node-labeled pushdown automaton of order 1. To this end, we consider an automaton $\mathcal{A}_{\mathsf{n1}} = (\{q_1, q_2, q_3, q_1', q_2', q_3'\}, \{d, e, f\}, \{\bot\}, q_1, \delta_1)$ whose transition function is defined by

$$\delta(q_1, \bot) = (d, q_2, q_1')\,, \qquad \delta(q_2, \bot) = (e, q_2', q_3)\,, \qquad \delta(q_3, \bot) = (f, q_3')\,,$$
$$\delta(q_1', \bot) = (q_1, push^1)\,, \qquad \delta(q_2', \bot) = (q_2, pop^1)\,, \qquad \delta(q_3', \bot) = (q_3, pop^1)\,.$$

Observe that the automaton indeed generates T_{n1}. □

3. Between Caucal Hierarchy and Safe Recursion Schemes

This section is devoted to a proof of the following theorem, making a bridge between the Caucal hierarchy and safe recursion schemes.

Theorem 3.1. *For every $n \geq 1$, a graph G is in Graph(n) if and only if it can be obtained by applying an* MSO*-interpretation to a tree generated by a safe recursion scheme of order $n - 1$.*

Although it is known that the Caucal hierarchy is closely related to safe recursion schemes, we are not aware of any paper in which a theorem like our Theorem 3.1 is shown. Nevertheless, we have the following two results, from Carayol and Wöhrle [5, Theorem 3] and Knapik et al. [7, Theorems 5.1 and 5.3].

Theorem 3.2. *For every $n \in \mathbb{N}$, a graph G belongs to $Graph(n)$ if and only if it is generated by some edge-labeled pushdown automaton of order n.* □

Theorem 3.3. *For every $n \in \mathbb{N}$, a tree T is generated by some node-labeled pushdown automaton of order n if and only if it is generated by some safe recursion scheme of order n.* □

Example 3.4. We have seen in Examples 2.1 and 2.8–2.10, for $i \in \{0, 1, 2\}$, a graph G_i that belongs to $Graph(i)$ and that is generated by an edge-labeled pushdown automaton of order i.

Likewise, the tree $T_{\mathfrak{n}1}$ from Examples 2.2 and 2.11 was generated both by a node-labeled pushdown automaton of order 1, and by a safe recursion scheme of order 1. By Theorem 3.3 also the tree $T_{\mathfrak{n}3}$ from Example 2.3, generated by a safe recursion scheme of order 3, can be generated by some node-labeled pushdown automaton of order 3. □

At the first glance it may seen that Theorem 3.1 can be obtained by directly composing the above two theorems, but the settings of edge-labeled and node-labeled pushdown automata are not immediately compatible. Indeed, besides the superficial syntactical difference between edge-labeled graphs from Theorem 3.2 (and trees being their unfoldings) and node-labeled trees from Theorem 3.3 we have two problems. First, the number of children of a node in an edge-labeled tree may be infinite, while in a node-labeled tree it is always finite, and moreover bounded in the whole tree. In this aspect, node-labeled trees are similar to deterministic edge-labeled trees instead of all edge-labeled trees. To deal with this, we use a result from Carayol and Wöhrle [5, Theorem 2].

Theorem 3.5. *For every $G \in Graph(n)$, where $n \geq 1$, there exists a tree T that is an unfolding of a deterministic graph $G_{n-1} \in Graph(n-1)$, and an* MSO*-interpretation*[1] *I such that $G = I(T)$.* □

Remark 3.6. We say here that T is an unfolding of a deterministic graph $G_{n-1} \in Graph(n-1)$. This is a little bit stronger than saying that T is deterministic and belongs to $Tree(n)$. Indeed, the former implies the latter; from the latter it follows that T is an unfolding of a graph $G_{n-1} \in Graph(n-1)$, but it is possible that T is deterministic while G_{n-1} is not (not all nodes of G_{n-1} have to contribute to the unfolding T).

Example 3.7. In order to prove that the graph G_2 from Figure 5 belongs to $Graph(2)$, we have shown in Example 2.1 that it can be MSO-interpreted in

[1] Carayol and Wöhrle speak about an inverse rational mapping, which is a special case of an MSO-interpretation.

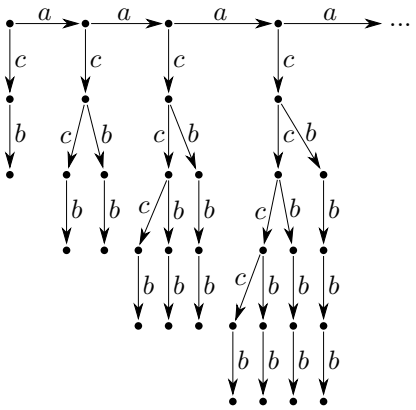


Figure 11: A tree T_2'

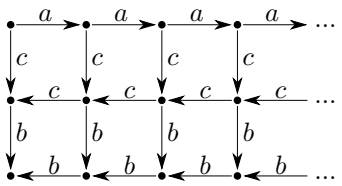


Figure 12: A graph G_1', whose unfolding is T_2'

the tree T_2 from Figure 4. The tree T_2 is not deterministic. Moreover, it has nodes with arbitrarily large arity (the nodes depicted in the top row). Thus, it is impossible to represent T_2 directly as a node-labeled tree generated by some node-labeled pushdown automaton, because every node-labeled pushdown automaton enforces some upper bound on the arity of nodes.

However, instead of MSO-interpreting G_2 in T_2, we can MSO-interpret this graph in some other tree, for example in T_2' depicted in Figure 11. In this tree, instead of a single node with k outgoing b-labeled edges, we have k nodes with a single b-labeled edge leaving every node; these nodes are connected by c-labeled edges. We can modify the MSO-interpretation from Example 2.1 so that it will work well with T_2'. To this end, we additionally require in $\varphi_{E_a}(x, y)$ and $\varphi_{E_b}(x, y)$ that x and y are not targets of a c-labeled edge, and we replace subformulae $conn_b(x, y)$ by formulae saying that y is reachable from x via b- or c-labeled edges.

The tree T_2' is an unfolding of the graph G_1' depicted in Figure 12. It is not difficult to see that $G_1' \in Graph(1)$. To this end, we can MSO-interpret G_1' in a tree similar to T_1 from Figure 2, but with three rows of nodes instead of two. □

A second problem is that an edge-labeled pushdown automaton of order n generating a deterministic graph need not to be deterministic itself (and only deterministic edge-labeled automata can be easily turned into node-labeled au-

tomata). We thus need a result from Parys [24, Theorem 1.1] (proved also in Carayol's PhD thesis [22, Corollary 3.5.3]). In the theorem below, we say that an edge-labeled pushdown automaton is *nonblocking* if whenever for a reachable configuration (p, s) there is a transition $(p, top(s), a, q, op)$, then the operation op can be applied to s (this means that the automaton never tries to pop from a stack having only a single element).

Theorem 3.8. *If a deterministic graph is generated by some edge-labeled pushdown automaton of order n, then it is also generated by some nonblocking deterministic edge-labeled pushdown automaton of order n.* □

Example 3.9. The most natural edge-labeled pushdown automaton of order 1 generating the graph G'_1 is obtained by changing the letters labeling transitions in the automaton $\mathcal{A}_1$ from Example 2.9 (and changing the alphabet to $\{a, b, c\}$). Namely, the transition relation should become

$$\begin{aligned}\Delta'_1 = \{&(q_1, \bot, a, q_1, push^1), (q_1, \bot, c, q_2, id), (q_2, \bot, c, q_2, pop^1),\\ &(q_2, \bot, b, q_3, id), (q_3, \bot, b, q_3, pop^1), (q_3, \bot, a, q_4, id)\}\,.\end{aligned}$$

This automaton is not deterministic, formally, but only for a very superficial reason. Namely, in the definition of a deterministic automaton, we require that for every pair (q, γ) there is either an ε-transition, or exactly one transition for every letter in the alphabet; here some of those transitions are missing. We can thus add a transition $(q_4, \bot, \varepsilon, q_4, id)$, and transitions $(q_i, \bot, d, q_4, id)$ for $(i, d) \in \{(1, b), (2, a), (3, c)\}$; this does not change the generated graph.

There are, however, other automata generating G'_1, which can be nondeterministic in a more severe sense. For example, besides the transition $(q_4, \bot, \varepsilon, q_4, id)$, we may have a transition $(q_4, \bot, \varepsilon, q_4, push^1)$. They cannot exist both in a deterministic automaton, one of them has to be removed.

Another example: assuming now that q_0 is initial (and that there is also a stack symbol γ besides $\bot$), there can be additional transitions:

$$\begin{aligned}\{&(q_0, \bot, \varepsilon, q_5, push^1), (q_5, \bot, \varepsilon, q_5, rew_\gamma), (q_5, \gamma, \varepsilon, q_5, push^1),\\ &(q_5, \gamma, \varepsilon, q_6, pop^1), (q_6, \gamma, \varepsilon, q_1, pop^1)\}\,.\end{aligned}$$

These transitions push the γ symbol arbitrarily many times, then they pop it exactly twice, and later the automaton requires that the topmost stack symbol is $\bot$. Thus, there is exactly one "successful" way of proceeding: the automaton has to push γ exactly twice. Notice, however, that such a run uses both the "conflicting" transitions, $(q_5, \gamma, \varepsilon, q_5, push^1)$ and $(q_5, \gamma, \varepsilon, q_6, pop^1)$. This means that the automaton cannot be determinised by simply removing some transitions.

The above gives us some taste of why Theorem 3.8 is nontrivial. Let us also remark that the problem is not artificial: the automata created in the proof of Theorem 3.2 for deterministic graphs can be indeed nondeterministic. □

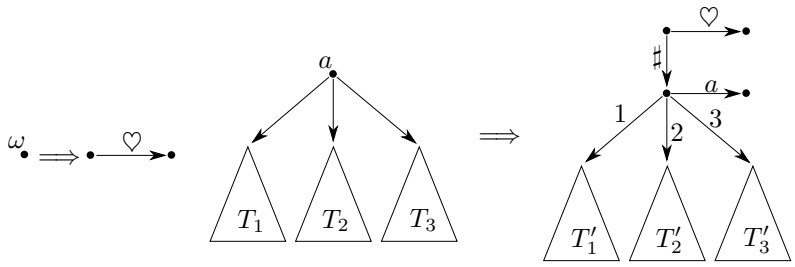


Figure 13: A transformation of a node-labeled tree T to an edge-labeled tree $tr_{\mathsf{n2e}}(T)$. Trees T'_1, T'_2, T'_3 are obtained by transforming the subtrees T_1, T_2, T_3. Recall that children in a node-labeled tree are ordered.

