Extensions of the Caucal Hierarchy?

Paweł Parys(✉)

Institute of Informatics, University of Warsaw, Warsaw, Poland
parys@mimuw.edu.pl

Abstract. The Caucal hierarchy contains graphs that can be obtained from finite 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. 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 undecidable MSO theory, hence outside of the Caucal hierarchy.

Keywords: Caucal hierarchy · Boundedness · WMSO+U logic

1 Introduction

This paper concerns the class of finitely describable infinite graphs introduced in Caucal [9], called a Caucal hierarchy. Graphs on consecutive levels of this hierarchy are obtained from finite graphs by alternately applying the unfolding operation [14] and inverse rational mappings [8]. Since both these operations preserve decidability of the monadic second-order (MSO) theory, graphs in the Caucal hierarchy have decidable MSO theory. It turns out that this class of graphs has also other definitions. It was shown [5,7] that the Caucal hierarchy contains exactly ε-closures of configuration graphs of all higher-order pushdown automata [15]; while generating trees, these automata are in turn equivalent to a subclass of higher-order recursion schemes called safe schemes [19]. Moreover, Carayol and Wöhrle [7] prove that the defined classes of graphs do not change if we replace the unfolding operation by the treegraph operation [26], and similarly, if we replace inverse rational mappings by the stronger operation of MSO-transductions [13]. One can also replace inverse rational mappings by the operation of FO-interpretations, assuming that the FO formulae have access to the descendant relation [10].

In this paper we try to replace inverse rational mappings or MSO-interpretations in the definition of the Caucal hierarchy by interpretations in

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

C. Martín-Vide et al. (Eds.): LATA 2019, LNCS 11417, pp. 368–380, 2019.
https://doi.org/10.1007/978-3-030-13435-8_27

some extensions of the MSO logic. Namely, we investigate logics obtained from MSO by adding the unbounding quantifier U introduced by Bojańczyk [1]. 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 variables cannot represent infinite sets (this fragment subsumes the more known WMSO+U logic in which all monadic variables can only represent finite sets [2,4]). We prove 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, but 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 [17]) is such a class: these trees have decidable MSO theory [20], and some of them are not contained in the Caucal hierarchy [24]. 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 [12,25].

Going further, we also check the full MSO+U logic, where the use of the U quantifier is unrestricted. For this logic we obtain graphs outside of the Caucal hierarchy; among them there are graphs with undecidable MSO theory. This is very expected, since the MSO+U logic is undecidable itself [3].

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 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 $\mathcal{S}$, 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 [5–9] 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 monadic variables representing finite sets, and $\mathcal{V}^{inf}$ of monadic variables representing arbitrary sets. First-order variables are denoted using lowercase letters $x, y, \ldots$, and monadic 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 monadic 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 monadic quantifiers $\mathsf{U}X$, $\exists_{\mathsf{fin}} X$, and $\forall_{\mathsf{fin}} X$ for $X \in \mathcal{V}^{fin}$, and the monadic 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 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 of the form $\mathsf{U}X.\psi$: 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: $\mathsf{U}X.\psi$ says that ψ is satisfied for arbitrarily large finite sets X).

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 $\mathcal{L}$-formulas $\varphi_R(x_1, \ldots, x_n)$ 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).

2.2 Graphs and the Caucal Hierarchy

We consider directed, edge-labeled graphs. 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 vertices, 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 vertices, 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 vertex $w \in V^G$ such that $(v, w) \in E_a^G$.

A *path* from a vertex u to a vertex v labeled by $w = a_1 \dots a_n$ is a sequence $v_0 a_1 v_1 \dots a_n v_n \in V^G(\Sigma V^G)^*$, where $v_0 = u$, and $v_n = v$, and $(v_{i-1}, v_i) \in E_{a_i}^G$ for all $i \in \{1, \dots, n\}$. A graph is called an (edge-labeled) *tree* when it contains a vertex r, called the *root*, such that for every vertex $v \in V^G$ 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 vertex $r \in V^G$ 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 [7] as a definition. We define $Graph(0)$ to be the class containing all finite Σ-labeled graphs, for all finite sets of labels Σ. For all $n \geq 0$, we let

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

We do not distinguish between isomorphic graphs.

2.3 Higher-Order Recursion Schemes

The set of *sorts* (aka. simple types) is constructed from a unique ground sort o using a binary operation $\to$; namely o is a sort, and if α and β are sorts, 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 sort α, denoted $ord(\alpha)$ is defined by induction: $ord(\mathsf{o}) = 0$ and $ord(\alpha_1 \to \dots \to \alpha_k \to \mathsf{o}) = \max_i(ord(\alpha_i)) + 1$ for $k \geq 1$.

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

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

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

A *(higher-order) recursion scheme* is a tuple $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, where Σ is a finite set of symbols, $\mathcal{N}$ a finite set of sorted nonterminals, and $\mathcal{R}$ a function assigning to every nonterminal $N \in \mathcal{N}$ of sort $\alpha_1 \to \dots \to \alpha_k \to \mathsf{o}$ a rule of the form $N\,x_1 \dots x_k \to K$, where the sorts of variables $x_1, \dots, x_k$ are $\alpha_1, \dots, \alpha_k$, respectively, and K is a term of sort o over $(\Sigma, \mathcal{N}, \{x_1, \dots, x_k\})$; finally, $S \in \mathcal{N}$

is a starting nonterminal of sort o. 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: for a finite set Σ and for $r \in \mathbb{N}$, a *Σ-node-labeled tree of maximal arity r* is of the form $a\langle T_1, \dots, T_k\rangle$, where $a \in \Sigma$, and $k \leq r$, and $T_1, \dots, T_k$ are again Σ-node-labeled trees of maximal arity r. For a tree $T = a\langle T_1, \dots, T_k\rangle$, its set of vertices is defined as the smallest set such that

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

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

Having a recursion scheme $\mathcal{G}$, we define a rewriting relation $\to_{\mathcal{G}}$ among terms of sort o over $(\Sigma, \mathcal{N}, \emptyset)$: we have $N\, L_1 \dots L_k \to_{\mathcal{G}} K[L_1/x_1, \dots, L_k/x_k]$, where N is a nonterminal such that the rule $\mathcal{R}(N)$ is $N\, x_1 \dots x_k \to K$ (and where $K[L_1/x_1, \dots, L_k/x_k]$ 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 sort o over $(\Sigma, \mathcal{N}, \emptyset)$, by coinduction:

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

A tree generated by $\mathcal{G}$ (without a term specified) is the tree generated by $\mathcal{G}$ from the starting nonterminal S.

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

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

Notice that not all subterms of a safe term need to be safe. A recursion scheme is safe if right sides of all its rules are safe.

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 [7], where letters are read by transitions, and deterministic tree-generating automata of Knapik, Niwiński, and Urzyczyn [19], where there are special commands for

creating labeled tree vertices. We use the name *edge-labeled pushdown automata* for the former model, and *node-labeled pushdown automata* for the latter model. We only recall those fragments of definitions of these automata that are relevant for us.

For every $n \in \mathbb{N}$, and every finite set Γ containing a distinguished initial symbol $\bot \in \Gamma$, there are defined

- a set $PD_n(\Gamma)$ of pushdowns of order n over the stack alphabet Γ,
- an initial pushdown $\bot_n \in PD_n(\Gamma)$,
- a finite set $Op_n(\Gamma)$ of operations on these pushdowns, where every $op \in Op_n(\Gamma)$ is a partial function from $PD_n(\Gamma)$ to $PD_n(\Gamma)$, and
- a function $top\colon PD_n(\Gamma) \to \Gamma$ (returning the topmost symbol of a pushdown).

We assume that $Op_n(\Gamma)$ contains the identity operation id, mapping every element of $PD_n(\Gamma)$ to itself.