In order to finish a proof of Theorem 3.1, it remains to show how to switch between node-labeled pushdown automata and deterministic edge-labeled pushdown automata. We do that in the remaining part of this section. The proofs are a bit technical, but essentially straightforward.

We first show how to switch from the node-labeled setting to the edge-labeled setting. To this end, we define a transformation tr_{n2e} that maps every node-labeled tree T (which is a structure over the signature $\Xi^{nlt}_{\Sigma,k_{\max}}$) to a similar edge-labeled tree $tr_{\mathsf{n2e}}(T)$ (which is a structure over the signature $\Xi_{\Sigma\cup\{\heartsuit,\sharp,1,\ldots,k_{\max}\}}$). In order to obtain the tree $tr_{\mathsf{n2e}}(T)$ out of T we proceed as follows:

- for every a-labeled node u of T having children $u_1, \ldots, u_k$, where $a \in \Sigma$, we create in $tr_{\mathsf{n2e}}(T)$ three additional nodes v_u, w_u, z_u, and we create a $\heartsuit$-labeled edge from u to z_u, a $\sharp$-labeled edge from u to v_u, an a-labeled edge from v_u to w_u, and an i-labeled edge from v_u to u_i, for every $i \in \{1, \ldots, k\}$;
- for every ω-labeled node u of T we create in $tr_{\mathsf{n2e}}(T)$ an additional node z_u, and a $\heartsuit$-labeled edge from u to z_u.

The transformation is depicted on Figure 13.

Having defined the transformation, we prove that it is compatible with pushdown automata and with MSO-interpretations.

Lemma 3.10. *For every node-labeled pushdown automaton $\mathcal{A}_{\mathsf{n}}$ generating a tree T there exists an edge-labeled pushdown automaton $\mathcal{A}_{\mathsf{e}}$ being of the same order as $\mathcal{A}_{\mathsf{n}}$ and such that $tr_{\mathsf{n2e}}(T)$ is an unfolding of the graph generated by $\mathcal{A}_{\mathsf{e}}$.*

Proof. Let $\mathcal{A}_{\mathsf{n}} = (Q, \Sigma, \Gamma, q_I, \delta)$. As the set of states of $\mathcal{A}_{\mathsf{e}}$ we take $(Q\times\{0,1\})\uplus\{q_A, q_B\}$ (the states of the form (q,i) are needed for splitting an edge to a child into two edges, and the states q_A, q_B are needed for creating the additional a-labeled and $\heartsuit$-labeled edges). The input alphabet of $\mathcal{A}_{\mathsf{e}}$ is $\Sigma\uplus\{\heartsuit,\sharp,1,\ldots,k_{\max}\}$, where $k_{\max}$ is the maximal number k for which there is a transition of the form $\delta(p,\gamma) = (a, q_1, \ldots, q_k)$ in $\mathcal{A}_{\mathsf{n}}$. The stack alphabet of $\mathcal{A}_{\mathsf{e}}$ remains Γ, and the initial state becomes $(q_I, 0)$. We modify the transitions as follows:

- for every state $q \in Q$ and every stack symbol $\gamma \in \Gamma$, we create transitions $((q,0),\gamma,\heartsuit,q_A,id)$ and $((q,0),\gamma,\sharp,(q,1),id)$;
- for every stack symbol $\gamma \in \Gamma$, we create a transition $(q_A,\gamma,\sharp,q_B,id)$;

- whenever $\delta(p,\gamma) = (q, op)$ is a transition of $\mathcal{A}_{\mathsf{n}}$, in $\mathcal{A}_{\mathsf{e}}$ we create a transition $((p,1),\gamma,\varepsilon,(q,1),op)$;
- whenever $\delta(p,\gamma) = (a, q_1, \ldots, q_k)$ is a transition of $\mathcal{A}_{\mathsf{n}}$, in $\mathcal{A}_{\mathsf{e}}$ we create transitions $((p,1),\gamma,i,(q_i,0),id)$ for $i \in \{1,\ldots,k\}$, and $((p,1),\gamma,a,q_A,id)$.

It is not difficult to see that $\mathcal{A}_{\mathsf{e}}$ actually generates a graph whose unfolding (from the initial configuration) is $tr_{\mathsf{n2e}}(T)$. Indeed, for every configuration (p,s) of $\mathcal{A}_{\mathsf{n}}$ we have one of the following two cases:

- One possibility is that from (p,s) the automaton $\mathcal{A}_{\mathsf{n}}$ creates an ω-labeled node of T, that is, while performing transitions from (p,s) it never reaches a configuration (q,t) with $\delta(q, top(t))$ being of the form $(a, q_1, \ldots, q_k)$. Then from $((p,0),s)$ in $\mathcal{A}_{\mathsf{e}}$ we have a $\heartsuit$-labeled transition to (q_A,s), and a $\sharp$-labeled transition to $((p,1),s)$. From (q_A,s) there is a $\sharp$-labeled transition to (q_B,s), and then the automaton gets stuck. From $((p,1),s)$ the automaton $\mathcal{A}_{\mathsf{e}}$ can perform some ε-transitions, simulating the transitions of $\mathcal{A}_{\mathsf{n}}$, but it never reaches a configuration from which some letter can be read. In consequence, in the ε-closure (i.e., in the generated graph) there remain the configurations $((p,0),s)$ and (q_A,s), together with the $\heartsuit$-labeled edge; the configurations $((p,1),s)$ and (q_B,s) get removed (together with the $\sharp$-labeled edges leading to them).
- The other possibility is that from (p,s) the automaton $\mathcal{A}_{\mathsf{n}}$ creates an a-labeled node of T for some $a \in \Sigma$, that is, that $(p,s) \to^*_{\mathcal{A}_{\mathsf{n}}} (q,t)$ for a configuration with $\delta(q, top(t)) = (a, q_1, \ldots, q_k)$. Then again from $((p,0),s)$ in $\mathcal{A}_{\mathsf{e}}$ we have a $\heartsuit$-labeled transition to (q_A,s), and a $\sharp$-labeled transition to $((p,1),s)$. From (q_A,s) there is a $\sharp$-labeled transition to (q_B,s), and then the automaton gets stuck. From $((p,1),s)$ the automaton $\mathcal{A}_{\mathsf{e}}$ (without reading any letters) simulates the transitions of $\mathcal{A}_{\mathsf{n}}$, leading to $((q,1),t)$. From this configuration there is an a-labeled transition to (q_A,t), then a $\sharp$-labeled transition to (q_B,t), and then the automaton gets stuck. Because the configurations (q_B,s) and (q_B,t) have no successors, they are not present in the ε-closure; what remains is a $\heartsuit$-labeled edge from $((p,0),s)$ to (q_A,s), a $\sharp$-labeled edge from $((p,0),s)$ to $((q,1),t)$, and an a-labeled edge from $((q,1),t)$ to (q_A,t). From $((q,1),t)$ the automaton also has an i-labeled transition to $((q_i,0),t)$, for every $i \in \{1,\ldots,k\}$. This edge remains in the ε-closure, since from $((q_i,0),t)$ at least the $\heartsuit$ letter can be read. □

Lemma 3.11. *Let $\mathcal{L} \in \{\mathrm{MSO}, \mathrm{MSO{+}U}\}$. If $G = I(T)$ for some $\mathcal{L}$-interpretation, and for some node-labeled tree T, then there exists an $\mathcal{L}$-interpretation I' such that $G = I'(tr_{\mathsf{n2e}}(T))$.*

Proof. In order to obtain I' out of I, in every formula $\varphi_{E_a}(x,y)$,

- we additionally say that in both x and y there starts a $\heartsuit$-labeled edge (i.e., that $\exists z.\, E_\heartsuit(x,z) \wedge \exists z.\, E_\heartsuit(y,z)$ holds); nodes satisfying these requirements are exactly the nodes that come from T;
- likewise, we relativize the quantification in every formula of I' to nodes with an outgoing $\heartsuit$-labeled edge;

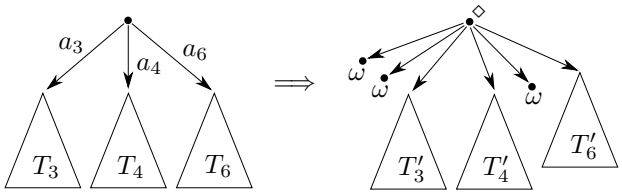


Figure 14: A transformation of a deterministic edge-labeled tree T to a node-labeled tree $tr_{\mathsf{e2n}}(T)$. It is assumed that $\Sigma = \{a_1, \ldots, a_6\}$. Trees T'_3, T'_4, T'_6 are obtained by transforming the subtrees T_3, T_4, T_6.

- moreover, we replace every atom $L_a(x)$ for $a \in \Sigma$ (checking that the node x is labeled by a) by the formula $\exists y.\, \exists z.\, (E_\sharp(x,y) \wedge E_a(y,z))$, every atom $L_\omega(x)$ (checking that x is labeled by ω) by the formula $\neg\exists y.\, E_\sharp(x,y)$, and every atom $Ch_i(x,y)$ (checking that y is the i-th child of x) by the formula $\exists z.\, (E_\sharp(x,z) \wedge E_i(z,y))$.

The equality $I(T) = I'(tr_{\mathsf{n2e}}(T))$ is straightforward. □

While proving Theorem 3.1, Lemma 3.11 is needed only for $\mathcal{L} = \mathrm{MSO}$; the version with $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}$ is useful in Section 5. The situation is the same for the next lemma, which summarizes the right-to-left part of Theorem 3.1, and simultaneously has a counterpart for the MSO+U logic.