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 $c \xrightarrow{a}_{\mathcal{A}} d$. The ε-closure of such a graph G is the Σ-labeled graph obtained from G by removing all vertices 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 $a\varepsilon^*$. The graph *generated* by $\mathcal{A}$ is the ε-closure of the configuration graph of $\mathcal{A}$.

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 define when a node-labeled tree over alphabet $\Sigma \cup \{\omega\}$ 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 $(q_I, \bot_n)$.

3 Between Caucal Hierarchy and Safe Recursion Schemes

The Caucal hierarchy is closely related to safe recursion schemes. Indeed, we have the following two results, from Carayol and Wöhrle [7, Theorem 3] and Knapik et al. [19, Theorems 5.1 and 5.3].

Fact 1. *For every $n \in \mathbb{N}$, a graph G is generated by some edge-labeled pushdown automaton of order n if and only if $G \in Graph(n)$.*

Fact 2. *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.*

It looks like the connection between the Caucal hierarchy and safe recursion schemes is already established by these two facts, but the settings of edge-labeled and node-labeled pushdown automata are not immediately compatible. Indeed, beside of the superficial syntactical difference between edge-labeled graphs from Fact 1 (and trees being their unfoldings) and node-labeled trees from Fact 2 we have two problems. First, node-labeled trees are only finitely branching (and moreover deterministic), while edge-labeled trees may have infinite branching. To deal with this, we use a fact from Carayol and Wöhrle [7, Theorem 2].

Fact 3. *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 $G = I(T)$.*

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 automata). We thus need a fact from Parys [21, Theorem 1.1] (proved also in the Carayol's Ph.D. thesis [6, Corollary 3.5.3]).

Fact 4. *If a deterministic graph is generated by some edge-labeled pushdown automaton of order n, then it is also generated by some deterministic edge-labeled pushdown automaton of order n.*

Having all the recalled facts, it is now easy to prove the following lemma.

Lemma 5. *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 $n-1$.*

Proof (sketch). Suppose first that $G = I(T)$ for some MSO-interpretation I and for some safe recursion scheme $\mathcal{G}$ of order $n-1$ generating a tree T. By Fact 2, T is generated by a node-labeled pushdown automaton $\mathcal{A}$ of order $n-1$. It is a routine to switch to the formalism of edge-labeled pushdown automata, that is,

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

- change the node-labeled tree T (which is a structure over the signature $\Xi^{nlt}_{\Sigma,r}$) to a "similar" edge-labeled tree T' (which is a structure over the signature $\Xi_{\Sigma\cup\{1,\ldots,r\}}$), where every edge of T from a vertex to its i-th child becomes an i-labeled edge, and where below every a-labeled vertex u we create a fresh vertex v_u with an a-labeled edge from u to v_u;
- change the node-labeled pushdown automaton $\mathcal{A}$ to an edge-labeled pushdown automaton $\mathcal{A}'$ of the same order $n-1$ such that T' is the unfolding of the graph generated by $\mathcal{A}'$;
- change the MSO-interpretation I evaluated in T to an MSO-interpretation I' evaluated in T', such that $I'(T') = I(T)$.

Finally, we use Fact 1 to say that the graph generated by $\mathcal{A}'$ belongs to $\mathit{Graph}(n-1)$. In effect its unfolding T' belongs to $\mathit{Tree}(n)$, and hence $G = I'(T')$ belongs to $\mathit{Graph}(n)$.

For the opposite direction, consider some graph $G \in \mathit{Graph}(n)$. We first use Fact 3 to say that there exists a tree T that is an unfolding of a deterministic graph $G_{n-1} \in \mathit{Graph}(n-1)$, and an MSO-interpretation I such that $G = I(T)$. By Fact 1 we obtain that G_{n-1} is generated by some edge-labeled pushdown automaton $\mathcal{A}$ of order $n-1$. Because of Fact 4 we can assume that $\mathcal{A}$ is deterministic. By definition, the vertex r such that $T = \mathit{Unf}(G_{n-1}, r)$ can be arbitrary; let $\mathcal{A}'$ be a modification of $\mathcal{A}$ that first reaches configuration r using ε-transitions, and then operates as $\mathcal{A}$ from r.