Lemma 3.12. *Let $\mathcal{L} \in \{\mathrm{MSO}, \mathrm{MSO}{+}\mathsf{U}\}$. For every $n \geq 1$, if a graph G can be obtained by applying an $\mathcal{L}$-interpretation to a tree generated by a safe recursion scheme of order $n-1$, then G can be obtained by applying an $\mathcal{L}$-interpretation to a tree from $Tree(n)$.*

Proof. Suppose that $G = I(T)$ for some $\mathcal{L}$-interpretation I and for some safe recursion scheme $\mathcal{G}$ of order $n-1$ generating a tree T. By Theorem 3.3, T is generated by a node-labeled pushdown automaton $\mathcal{A}_{\mathsf{n}}$ of order $n-1$. Lemma 3.10 gives us then an edge-labeled pushdown automaton $\mathcal{A}_{\mathsf{e}}$ of order $n-1$ such that $tr_{\mathsf{n2e}}(T)$ is an unfolding of the graph generated by $\mathcal{A}_{\mathsf{e}}$. Moreover, by Lemma 3.11 there is an $\mathcal{L}$-interpretation I' such that $G = I'(tr_{\mathsf{n2e}}(T))$. To finish the proof, we use Theorem 3.2 to say that the graph generated by $\mathcal{A}_{\mathsf{e}}$ belongs to $Graph(n-1)$; in consequence, its unfolding $tr_{\mathsf{n2e}}(T)$ belongs to $Tree(n)$. □

We now come to the opposite direction of Theorem 3.1. In this part, for every deterministic edge-labeled tree T over an alphabet Σ we define a $\{\diamond, \omega\}$-node-labeled tree $tr_{\mathsf{e2n}}(T)$ corresponding to T. To this end, we fix some order on the letters in Σ: let $\Sigma = \{a_1, \ldots, a_k\}$. To the tree $tr_{\mathsf{e2n}}(T)$ we take all nodes of T, and we label them by $\diamond$. Moreover, we say that the edge that was a_i-labeled in T, in $tr_{\mathsf{e2n}}(T)$ leads to the i-th child (recall that children in node-labeled trees are ordered, while in edge-labeled trees they are unordered). Finally, for every node u of T, and every $i \in \{1, \ldots, k\}$ such that there is no a_i-labeled edge leaving u, as the i-th child of u in T' we attach a fresh ω-labeled node. The tr_{e2n} transformation is depicted on Figure 14.

Like previously, we have to prove that the transformation is compatible with pushdown automata and with interpretations.

Lemma 3.13. *For every nonblocking deterministic edge-labeled pushdown automaton $\mathcal{A}_{\mathsf{e}}$ generating a graph G, and for every node r of G, there exists a node-labeled pushdown automaton $\mathcal{A}_{\mathsf{n}}$ being of the same order as $\mathcal{A}_{\mathsf{e}}$ and generating the tree $tr_{\mathsf{e2n}}(Unf(G,r))$.*

Proof. Let $\mathcal{A}'_{\mathsf{e}}$ be a modification of $\mathcal{A}_{\mathsf{e}}$ that first (deterministically) reaches the configuration r using ε-transitions, and then operates as $\mathcal{A}_{\mathsf{e}}$ from r. Notice that $\mathcal{A}'_{\mathsf{e}}$ remains deterministic and nonblocking.

We now change $\mathcal{A}'_{\mathsf{e}} = (Q, \Sigma, \Gamma, q_I, \Delta)$ into a node-labeled pushdown automaton $\mathcal{A}_{\mathsf{n}}$, intended to generate $tr_{\mathsf{e2n}}(Unf(G,r))$. Let n be the order of $\mathcal{A}_{\mathsf{e}}$ (and of $\mathcal{A}'_{\mathsf{e}}$). We take $Q \cup (Q \times Op_n(\Gamma))$ as the set of states of $\mathcal{A}_{\mathsf{n}}$, and $\{\diamond\}$ as its input alphabet. The stack alphabet remains Γ, and the initial state remains q_I. We suppose here that $\Sigma = \{a_1, \ldots, a_k\}$, like in the definition of tr_{e2n}. For pairs (p,γ) being a source of ε-transitions $(p, \gamma, \varepsilon, q, op)$ we define $\delta(p,\gamma) = (q, op)$. If from a pair (p,γ) we have transitions $(p,\gamma,a_1,q_1,op_1), \ldots, (p,\gamma,a_k,q_k,op_k)$ (by the definition of a deterministic automaton, there is exactly one transition for every letter in Σ), we define $\delta(p,\gamma) = (\diamond, (q_1, op_1), \ldots, (q_k, op_k))$. Moreover, for every state $(q, op) \in Q \times Op_n(\Gamma)$ and for every $\gamma \in \Gamma$ we define $\delta((q,op),\gamma) = (q, op)$.

Let us now observe that $\mathcal{A}_{\mathsf{n}}$ indeed generates $tr_{\mathsf{e2n}}(Unf(G,r))$. To this end, we prove by coinduction that if (p,s) is a node of G (which simultaneously is a configuration of $\mathcal{A}'_{\mathsf{e}}$ and of $\mathcal{A}_{\mathsf{n}}$) then $tr_{\mathsf{e2n}}(Unf(G,(p,s)))$ equals the tree generated by $\mathcal{A}_{\mathsf{n}}$ from (p,s). When applied to the configuration r this gives what we want, because the tree generated by $\mathcal{A}_{\mathsf{n}}$ from r equals the tree generated by $\mathcal{A}_{\mathsf{n}}$ from its initial configuration $(q_I, \bot_n)$ (this is the case because $(q_I, \bot_n) \to^*_{\mathcal{A}_{\mathsf{n}}} r$).

For a step of coinduction, consider thus a node (p,s) of G. On the one hand, because (p,s) remains in the ε-closure G, in the transition relation of $\mathcal{A}'_{\mathsf{e}}$ we have some transitions $(p, top(s), a_1, q_1, op_1), \ldots, (q, top(s), a_k, q_k, op_k)$, one per every letter of Σ. On the other hand, in $\mathcal{A}_{\mathsf{n}}$ we have $\delta(p, top(s)) = (\diamond, (q_1, op_1), \ldots, (q_k, op_k))$, so the tree generated by $\mathcal{A}_{\mathsf{n}}$ from (p,s) is $\diamond\langle T'_1, \ldots, T'_k\rangle$, where T'_i are the trees generated by $\mathcal{A}_{\mathsf{n}}$ from $((q_i, op_i), s)$, for $i \in \{1, \ldots, k\}$. For every $i \in \{1, \ldots, k\}$ we have one of two cases.

- One case is that in G an a_i-labeled edge starts in (p,s). This means that from $(q_i, op_i(s))$ there is a sequence of ε-transitions to a configuration (q,t) that belongs to G. Then, in $\mathcal{A}_{\mathsf{n}}$ we have $((q_i, op_i), s) \to_{\mathcal{A}_{\mathsf{n}}} (q_i, op_i(s)) \to^*_{\mathcal{A}_{\mathsf{n}}} (q,t)$. This in particular means that the tree generated by $\mathcal{A}_{\mathsf{n}}$ from (q,t) equals T'_i, that is, the tree generated by $\mathcal{A}_{\mathsf{n}}$ from $((q_i, op_i), s)$. By the assumption of coinduction, $tr_{\mathsf{e2n}}(Unf(G,(q,t))) = T'_i$.
- The opposite case is that in G no a_i-labeled edge starts in (p,s). This means that when $\mathcal{A}'_{\mathsf{e}}$ starts performing ε-transitions from $(q_i, op_i(s))$ it only reaches configurations (q,t) for which in Δ we only have ε-transitions $(q, top(t), \varepsilon, q', op)$. The fact that $\mathcal{A}'_{\mathsf{e}}$ is nonblocking is important here: if $\mathcal{A}'_{\mathsf{e}}$ reaches a configuration (q,t) with a transition $(q, top(t), a_j, q', op)$ in

Δ, then the a_j-labeled transition could be indeed executed (op could be applied to t), so (q,t) would be a node of G (there would be an a_i-labeled edge from (p,s) to (q,t)). In consequence also in $\mathcal{A}_{\mathsf{n}}$ there is no (q,t) such that $((q_i, op_i), s) \to^*_{\mathcal{A}_{\mathsf{n}}} (q,t)$ and such that $\delta(q, top(t))$ wants to read the $\diamond$ letter; the tree generated by $\mathcal{A}_{\mathsf{n}}$ from $((q_i, op_i), s)$ is $\omega\langle\rangle$.

In the light of the above, it follows directly from the definition of the transformation tr_{e2n} that indeed $tr_{\mathsf{e2n}}(Unf(G,(p,s)))$ equals $\diamond\langle T'_1, \ldots, T'_k\rangle$. □

Lemma 3.14. *If $G = I(T)$ for some* MSO*-interpretation, and for some deterministic edge-labeled tree T, then there exists an* MSO*-interpretation I' such that $G = I'(tr_{\mathsf{e2n}}(T))$.*

Proof. In order to obtain I' out of I, in every formula $\varphi_{E_a}(x,y)$,

- we additionally say that both x and y are labeled by $\diamond$ (i.e., that $L_\diamond(x) \wedge L_\diamond(y)$ holds); nodes satisfying these requirements are exactly the nodes that come from T;
- likewise, we relativize the quantification in every formula of I' to nodes x satisfying $L_\diamond(x)$;
- moreover, we replace every atom $E_{a_i}(x,y)$ (checking that there is an a_i-labeled edge from x to y) by the atom $Ch_i(x,y)$ (checking that y is the i-th child of x).

The equality $I(T) = I'(tr_{\mathsf{e2n}}(T))$ is straightforward. □

Having all the above theorems and lemmata, it is now easy to prove Theorem 3.1.

Proof (Theorem 3.1). Suppose first that G can be obtained by applying an MSO-interpretation to a tree generated by a safe recursion scheme of order $n-1$. Then, by Lemma 3.12, it can be obtained by applying an MSO-interpretation to a tree from $Tree(n)$, which simply means that G belongs to $Graph(n)$.

For the opposite direction, consider some graph $G \in Graph(n)$. We first use Theorem 3.5 to say that there exists a tree T that is an unfolding of a deterministic graph $G_{n-1} \in Graph(n-1)$, and an MSO-interpretation I such that $G = I(T)$. By Theorem 3.2 we obtain that G_{n-1} is generated by some edge-labeled pushdown automaton $\mathcal{A}_{\mathsf{e}}$ of order $n-1$. Because of Theorem 3.8 we can assume that $\mathcal{A}_{\mathsf{e}}$ is deterministic and nonblocking. Lemma 3.13 gives us a node-labeled pushdown automaton $\mathcal{A}_{\mathsf{n}}$ of order $n-1$ generating $tr_{\mathsf{e2n}}(T)$, and Lemma 3.14 gives us an MSO-interpretation I' such that $G = I'(tr_{\mathsf{e2n}}(T))$. Finally, we use Theorem 3.3 to say that $tr_{\mathsf{e2n}}(T)$ (a tree generated by a node-labeled pushdown automaton of order $n-1$) is generated by a safe recursion scheme of order $n-1$. □

Remark 3.15. The technicality of the proof of Theorem 3.1 is a consequence of a syntactic incompatibility between the node- and edge-labeled settings. Besides the details arising from this incompatibility, in the right-to-left direction of this proof, we simply need to use right-to-left directions of Theorems 3.3 and 3.2. In the left-to-right direction, besides the left-to-right directions of Theorems 3.2 and 3.3, we needed to use also Theorems 3.5 and 3.8.

Example 3.16. We have seen in Example 2.1 a graph G_2, which belonged to $\mathit{Graph}(2)$. By Theorem 3.1, this graph can be obtained by applying an MSO-interpretation to a tree generated by a safe recursion scheme of order 1. Indeed, we show below how to MSO-interpret this graph in the tree T_{n1} from Example 2.2, generated by the recursion scheme $\mathcal{G}_1$. Recall that T_{n1} is consists of a shaft of d-labeled nodes, and below each of those nodes there is a tooth containing nodes labeled by e, f, and ω. To the k-th clique G_2 we should take all f-labeled nodes from the k-th tooth of T_{n1}. We now write consecutive formulae:

$$\varphi_1(z, z') \equiv (\mathit{Ch}_1(z, z') \vee \mathit{Ch}_2(z, z')) \wedge (L_e(z) \vee L_f(z)) \wedge (L_e(z') \vee L_f(z'))$$

says that that z, z' belong to a tooth, and that there is an edge from z to z';

$$\varphi_2(X) \equiv \forall z.\, \forall z'.\, (z \in X \wedge (\varphi_1(z, z') \vee \varphi_1(z', z)) \Rightarrow z' \in X)$$

says that X contains whole teeth;

$$\varphi_3(x, y) \equiv \forall X.\, (x \in X \wedge \varphi_2(X) \Rightarrow y \in X)\ .$$

says that x and y belong to the same tooth; finally

$$\varphi_4(x, y) \equiv \exists z_x.\, \exists z_y.\, (L_d(z_x) \wedge \mathit{Ch}_1(z_x, x) \wedge \mathit{Ch}_2(z_x, z_y) \wedge \mathit{Ch}_1(z_y, y))$$

says that x and y are topmost nodes of two consecutive teeth. In G_2, we create a-labeled edges between f-labeled nodes remaining in the same tooth:

$$\varphi_{E_a}(x, y) \equiv L_f(x) \wedge L_f(y) \wedge \varphi_3(x, y)\,,$$

and we create b-labeled edges between f-labeled nodes in consecutive teeth:

$$\varphi_{E_b}(x, y) \equiv \exists t_x.\, \exists t_y.\, (L_f(x) \wedge L_f(y) \wedge \varphi_3(x, t_x) \wedge \varphi_3(y, t_y) \wedge \varphi_4(t_y, t_x))\ . \qquad \square$$

4. Closure under MSO+$\mathsf{U}^{\mathsf{fin}}$-interpretations

We now present the main theorem of this paper.

Theorem 4.1. *For every $n \in \mathbb{N}$, if $G \in \mathit{Graph}(n)$ and if I is an* MSO+$\mathsf{U}^{\mathsf{fin}}$*-interpretation, then $I(G) \in \mathit{Graph}(n)$.*

In Section 5 we prove that Theorem 4.1 does not hold if we consider interpretations in the full MSO+U logic, already when $G \in \mathit{Tree}(2) \subseteq \mathit{Graph}(2)$. Nevertheless, we can prove the result for $n \leq 1$, as stated below.

Theorem 4.2. *For $n \in \{0, 1\}$, if $G \in \mathit{Graph}(n)$ and if I is an* MSO+U*-interpretation, then $I(G) \in \mathit{Graph}(n)$.*

Proofs of these theorems base on our previous work [25], which we recall below, in Lemma 4.3. We say that a $(\Sigma \times \Gamma)$-node-labeled tree T' enriches a Σ-node-labeled tree T, if it has the same nodes, and every node u labeled in T by some a is labeled in T' by a pair in $\{a\} \times \Gamma$.

Lemma 4.3. *Let $n \in \mathbb{N}$ and $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}^{\mathsf{fin}}$, or let $n = 0$ and $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}$. For every formula φ of $\mathcal{L}$ with free variables in $\mathcal{V}^{FO} \cup \mathcal{V}^{fin}$, and for every safe recursion scheme $\mathcal{G}$ of order n generating a tree T, there exists a safe recursion scheme $\mathcal{G}_+$ of order n that generates a tree T' enriching T, and a formula φ_{MSO} of* MSO *such that for every valuation ν in T (defined at least for all free variables of φ) it holds that $T', \nu \models \varphi_{MSO}$ if and only if $T, \nu \models \varphi$.*

Proof. For $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}^{\mathsf{fin}}$, this result was shown by Parys [25, Lemma 5.4], without observing that the resulting recursion scheme $\mathcal{G}_+$ is of the same order as $\mathcal{G}$, and that it is safe when $\mathcal{G}$ is safe. We thus need to inspect the proof, to see this. Although the proof is not so simple, it applies only two basic kinds of modifications to the recursion scheme $\mathcal{G}$, in order to obtain $\mathcal{G}_+$.

First, it uses a construction of Haddad [26, Section 4.2] (described also by Parys [27, Section B.1]) to compose a recursion scheme with a morphism into a finitary applicative structure. It is already observed by Parys [27, Lemma 10.3] that this construction preserves the order. We shall see that it preserves safety as well. To this end, we recall basic elements of this translation. First, for every type α, a new type $\alpha^\bullet$ is defined by a structural induction:

$$\mathsf{o}^\bullet = \mathsf{o} \qquad \text{and} \qquad (\beta \to \gamma)^\bullet = \underbrace{\beta^\bullet \to \cdots \to \beta^\bullet}_{k_\beta} \to \gamma^\bullet\,,$$

where k_β is some positive number depending on β. We immediately see that $ord(\alpha^\bullet) = ord(\alpha)$ (this transformation replicates some arguments, but their orders remain unchanged).

Next, there is defined a transformation on terms, again by a structural induction:

- a variable of type α is transformed into some variable of type $\alpha^\bullet$;
- a nonterminal of type α is transformed into some nonterminal of type $\alpha^\bullet$;
- a term of the form $a\langle K_1, \dots, K_k\rangle$ is transformed into a term of the form $a'\langle K'_1, \dots, K'_k\rangle$ for some letter a' (depending on the original term), and for some terms $K'_1, \dots, K'_k$, obtained by transforming $K_1, \dots, K_k$, respectively.
- a term of the form $K\,L$, where L is of type β, is transformed into a term of the form $K'\,L'_1\,\dots\,L'_{k_\beta}$, for some term K' obtained by transforming K, and for some terms $L'_1, \dots, L'_{k_\beta}$ obtained by transforming L.

Let us observe two things; both follow from the above definition by a straightforward structural induction. First, if M' is obtained by transforming a term M of type α, then M' is of type $\alpha^\bullet$; in particular $ord(M) = ord(M')$. Second, if M' has a free variable x', then M has a free variable x of some type α such that x' is of type $\alpha^\bullet$ (again, in particular $ord(x') = ord(x)$).

Suppose now that a term M' was obtained by transforming a term M. We prove, by a structural induction, that if M is safe, then also M' is safe. If M is a variable or a nonterminal, then M' as well, so M' is safe. Likewise, if M is of the form $a\langle K_1, \dots, K_k\rangle$, then the subterms $K'_1, \dots, K'_k$ of $M' = a'\langle K'_1, \dots, K'_k\rangle$ are safe by the induction assumption, so M' is safe. The only interesting case is

when M is an application, $M = K\, L_1\, \ldots\, L_k$, and all the subterms $K, L_1, \ldots, L_k$ are safe. The term after the transformation can be written as

$$M' = K'\, L'_{1,1}\, \ldots\, L'_{1,k_1} \ldots L'_{k,1}\, \ldots\, L'_{k,k_k}\,,$$

where K' is obtained by transforming K, and every $L'_{i,j}$ is obtained by transforming L_i. By the induction assumption, K' and all $L'_{i,j}$ are safe. Consider now a free variable x' of M'. As said above, M has a free variable x such that $ord(x') = ord(x)$, and $ord(M) = ord(M')$. By safety of M we have that $ord(x') = ord(x) \geq ord(M) = ord(M')$, which implies that M' is safe. This shows that the construction of Haddad [26, Section 4.2] preserves safety of the recursion scheme.

Besides the above construction, in the proof of Parys [25, Lemma 5.4], there is a second basic set of modifications applied to the recursion scheme: composition with finite tree transducers. This is realized by converting the recursion scheme to a collapsible pushdown automaton that generates the same tree [15], composing the automaton with the transducer, and then converting it back to a recursion scheme. When the original recursion scheme is safe, we can convert it to a higher-order pushdown automaton. As stated in Theorem 3.3, this preserves the order. Composing a higher-order pushdown automaton with a finite tree transducer is as easy as for collapsible pushdown automata, and clearly preserves the order. Then, the resulting pushdown automaton can be converted back to a safe recursion scheme, again using Theorem 3.3.

The above finishes the proof of Lemma 4.3 for $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}^{\mathsf{fin}}$. Let us now concentrate on the MSO+U logic, where we assume that n (i.e., the order of $\mathcal{G}$) equals 0. We base here on the same result of Parys [25, Lemma 5.4]. Its proof can be split into two parts:

1. given a node-labeled tree T, and a formula φ with free variables in $\mathcal{V}^{FO} \cup \mathcal{V}^{fin}$, the proof defines a tree T' enriching T, and a formula φ_{MSO} of MSO such that for every valuation ν in T it holds that $T', \nu \models \varphi_{MSO}$ if and only if $T, \nu \models \varphi$;
2. then, it is shown that if T is generated by a recursion scheme of order n, then T' as well.

As observed by Parys [28, Theorem 3.2], the proof of part 1 works (without any significant changes) also for formulae φ from the full MSO+U logic. The assumption that $\varphi \in \mathrm{MSO}{+}\mathsf{U}^{\mathsf{fin}}$ (i.e., the assumption that subformulae starting with U have all their free variables in $\mathcal{V}^{FO} \cup \mathcal{V}^{fin}$) is only needed for part 2. Thus, from part 1, which we are allowed to use, we obtain a tree T', a formula φ_{MSO} of MSO, and the equivalence $(T', \nu \models \varphi_{MSO}) \Leftrightarrow (T, \nu \models \varphi)$.