We now change $\mathcal{A}'$ into a node-labeled pushdown automaton $\mathcal{A}''$. To this end, we fix some order on the letters in Σ: let $\Sigma = \{a_1, \ldots, a_k\}$. Moreover, without loss of generality we assume that for all transitions (q, γ, a, q', op) of $\mathcal{A}$ with $a \neq \varepsilon$, the operation op is id. Then, if from a pair (q, γ) we have transitions $(q, \gamma, a_1, q_1, id), \ldots, (q, \gamma, a_k, q_k, id)$, we define $\delta(q, \gamma) = (\diamond, q_1, \ldots, q_k)$, and for pairs (q, γ) being a source of ε-transitions $(q, \gamma, \varepsilon, q', op)$ we define $\delta(q, \gamma) = (q', op)$. The $\{\diamond, \omega\}$-node-labeled tree T' generated by $\mathcal{A}'$, after removing all ω-labeled vertices, and while treating an edge leading to the i-th child as a_i-labeled, equals T. It is easy to modify the interpretation I into I' such that $I'(T') = I(T) = G$. Finally, we use Fact 2 to say that T' is generated by a safe recursion scheme of order $n-1$. □

4 Closure Under MSO+$\mathsf{U}^{\mathsf{fin}}$-Interpretations

We now present the main theorem of this paper.

Theorem 6. *For every $n \in \mathbb{N}$, if $G \in \mathit{Graph}(n)$ and if I is an* MSO+$\mathsf{U}^{\mathsf{fin}}$*-interpretation, then $I(G) \in \mathit{Graph}(n)$.*

This theorem can be deduced from our previous result, which we recall now. We say that a $\Sigma \times \Gamma$-node-labeled tree T' enriches a Σ-node-labeled tree T, if it has the same vertices, and every vertex u labeled in T by some a is labeled in T' by a pair in $\{a\} \times \Gamma$.

Lemma 7. *Let $n \in \mathbb{N}$. For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula φ and 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 an* MSO *formula φ_{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. This result was shown in Parys [22, 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 [16, Sect. 4.2] (described also in Parys [23, Sect. B.1]) to compose a recursion scheme with a morphism into a finitary applicative structure. It is already observed in Parys [23, Lemma 10.2] that this construction preserves the order. It is not difficult to see that it preserves safety as well: when a subterm K is transformed into a subterm M, then their order is the same, and their sets of free variables are essentially also the same, up to the fact that every single free variable of K corresponds to multiple free variables of M, all being of the same order as the free variable of K.

The second basic kind of modifications applied to the recursion scheme is the composition with finite tree transducers. This is realized by converting the recursion scheme to a collapsible pushdown automaton generating the same tree [17], 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, which can be converted back to a safe recursion scheme; as stated in Fact 2, this preserves the order. Moreover composing a higher-order pushdown automaton with a finite tree transducer is as easy as for collapsible pushdown automata, and clearly preserves the order. □

Corollary 8. *Let $n \in \mathbb{N}$. For every safe recursion scheme $\mathcal{G}$ of order n generating a tree T, and every* MSO+$\mathsf{U}^{\mathsf{fin}}$*-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 $I = (\varphi_i)_{i \in \{1,\dots,k\}}$. Basically, we apply Lemma 7 consecutively for all the formulae of I. More precisely, after $i-1$ steps (where $i \in \{1, \dots, k\}$) 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 7 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 $i \in \{1, \dots, k\}$ we modify $\varphi'_{MSO,i}$ into $\varphi_{MSO,i}$ that ignores the part 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. □