Moreover, as again observed by Parys [28, Theorem 3.2], in the tree T' resulting from the proof, the new part of the label of every node x of T is determined by the subtree of T starting in x. In consequence, it is easy to convert the scheme $\mathcal{G}$ (of order 0) generating T into a scheme $\mathcal{G}'$ (again of order 0) generating T'. To this end, we only need to change the labels that appear in rules of the scheme; the shape of the rules remains the same. Namely, consider a place in a rule where some label is generated, that is, a subterm of the form

$a\langle K_1, \ldots, K_k\rangle$. Let U be the tree generated by $\mathcal{G}$ from $a\langle K_1, \ldots, K_k\rangle$ (this is well defined, because the term uses no variables—the scheme is of order 0). If U is a subtree of T, we change the label a in $a\langle K_1, \ldots, K_k\rangle$ into the root's label of the corresponding subtree of T'. (Formally, it is also possible that U is not a subtree of T; this means that the subterm is not used while generating T from the starting nonterminal, so we do not need to change the label in this case.) This finishes the proof in the case of $n = 0$ and $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}$. □

Corollary 4.4. *Let $n \in \mathbb{N}$ and $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}^{\mathsf{fin}}$, or let $n = 0$ and $\mathcal{L} = \mathrm{MSO}{+}\mathsf{U}$. For every safe recursion scheme $\mathcal{G}$ of order n generating a tree T, and every $\mathcal{L}$-interpretation I evaluated in T, there exists a safe recursion scheme $\mathcal{G}_+$ of order n generating a tree T_+, and an* MSO*-interpretation I_{MSO} such that $I_{MSO}(T_+) = I(T)$.*

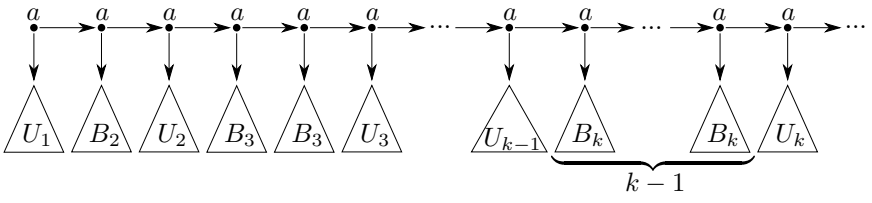
Proof. Suppose that $I = (\varphi_i)_{i\in\{1,\ldots,k\}}$. Basically, we apply Lemma 4.3 consecutively for all the formulae of I. More precisely, after $i-1$ steps (where $i \in \{1, \ldots, k\}$) we have a recursion scheme $\mathcal{G}_{i-1}$ (assuming $\mathcal{G}_0 = \mathcal{G}$) that generates a tree T_{i-1} enriching T. We modify φ_i to φ'_i that evaluated in T_{i-1} behaves like φ_i evaluated in T, that is, ignores the part of labels of T_{i-1} that was not present in T. Using Lemma 4.3 for the recursion scheme $\mathcal{G}_{i-1}$ and for the formula φ'_i we obtain a recursion scheme $\mathcal{G}_i$ that generates a tree T_i enriching T_{i-1} (hence enriching T), and an MSO formula $\varphi'_{MSO,i}$ such that for every valuation ν in T (defined at least for free variables of φ_i) it holds that $T_i, \nu \models \varphi'_{MSO,i}$ if and only if $T_{i-1}, \nu \models \varphi'_i$, that is, if and only if $T, \nu \models \varphi_i$. At the very end, for every $i \in \{1, \ldots, k\}$ we modify $\varphi'_{MSO,i}$ into $\varphi_{MSO,i}$ that ignores the part of labels of T_k appended after step i; we then have $T_k, \nu \models \varphi_{MSO,i}$ if and only if $T_i, \nu \models \varphi'_{MSO,i}$, that is, if and only if $T, \nu \models \varphi_i$. Taking $\mathcal{G}_+ = \mathcal{G}_k$, $T_+ = T_k$, and $I_{MSO} = (\varphi_{MSO,i})_{i\in\{1,\ldots,k\}}$ we have $I_{MSO}(T_+) = I(T)$, as required. All the created recursion schemes are safe and of order n. □

Remark 4.5. Lemma 4.3 and Corollary 4.4 hold also for arbitrary (not necessarily safe) recursion schemes. It is important for us, however, that if we start with a safe recursion scheme $\mathcal{G}$, the resulting recursion scheme $\mathcal{G}^+$ is safe as well.

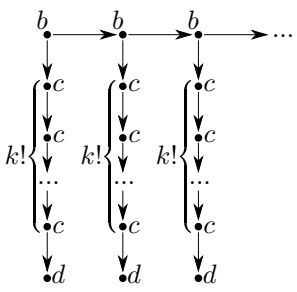
Example 4.6. Consider a tree $T_{\mathsf{sq}} = T_{1,1}$, where

$$T_{k,1} = a\langle U_k, T_{k+1,k+1}\rangle\,; \qquad T_{k,j} = a\langle B_k, T_{k,j-1}\rangle \qquad \text{for } j \geq 2\,;$$
$$U_k = b\langle F_{k!}, b\langle F_{2k!}, b\langle F_{3k!}, \ldots\rangle\rangle\rangle\,; \qquad B_k = b\langle F_{k!}, b\langle F_{k!}, b\langle F_{k!}, \ldots\rangle\rangle\rangle\,;$$
$$F_0 = d\langle\rangle\,; \qquad \text{and} \qquad F_i = c\langle F_{i-1}\rangle \qquad \text{for } i \geq 1.$$

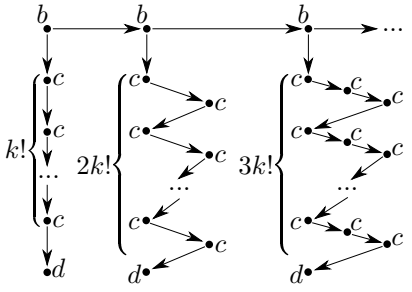
In other words, on the rightmost branch of T_{sq} we have an infinite sequence of a-labeled nodes, and to the left of every a-labeled node there is attached some comb. On the rightmost branch of each comb we have an infinite sequence of b-labeled nodes, and to the left of every b-labeled node there is attached a branch consisting of some number of c-labeled nodes, and finished with a d-labeled node. There are two kinds of combs; call them bounded combs and unbounded combs. The first comb is unbounded, and then between the $(k-1)$-th unbounded comb



(a) The tree T_{sq}



(b) A subtree B_k



(c) A subtree U_k

Figure 15: The tree T_{sq} from Example 4.6

and the k-th unbounded comb there are $k-1$ bounded combs. In a bounded comb, located between the $(k-1)$-th and the k-th unbounded comb, all teeth contain the same number of c-labeled nodes, namely $k!$. On the other hand, in the k-th unbounded comb, the number of c-labeled nodes grows: it equals $i \cdot k!$ in the i-th tooth. The tree T_{sq} is depicted on Figure 15.

One can see that T_{sq} is generated by a safe recursion scheme $\mathcal{G}_{\mathsf{sq}}$ with the following rules:

$$\begin{gathered} S \to X\,F\,D\,C\,, \qquad X\,t\,x\,y \to t\,y\,(X\,(G\,t)\,(Y\,x)\,(x\,y))\,, \\ Y\,x\,y\,z \to y\,(x\,y\,z)\,, \qquad D\,y\,z \to y\,(y\,z)\,, \qquad C\,z \to c\langle z\rangle\,, \\ F\,y\,u \to a\langle U\,y\,d\langle\rangle, u\rangle\,, \qquad G\,t\,y\,u \to a\langle B\,(y\,d\langle\rangle), t\,y\,u\rangle\,, \\ B\,z \to b\langle z, Bz\rangle\,, \qquad U\,y\,z \to b\langle z, U\,y\,(y\,z)\rangle\,, \end{gathered}$$

where types of the variables are, respectively,

$$x, t\colon (\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}\,, \qquad y\colon \mathsf{o} \to \mathsf{o}\,, \qquad z, u\colon \mathsf{o}\,.$$

It is quite clear that in MSO+$\mathsf{U}^{\mathsf{fin}}$ we can distinguish bounded combs from unbounded combs. Indeed, we can write the following formulae:

$$\psi_i(y, X) \equiv \forall x \in X.\, \exists z.\, (Ch_i(z, x) \wedge (z \in X \vee z = y))\,,$$

for $i \in \{1, 2\}$, saying that elements of X are consecutive nodes on a branch starting directly below the node y, and going down always to the i-th child; and

$$\psi_U(x) \equiv \mathsf{U}X.\, \exists_{\mathsf{fin}} Y.\, \exists y \in Y.\, \psi_1(y, X) \wedge \psi_2(x, Y)\,.$$

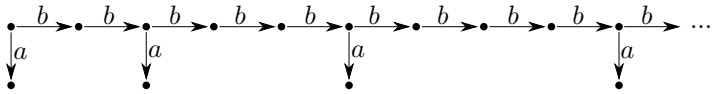


Figure 16: The graph $I(T_{\mathsf{sq}})$ from Example 4.6

The formula $\psi_U(x)$ holds for x being a root of a comb exactly when the comb is unbounded (here X is a set of nodes on a single tooth of the comb and Y is a top part of the shaft of the comb).

We now define an MSO+$\mathsf{U}^{\mathsf{fin}}$-interpretation I. For every unbounded comb we create an a-labeled edge leading to the root of this comb from its parent:

$$\varphi_{E_a}(x,y) \equiv L_a(x) \wedge Ch_1(x,y) \wedge \psi_U(y)\,.$$

Moreover, we connect all a-labeled nodes (located on the rightmost branch of T_{sq}) by b-labeled edges:

$$\varphi_{E_b}(x,y) \equiv L_a(x) \wedge Ch_2(x,y)\,.$$

In consequence, the graph $I(T_{\mathsf{sq}})$, depicted on Figure 16, consists of an infinite branch (b-labeled edges), where in some nodes there additionally starts an a-labeled edge. The distance between the $(k-1)$-th and the k-th additional edge is k, exactly like the distance between the $(k-1)$-th and the k-th unbounded comb in T_{sq}.

In our formulae we have used the U quantifier. One can prove that there is no MSO formula $\psi'_U(x)$ that holds only in roots of unbounded combs. Indeed, in MSO we can count the length of a tooth only modulo some fixed number n, but $k!$ equals $j \cdot k!$ modulo n, for every $k \geq n$. Likewise, we cannot determine which comb is bounded by counting its number from the top of the whole tree T_{sq}. (More formally, we can convert the hypothetical MSO formula $\psi'_U(x)$ into a finite automaton that reads a tree with one node marked. Then, using a pumping lemma, we can show that if the automaton accepts the tree where the root of an unbounded comb is marked, then it also accepts the tree where the root of some bounded comb is marked). Having this in mind, we conjecture that $I(T_{\mathsf{sq}})$ cannot be MSO-interpreted in T_{sq}: it is difficult to imagine how nodes in appropriate distances can be found somewhere in T_{sq}.

Nevertheless, by Corollary 4.4 there is a recursion scheme $\mathcal{G}_+$ of the same order as $\mathcal{G}_{\mathsf{sq}}$, generating a tree T_+ (from the proof we know that T_+ enriches T_{sq}), and an MSO-interpretation I_{MSO} such that $I_{MSO}(T_+) = I(T_{\mathsf{sq}})$. In our case it is easy to show appropriate $\mathcal{G}_+$, T_+, and I_{MSO}. Indeed, it is enough to use some different letter b' instead of b on shafts of unbounded combs. This way, it becomes trivial to detect whether a comb is unbounded: we can replace the MSO+$\mathsf{U}^{\mathsf{fin}}$ formula φ_{E_a} by the MSO formula

$$\varphi_{E_a}(x,y) \equiv L_a(x) \wedge Ch_1(x,y) \wedge L_{b'}(y)\,.$$

Moreover, in order to generate T_+ by a recursion scheme, we should only change b to b' in the rule for the nonterminal U. □

Having Corollary 4.4 and Theorem 3.1, we conclude the proof of our main theorem, Theorem 4.1, along with Theorem 4.2.

PROOF (THEOREMS 4.1 AND 4.2). Let $\mathcal{L}$ be the considered logic; namely, let $\mathcal{L} = \text{MSO}+\mathsf{U}^{\mathsf{fin}}$ for Theorem 4.1 and $\mathcal{L} = \text{MSO}+\mathsf{U}$ for Theorem 4.2. The class $\mathit{Graph}(0)$ contains exactly all finite graphs, and while interpreting in a finite graph we can only obtain a finite graph; this establishes the theorem for $n = 0$. We thus assume below that $n \geq 1$. In this case, the left-to-right implication of Theorem 3.1 gives us a safe recursion scheme $\mathcal{G}$ of order $n-1$ generating a tree T, and an MSO-interpretation I_2 such that $I_2(T) = G$.

Suppose that $I_2 = (\varphi_{E_a}(x_1, x_2))_{a\in\Lambda}$ and $I = (\psi_{E_\alpha}(x_1, x_2))_{\alpha\in\Sigma}$ (where I is the $\mathcal{L}$-interpretation from the statement of the theorems). We create an $\mathcal{L}$-interpretation I_3 such that $I_3(T) = I(I_2(T)) = I(G)$. To this end, in every formula ψ_{E_α} of I we replace every atomic formula $E_a(y, z)$ by the corresponding formula $\varphi_{E_a}(y, z)$ of I_2. Moreover, quantification in ψ_{E_α} should be relativized to those nodes of T that are actually taken to G, that is, to nodes y satisfying $\exists z.\ \bigvee_{a\in\Lambda}(\varphi_{E_a}(y, z) \vee \varphi_{E_a}(z, y))$ (i.e., satisfying $\varphi_{E_a}(y, z)$ or $\varphi_{E_a}(z, y)$ for some $a \in \Lambda$ and for some node z of T).

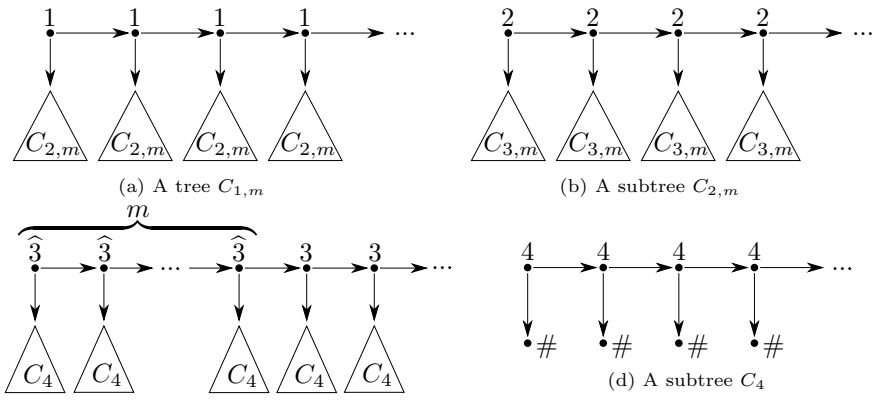
Corollary 4.4 gives us then a safe recursion scheme $\mathcal{G}_+$ of order $n-1$ generating a tree T_+, and an MSO-interpretation I_{MSO} such that $I_{MSO}(T_+) = I_3(T) = I(G)$. We conclude that $I(G) \in \mathit{Graph}(n)$ by the right-to-left implication of Theorem 3.1. □

5. MSO+U-Interpretations Lead to Difficult Graphs

In this section we consider the full MSO+U logic, for which we prove the following theorem.

Theorem 5.1. *There is a tree $T \in \mathit{Tree}(2)$ and an* MSO+U*-interpretation I such that $I(T)$ is a graph with an undecidable* MSO *theory; in consequence, $I(T) \notin \mathit{Graph}(n)$ for any $n \in \mathbb{N}$.*

One can expect such a result, since the MSO+U logic is undecidable over infinite words [20]. We remark, though, that undecidability of a logic does not automatically imply that the logic can define some complicated ("undecidable") sets. For example, over rational numbers the MSO logic with quantification over cuts (real numbers) defines the same sets as the standard MSO logic quantifying only over rational numbers [29], but the latter logic is decidable while the former is not [30, 31]. However, using arguments from topological complexity one can easily see that MSO+U is more expressive than MSO+$\mathsf{U}^{\mathsf{fin}}$: it is known that MSO+U can define sets located arbitrarily high in the projective hierarchy [32], while the topological complexity of MSO+$\mathsf{U}^{\mathsf{fin}}$ can be bounded using the automata model given by Parys [25]. Nevertheless, expressivity of the logic itself does not imply anything in the matter of interpretations: as we have seen in previous sections, MSO+$\mathsf{U}^{\mathsf{fin}}$ is more expressive than MSO, and MSO is more expressive than FO, but interpretations in these logics define the same hierarchy of graphs.



(a) A tree $C_{1,m}$

(b) A subtree $C_{2,m}$

(c) A subtree $C_{3,m}$

(d) A subtree C_4

Figure 17: Trees $C_{1,m}$ (additionally $C_1 = C_{1,0}$, $C_2 = C_{2,0}$, and $C_3 = C_{3,0}$)

While saying that a logic is undecidable, we mean undecidability of the problem in which on input we are given a sentence of the logic, and we have to tell whether this sentence holds in a fixed structure (e.g., in the infinite word or infinite binary tree, without any labels). A problem more related to Theorem 5.1 is different: the formula should be fixed, but the structure should be given on input. Theorem 5.2 says that this problem is undecidable, even for structures being regular trees (i.e., trees generated by recursion schemes of order 0).

Theorem 5.2. *There is a fixed sentence φ_{und} of* MSO+U *such that it is undecidable, given a safe recursion scheme $\mathcal{G}$ of order* 0*, whether φ_{und} holds in the tree generated by $\mathcal{G}$.*

We believe that Theorem 5.2 is interesting on its own. It complements the "standard" undecidability result for the MSO+U logic shown by Bojańczyk, Parys, and Toruńczyk [20].

Let us now present the "difficult" trees, giving undecidability results described by Theorems 5.1 and 5.2. These are combs of depth 4. Let $C_5 = \sharp\langle\rangle$, and for $k = 4, 3, 2, 1$ let C_k be the tree such that $C_k = k\langle C_{k+1}, C_k\rangle$ (labels appearing in C_1 are $1, 2, 3, 4, \sharp$). We also consider a variant of C_1, where the first m nodes on the shaft of every C_3 subtree are labeled by $\widehat{3}$ (instead of 3): let $C_{3,0} = C_3$ and $C_{3,m} = \widehat{3}\langle C_4, C_{3,m-1}\rangle$ for $m \geq 1$; let also $C_{k,m}$ for $k = 2, 1$ be the tree such that $C_{k,m} = k\langle C_{k+1,m}, C_{k,m}\rangle$. See also Figure 17.

The following lemma puts the basis for Theorems 5.1 and 5.2.

Lemma 5.3. *There is a fixed sentence φ_{und} of* MSO+U *such that it is undecidable, given a number $m \in \mathbb{N}$, whether φ_{und} holds in the tree $C_{1,m}$.*

Proof. We base on the undecidability proof from Bojańczyk et al. [20]. Let us thus recall some notions from that paper. A Minsky machine is a (possibly nondeterministic) device which has a finite state space, and two counters that can be incremented, decremented, and tested for zero.

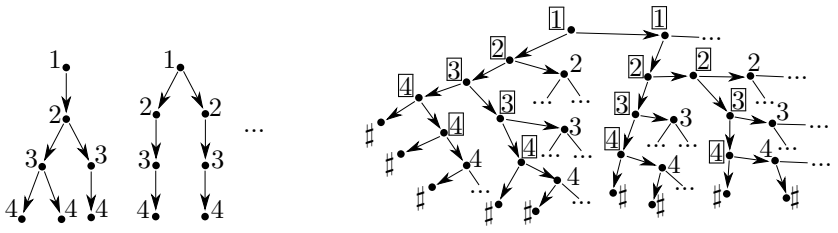


Figure 18: An example sequence S of depth-4 trees (a), its word representation $wr(S)$ (b), and the corresponding well-formed set X of nodes of C_1 (i.e., such that $S = seq(X)$), where labels of nodes in X are surrounded by a box (c)

Bojańczyk et al. [20] encode runs of Minsky machines in infinite sequences of finite trees of depth 4. These are ordered trees (i.e., children of every node are numbered). Depth of a node is defined as its distance from the root, plus one (i.e., the root is at depth 1). All leaves are at depth (exactly) 4.

Let ρ be a run of of a Minsky machine, consisting of d configurations. We assume, by definition, that a run starts in the initial state, and ends in an accepting state; we, however, do not require anything about values of the counters in the first or in the last configuration (the machine can itself ensure that they are zero by applying a zero test, but it does not have to do that). We say that a vector of natural numbers $(n_1, \ldots, n_{2d})$ *describes* ρ if, for $i \in \{1, \ldots, d\}$, the numbers n_{2i-1}, n_{2i} store the value of the two counters in the i-th configuration of ρ. An infinite sequence S of depth-4 trees *encodes* ρ if

- the degree (i.e., the number of children) of depth-3 nodes tends to infinity;
- all but finitely many depth-1 nodes have the same even degree $2d$;
- for every $i \in \{1, \ldots, 2d\}$, all but finitely many depth-2 nodes that are an i-th child have the same degree, call it n_i;
- $(n_1 - 1, \ldots, n_{2d} - 1)$ describes ρ.