Proof (Thorem 6). The class $Graph(0)$ 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 Lemma 5 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 $I_2 = (\varphi_a(x_1, x_2))_{a\in\Lambda}$ and $I = (\psi_\alpha(x_1, x_2))_{\alpha\in\Sigma}$ We create an MSO-interpretation I_3 such that $I_3(T) = I(I_2(T)) = I(G)$. To this end, in every formula ψ_α of I we replace every atomic formula $a(y, z)$ by the corresponding formula $\varphi_a(y, z)$ of I_2. Moreover, quantification in ψ_α should be restricted to those vertices of T that are actually taken to G, that is, to vertices y satisfying $\varphi_a(y, z)$ or $\varphi_a(z, y)$ for some $a \in \Lambda$ and some vertex z of T.

Corollary 8 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 $I(G) \in Graph(n)$ by Lemma 5. □

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

One can expect such a result, since the MSO+U logic is undecidable over infinite words [3]. 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, but the latter logic is decidable while the former is not [11]. However, using arguments from topological complexity one can easily see that MSO+U is more expressible than MSO+$\mathsf{U}^{\mathsf{fin}}$: it is known that MSO+U can define sets located arbitrarily high in the projective hierarchy [18], while the topological complexity of MSO+$\mathsf{U}^{\mathsf{fin}}$ can be bounded using the automaton model given in Parys [22]. 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.

Proof (Theorem 9, sketch). Because of Lemma 5, as the source of the interpretation I we can take a node-labeled tree T generated by a safe recursion scheme of order 2. We define the *depth-k comb* as the tree C_k such that $C_k = a\langle C_{k-1}, C_k\rangle$, where $C_0 = a\langle\rangle$. We also consider a depth-2 comb with first

i vertices marked by b: $C_{2,0} = C_2$ and $C_{2,i} = b\langle C_1, C_{2,i-1}\rangle$ for $i \geq 1$; and a depth-k comb (where $k \geq 3$) with first i vertices of every depth-2 comb marked by b: $C_{k,i} = a\langle C_{k-1,i}, C_{k,i}\rangle$.

We base on the undecidability proof from Bojańczyk, Parys, and Toruńczyk [3]. This proof, given a Minsky machine M constructs an MSO+U sentence φ_M that is true in an infinite forest of finite trees of height 3 if and only if the forest encodes a (finite) accepting run of M. Such a forest is then encoded in an infinite word, but is even easier to encode it in the depth-4 comb: we just need a set (a monadic variable) X saying which vertices of the comb appear in the considered forest (where roots of depth-k combs attached below a depth-$(k+1)$ comb represent children of the root of the latter comb).

Moreover, the recalled encoding of a run of M in the forest (checked by φ_M) requires that the arity of the first child of all (except finitely many) trees in the forest contains the initial value of the first counter, that is zero. We remove the part of φ_M saying that the initial value of the first counter is zero, and instead we add a part saying that from the first depth-2 comb in every depth-3 comb we take to X exactly left children of all b-labeled vertices. This way we obtain a sentence φ'_M (of the form $\exists X.\varphi''_M$) which, for every $i \in \mathbb{N}$, is true in $C_{4,i}$ if and only if M has an accepting run from the configuration c_i with value i in the first counter, value 0 in the second counter, and initial state.

We now consider the tree T_0 consisting of an infinite branch, where below the $(i+1)$-th node of this branch we attach $C_{4,i}$; formally, we define T_0 by coinduction: $T_i = a\langle C_{4,i}, T_{i+1}\rangle$ for $i \in \mathbb{N}$. We also consider the interpretation I_M consisting of two formulae: $\psi_a(x_1, x_2)$ that is true if x_1 and x_2 are consecutive vertices on the main branch, and $\psi_b(x_1, x_2)$ that is true if x_2 is a root of a comb $C_{4,i}$ in which φ'_M is true, and x_1 is its parent. The effect is that $I_M(T_0)$ consists of an infinite path with a-labeled edges, where for $i \in \mathbb{N}$ such that M accepts from c_i, we additionally have a b-labeled edge starting in the $(i+1)$-th vertex of that path.