Note that S does not fully specify ρ: the list of visited states is missing.

Sequences of depth-4 trees are then represented in infinite words over $\{1, 2, 3, 4\}$. The *word representation* of such a sequence S, denoted $wr(S)$, is the (unique) infinite word obtained by listing labels of all nodes of trees in S, in the prefix order (i.e., a node before its descendants). Consult Figure 18(a,b) for an example.

From Bojańczyk et al. [20, Lemma 3.2] we have the following theorem.

Theorem 5.4. *For every Minsky machine M there is a sentence φ_M of* MSO+U *which is true in the word $wr(S)$ (for an infinite sequence S of depth-4 trees) if and only if S encodes a run of M.*[2] □

[2] Bojańczyk et al. [20, Lemma 3.2] say how the formula works for all words over $\{1, 2, 3, 4\}$, while here we only say how it works for words of the form $wr(S)$.

We assume here that an infinite word is a structure over the signature consisting of a unary relation $L_k(x)$, for every $k \in \{1,2,3,4\}$, saying that the label at position x is k, and of a binary relation $Succ(x,y)$ saying that the position y is the successor of the position x.

We now want to encode the infinite words (and, indirectly, sequences of depth-4 trees) in sets of nodes of C_1 (or of $C_{1,m}$). We say that a set X of nodes of C_1 is *well-formed* if

1. X contains all 1-labeled nodes;
2. X contains no $\sharp$-labeled nodes;
3. X is upward closed (i.e., if $y \in X$, and if y is not the root, then the parent of y belongs to X);
4. if $x \in X$ is labeled by $k \leq 3$, then its left child belongs to X; and
5. for $k \in \{2,3,4\}$, every k-labeled node has a k-labeled descendant not in X.

Likewise we define a well-formed set of nodes of $C_{1,m}$ (in this definition we treat labels 3 and $\widehat{3}$ as identical).

We now write an MSO formula $\varphi_{wf}(X)$ saying that X is well-formed. To this end, we write

$$\varphi_{sib}(x,y) \equiv \forall Y.\,(x \in Y \wedge \forall z \in Y.\,\forall z'.\,(Ch_2(z,z') \Rightarrow z' \in Y) \Rightarrow y \in Y)$$

to say that y lies below x on the same shaft of the same comb, and

$$\varphi_{par}(x,y) \equiv \exists x'.\,(Ch_1(x,x') \wedge \varphi_{sib}(x',y))$$

to say that y belongs to the shaft that starts in the left child of x. Then, we write formulae corresponding to particular point of the definition of a well-formed set:

$$\eta_1(X) \equiv \forall x.\,(L_1(x) \Rightarrow x \in X)\,, \qquad \eta_2(X) \equiv \forall x \in X.\,\neg L_\sharp(x)\,,$$
$$\eta_3(X) \equiv \forall y \in X.\,\forall x((Ch_1(x,y) \vee Ch_2(x,y)) \Rightarrow x \in X)\,,$$
$$\eta_4(X) \equiv \forall x \in X.\,\forall y(Ch_1(x,y) \wedge \neg L_\sharp(y) \Rightarrow y \in X)\,,$$
$$\eta_5(X) \equiv \forall x.\,(\neg L_\sharp(x) \Rightarrow \exists y.\,(\varphi_{par}(x,y) \wedge y \notin X))\,.$$

Finally, we take

$$\varphi_{wf}(X) \equiv \eta_1(X) \wedge \eta_2(X) \wedge \eta_3(X) \wedge \eta_4(X) \wedge \eta_5(X)\,.$$

To every well-formed set X we assign a corresponding sequence of depth-4 trees, denoted $seq(X)$, as follows: we treat nodes in X as nodes of trees in S; label k of a node specifies its depth; k-labeled nodes of X lying on the same shaft (for $k \geq 2$) become consecutive siblings, and their closest $(k-1)$-labeled ancestor becomes their parent. Notice that the mapping *seq* establishes a one-to-one correspondence between well-formed sets X and sequences of depth-4 trees. Moreover, if we list the label of all elements of X in the prefix order, we obtain the word $wr(S)$. Again, consult Figure 18 for an example.

It follows from Theorem 5.4 that for every Minsky machine M there is a formula $\varphi'_M(X)$ of MSO+U such that for every $m \in \mathbb{N}$ and for every well-formed set X, the formula $\varphi'_M(X)$ is true in the tree $C_{1,m}$ if and only if $seq(X)$

encodes a run of M. We obtain φ'_M out of φ_M from Theorem 5.4 in the following way:

- we relativize quantification to those nodes that belong to X;
- we replace every atom $Succ(x, y)$ by a formula saying that y is the successor of x while visiting the set X according to the prefix order (clearly this can be written in MSO); and
- we replace every atom $L_3(x)$ by $L_3(x) \vee L_{\widehat{3}}(x)$ (atoms $L_k(x)$ for $k \neq 3$ remain unchanged).

We also construct a formula $\varphi_{ch}(X)$ such that for every $m \in \mathbb{N}$ and for every well-formed set X, the formula $\varphi_{ch}(X)$ is true in the tree $C_{1,m}$ if and only if every depth-2 node of $seq(X)$ that is a first child has degree m. This is possible, because in $C_{1,m}$ the first m nodes of every order-3 shaft are marked with $\widehat{3}$. We can thus write

$$\varphi_{ch}(X) \equiv \forall x.\, \forall y.\, \forall z.\, (L_1(x) \wedge Ch_1(x, y) \wedge \varphi_{par}(y, z) \Rightarrow (L_{\widehat{3}}(z) \Leftrightarrow z \in X))\,.$$

To finish the proof, take a Minsky machine M_{und} such that the following problem is undecidable: given a number $m \in \mathbb{N}$, does there exist a run of M_{und} starting in a configuration where the value of the first counter is m. Such a machine clearly exists: one can take a Minsky machine simulating a universal Turing machine, where the input to the latter is encoded in the value of the first counter.

We take

$$\varphi_{und} \equiv \exists X.\, (\varphi_{wf}(X) \wedge \varphi'_{M_{und}}(X) \wedge \varphi_{ch}(X))\,.$$

Observe that, for every $m \geq 1$, the sentence φ_{und} is true in $C_{1,m}$ if and only if there exists a run of M_{und} starting in a configuration where the value of the first counter is $m - 1$. Indeed, suppose that φ_{und} is true in $C_{1,m}$. Let X be a set that makes the sentence true. The subformula $\varphi_{wf}(X)$ ensures that X is well-formed, and the subformula $\varphi'_{M_{und}}(X)$ ensures that $seq(X)$ encodes a run of M_{und}. Moreover, $\varphi_{ch}(X)$ ensures that the first coordinate of the vector that describes this run (i.e., the value of the first counter in the first configuration) is $m - 1$. Thus, there exists a run as requested. Conversely, suppose that there exists a run of M_{und} starting in a configuration where the value of the first counter is $m - 1$. Clearly there exists a sequence S of depth-4 trees that encodes this run. Moreover, we can assume that all (not only all but finitely many) depth-2 nodes that are a first child have degree m. As already said, there is a well-formed set X such that $S = seq(X)$; this set makes φ_{und} true in $C_{1,m}$.

We have thus reduced an undecidable problem to the problem of determining, given $m \in \mathbb{N}$, whether φ_{und} holds in $C_{1,m}$. It follows that the latter problem is also undecidable; this finishes the proof. □

Having established Lemma 5.3, Theorems 5.1 and 5.2 follow easily.

Proof (Theorem 5.2). We take the sentence φ_{und} from Lemma 5.3, and we reduce the question whether φ_{und} holds in $C_{1,m}$ for a given $m \in \mathbb{N}$ (undecidable

by Lemma 5.3) to our question whether φ_{und} holds in the tree generated by a given safe recursion scheme of order 0 (showing that the latter question is undecidable). To this end, we observe that given $m \in \mathbb{N}$ one can construct a recursion scheme $\mathcal{G}_m$ of order 0 that generates $C_{1,m}$. This, in turn, is rather obvious. Indeed, we can take a scheme that "simulates" the definition of $C_{1,m}$. Namely, it has nonterminals $C_{1,m}, C_{2,m}, C_4$, and $C_{3,j}$ for all $j \in \{0, \ldots, m\}$ with $C_{1,m}$ being the starting nonterminal (recall that m is fixed; the scheme, and in particular the number of nonterminals it uses, may depend on m), and rules

$$\begin{aligned}
C_{k,m} &\to k\langle C_{k+1,m}, C_{k,m}\rangle && \text{for } k \in \{1,2\},\\
C_{3,j} &\to \widehat{3}\langle C_4, C_{3,j-1}\rangle && \text{for } j \in \{1,\ldots,m\},\\
C_{3,0} &\to 3\langle C_4, C_{3,0}\rangle\,, && \text{and}\\
C_4 &\to k\langle \sharp\langle\rangle, C_k\rangle\,. &&
\end{aligned}$$

□

Proof (Theorem 5.1). We consider a tree T_0 consisting of an infinite branch, where below the $(m+1)$-th node of this branch we attach $C_{1,m}$. Formally, we define T_0 by coinduction: $T_m = a\langle C_{1,m}, T_{m+1}\rangle$ for $m \in \mathbb{N}$.

Moreover, out of the sentence φ_{und} from Lemma 5.3 we construct a formula $\varphi'_{und}(x)$ of MSO+U which is true for x being a root of a comb $C_{1,m}$ in T_0 if and only if φ_{und} is true in $C_{1,m}$. To this end, we relativize the quantification in φ_{und} to nodes being descendants of x.

Having φ'_{und}, we consider an MSO+U-interpretation I_0 consisting of two formulae:

$$\psi_a(x_1, x_2) \equiv L_a(x_1) \wedge Ch_2(x_1, x_2)\,,$$

which is true if x_1 and x_2 are consecutive nodes on the main branch, and

$$\psi_b(x_1, x_2) \equiv L_a(x_1) \wedge Ch_1(x_1, x_2) \wedge \varphi'_{und}(x_2)\,,$$

which is true if x_2 is a root of a comb $C_{1,m}$ in which φ_{und} is true, and x_1 is its parent. The effect is that $I_0(T_0)$ consists of an infinite path with a-labeled edges, where for $m \in \mathbb{N}$ such that $C_{1,m} \models \varphi_{und}$ we additionally have a b-labeled edge starting in the m-th node of that path.

The question whether $C_{1,m} \models \varphi_{und}$ for a given $m \in \mathbb{N}$ is undecidable by Lemma 5.3. However, we can easily reduce it to the question whether a given sentence ψ of MSO holds in $I_0(T_0)$. Namely, given $m \in \mathbb{N}$, we consider the sentence saying that the m-th node on the main branch of $I_0(T_0)$ has an outgoing b-labeled edge; this sentence is true in $I_0(T_0)$ if and only if $C_{1,m} \models \varphi_{und}$. This means that $I_0(T_0)$ has an undecidable MSO theory.

Next, observe that T_0 is generated by a safe recursion scheme $\mathcal{G}$ of order 1, with the following rules:

$$\begin{aligned}
S &\to T\,C_3\,, & C_1\,x &\to 1\langle C_2\,x, C_1\,x\rangle\,, & C_3 &\to 3\langle C_4, C_3\rangle\,,\\
T\,x &\to a\langle C_1\,x, T\,\widehat{3}\langle C_4, x\rangle\rangle\,, & C_2\,x &\to 2\langle x, C_2\,x\rangle\,, & C_4 &\to 4\langle\sharp\langle\rangle, C_4\rangle\,.
\end{aligned}$$

Finally, because $I_0(T_0)$ is obtained by applying an MSO+U-interpretation to a tree generated by a safe recursion scheme of order 1, from Lemma 3.12 we know that $I_0(T_0) = I(T)$ for some MSO+U-interpretation I and some tree T from $\mathit{Tree}(2)$. This establishes the first part of the theorem.

The part saying that $I(T) \notin \mathit{Graph}(n)$ for any $n \in \mathbb{N}$ follows immediately, because all graphs in $\mathit{Graph}(n)$, for all $n \in \mathbb{N}$, have a decidable MSO theory (while $I(T)$ does not). □

6. Conclusion

In this paper we have answered to the following question: what can happen, if we apply MSO+$\mathsf{U}^{\mathsf{fin}}$- or MSO+$\mathsf{U}$-interpretations to graphs on the n-th level of the Caucal hierarchy. It turns out that for MSO+$\mathsf{U}^{\mathsf{fin}}$ we can only obtain graphs being again on the same level of the Caucal hierarchy (Theorem 4.1). The same holds for the full MSO+U logic only in the case of $n \leq 1$ (Theorem 4.2). For $n \geq 2$, MSO+U-interpretations can give us graphs with an undecidable MSO theory, thus outside of the Caucal hierarchy (Theorem 5.1). As a side effect, we have also exhibited a fixed sentence of MSO+U such that it is undecidable whether this sentence holds in a given regular tree (Theorem 5.2).

Acknowledgments.

We thank Mikołaj Bojańczyk, Szymon Toruńczyk, Arnaud Carayol, and Michał Skrzypczak for discussions preceding the process of creating this paper.

References

[1] D. Caucal, On infinite terms having a decidable monadic theory, in: K. Diks, W. Rytter (Eds.), Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings, Vol. 2420 of Lecture Notes in Computer Science, Springer, 2002, pp. 165–176 (2002). doi:10.1007/3-540-45687-2_13.

[2] B. Courcelle, I. Walukiewicz, Monadic second-order logic, graph coverings and unfoldings of transition systems, Ann. Pure Appl. Logic 92 (1) (1998) 35–62 (1998). doi:10.1016/S0168-0072(97)00048-1.

[3] D. Caucal, On infinite transition graphs having a decidable monadic theory, in: F. Meyer auf der Heide, B. Monien (Eds.), Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings, Vol. 1099 of Lecture Notes in Computer Science, Springer, 1996, pp. 194–205 (1996). doi:10.1007/3-540-61440-0_128.

[4] T. Cachat, Higher order pushdown automata, the Caucal hierarchy of graphs and parity games, in: J. C. M. Baeten, J. K. Lenstra, J. Parrow, G. J. Woeginger (Eds.), Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June

30 - July 4, 2003. Proceedings, Vol. 2719 of Lecture Notes in Computer Science, Springer, 2003, pp. 556–569 (2003). doi:10.1007/3-540-45061-0_45.

[5] A. Carayol, S. Wöhrle, The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata, in: P. K. Pandya, J. Radhakrishnan (Eds.), FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings, Vol. 2914 of Lecture Notes in Computer Science, Springer, 2003, pp. 112–123 (2003). doi:10.1007/978-3-540-24597-1_10.

[6] J. Engelfriet, Iterated stack automata and complexity classes, Inf. Comput. 95 (1) (1991) 21–75 (1991). doi:10.1016/0890-5401(91)90015-T.

[7] T. Knapik, D. Niwiński, P. Urzyczyn, Higher-order pushdown trees are easy, in: M. Nielsen, U. Engberg (Eds.), Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings, Vol. 2303 of Lecture Notes in Computer Science, Springer, 2002, pp. 205–222 (2002). doi:10.1007/3-540-45931-6_15.

[8] I. Walukiewicz, Monadic second-order logic on tree-like structures, Theor. Comput. Sci. 275 (1-2) (2002) 311–346 (2002). doi:10.1016/S0304-3975(01)00185-2.

[9] B. Courcelle, Monadic second-order definable graph transductions: A survey, Theor. Comput. Sci. 126 (1) (1994) 53–75 (1994). doi:10.1016/0304-3975(94)90268-2.

[10] B. Courcelle, J. Engelfriet, Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, Vol. 138 of Encyclopedia of mathematics and its applications, Cambridge University Press, 2012 (2012). URL http://www.cambridge.org/fr/knowledge/isbn/item5758776/

[11] T. Colcombet, A combinatorial theorem for trees, in: L. Arge, C. Cachin, T. Jurdziński, A. Tarlecki (Eds.), Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wrocław, Poland, July 9-13, 2007, Proceedings, Vol. 4596 of Lecture Notes in Computer Science, Springer, 2007, pp. 901–912 (2007). doi:10.1007/978-3-540-73420-8_77.

[12] M. Bojańczyk, A bounding quantifier, in: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings, Vol. 3210 of Lecture Notes in Computer Science, Springer, 2004, pp. 41–55 (2004). doi:10.1007/978-3-540-30124-0_7.

[13] M. Bojańczyk, Weak MSO with the unbounding quantifier, Theory Comput. Syst. 48 (3) (2011) 554–576 (2011). doi:10.1007/s00224-010-9279-2.

[14] M. Bojańczyk, S. Toruńczyk, Weak MSO+U over infinite trees, in: C. Dürr, T. Wilke (Eds.), 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France, Vol. 14 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 648–660 (2012). doi:10.4230/LIPIcs.STACS.2012.648.

[15] M. Hague, A. S. Murawski, C. L. Ong, O. Serre, Collapsible pushdown automata and recursion schemes, in: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA, IEEE Computer Society, 2008, pp. 452–461 (2008). doi:10.1109/LICS.2008.34.

[16] C. L. Ong, On model-checking trees generated by higher-order recursion schemes, in: 21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings, IEEE Computer Society, 2006, pp. 81–90 (2006). doi:10.1109/LICS.2006.38.

[17] P. Parys, On the significance of the collapse operation, in: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012, IEEE Computer Society, 2012, pp. 521–530 (2012). doi:10.1109/LICS.2012.62.

[18] T. Colcombet, C. Löding, Transforming structures by set interpretations, Log. Meth. Comput. Sci. 3 (2) (2007). doi:10.2168/LMCS-3(2:4)2007.

[19] V. Penelle, Rewriting higher-order stack trees, Theory Comput. Syst. 61 (2) (2017) 536–580 (2017). doi:10.1007/s00224-017-9769-6.

[20] M. Bojańczyk, P. Parys, S. Toruńczyk, The MSO+U theory of $(\mathbb{N}, <)$ is undecidable, in: N. Ollinger, H. Vollmer (Eds.), 33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France, Vol. 47 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 21:1–21:8 (2016). doi:10.4230/LIPIcs.STACS.2016.21.

[21] P. Parys, Extensions of the Caucal hierarchy?, in: C. Martín-Vide, A. Okhotin, D. Shapira (Eds.), Language and Automata Theory and Applications - 13th International Conference, LATA 2019, St. Petersburg, Russia, March 26-29, 2019, Proceedings, Vol. 11417 of Lecture Notes in Computer Science, Springer, 2019, pp. 368–380 (2019). doi:10.1007/978-3-030-13435-8_27.

[22] A. Carayol, Automates infinis, logiques et langages, Ph.D. thesis, Université de Rennes 1 (2006).

[23] Łukasz Czajka, Coinduction: an elementary approach, CoRR abs/1501.04354 (2015). arXiv:1501.04354.

[24] P. Parys, Variants of collapsible pushdown systems, in: P. Cégielski, A. Durand (Eds.), Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012, September 3-6, 2012, Fontainebleau, France, Vol. 16 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 500–515 (2012). doi:10.4230/LIPIcs.CSL.2012.500.

[25] P. Parys, Recursion schemes, the MSO logic, and the U quantifier, Log. Methods Comput. Sci. 16 (1) (Feb. 2020). doi:10.23638/LMCS-16(1:20)2020.

[26] A. Haddad, IO vs OI in higher-order recursion schemes, in: D. Miller, Z. Ésik (Eds.), Proceedings 8th Workshop on Fixed Points in Computer Science, FICS 2012, Tallinn, Estonia, 24th March 2012., Vol. 77 of EPTCS, 2012, pp. 23–30 (2012). doi:10.4204/EPTCS.77.4.

[27] P. Parys, A type system describing unboundedness, Discret. Math. Theor. Comput. Sci. 22 (4) (Aug. 2020).
URL https://dmtcs.episciences.org/6716

[28] P. Parys, Compositionality of the MSO+U logic, CoRR abs/2005.02384 (2020). arXiv:2005.02384.

[29] T. Colcombet, Composition with algebra at the background - on a question by Gurevich and Rabinovich on the monadic theory of linear orderings, in: A. A. Bulatov, A. M. Shur (Eds.), Computer Science - Theory and Applications - 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings, Vol. 7913 of Lecture Notes in Computer Science, Springer, 2013, pp. 391–404 (2013). doi:10.1007/978-3-642-38536-0_34.

[30] S. Shelah, The monadic theory of order, Ann. Math. 102 (3) (1975) 379–419 (1975). doi:10.2307/1971037.

[31] Y. Gurevich, S. Shelah, Monadic theory of order and topology in ZFC, Ann. of Math. Logic 23 (2) (1982) 179–198 (1982). doi:10.1016/0003-4843(82)90004-3.

[32] S. Hummel, M. Skrzypczak, The topological complexity of MSO+U and related automata models, Fundam. Inform. 119 (1) (2012) 87–111 (2012). doi:10.3233/FI-2012-728.