Take a Minsky machine M such that the problem "given i, does M accept from c_i?" is undecidable. For such a machine, the graph $I_M(T_0)$ has undecidable MSO theory. And 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.

It remains to observe that T_0 is generated by the safe recursion scheme of order 2 with the following rules:

$$\begin{array}{lll} S \to T\,C_2 & C_4\,x \to a\langle C_3\,x, C_4\,x\rangle & C_2 \to a\langle C_1, C_2\rangle \\ T\,x \to a\langle C_4\,x, T\,b\langle C_1, x\rangle\rangle & C_3\,x \to a\langle x, C_3\,x\rangle & C_1 \to a\langle a\langle\rangle, C_1\rangle \end{array}$$

□

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

References

1. Bojańczyk, M.: A bounding quantifier. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 41–55. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30124-0_7
2. Bojańczyk, M.: Weak MSO with the unbounding quantifier. Theory Comput. Syst. **48**(3), 554–576 (2011). https://doi.org/10.1007/s00224-010-9279-2
3. Bojańczyk, M., Parys, P., Toruńczyk, S.: The MSO+U theory of (N, <) is undecidable. In: STACS, pp. 21:1–21:8 (2016). https://doi.org/10.4230/LIPIcs.STACS.2016.21
4. Bojańczyk, M., Toruńczyk, S.: Weak MSO+U over infinite trees. In: STACS, pp. 648–660 (2012). https://doi.org/10.4230/LIPIcs.STACS.2012.648
5. Cachat, T.: Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 556–569. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45061-0_45
6. Carayol, A.: Automates infinis, logiques et langages. Ph.D. thesis. Université de Rennes 1 (2006)
7. Carayol, A., Wöhrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 112–123. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24597-1_10
8. Caucal, D.: On infinite transition graphs having a decidable monadic theory. In: Meyer, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 194–205. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61440-0_128
9. Caucal, D.: On infinite terms having a decidable monadic theory. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 165–176. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45687-2_13
10. Colcombet, T.: A combinatorial theorem for trees. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 901–912. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_77
11. Colcombet, T.: Composition with algebra at the background - on a question by Gurevich and Rabinovich on the monadic theory of linear orderings. In: Bulatov, A.A., Shur, A.M. (eds.) CSR 2013. LNCS, vol. 7913, pp. 391–404. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38536-0_34
12. Colcombet, T., Löding, C.: Transforming structures by set interpretations. Log. Methods Comput. Sci. **3**(2) (2007). https://doi.org/10.2168/LMCS-3(2:4)2007
13. Courcelle, B.: Monadic second-order definable graph transductions: a survey. Theoret. Comput. Sci. **126**(1), 53–75 (1994). https://doi.org/10.1016/0304-3975(94)90268-2
14. Courcelle, B., Walukiewicz, I.: Monadic second-order logic, graph coverings and unfoldings of transition systems. Ann. Pure Appl. Logic **92**(1), 35–62 (1998). https://doi.org/10.1016/S0168-0072(97)00048-1
15. Engelfriet, J.: Iterated stack automata and complexity classes. Inf. Comput. **95**(1), 21–75 (1991). https://doi.org/10.1016/0890-5401(91)90015-T
16. Haddad, A.: IO vs OI in higher-order recursion schemes. In: FICS, pp. 23–30 (2012). https://doi.org/10.4204/EPTCS.77.4
17. Hague, M., Murawski, A.S., Ong, C.L., Serre, O.: Collapsible pushdown automata and recursion schemes. In: LICS, pp. 452–461 (2008). https://doi.org/10.1109/LICS.2008.34

18. Hummel, S., Skrzypczak, M.: The topological complexity of MSO+U and related automata models. Fundam. Inform. **119**(1), 87–111 (2012). https://doi.org/10.3233/FI-2012-728
19. Knapik, T., Niwiński, D., Urzyczyn, P.: Higher-order pushdown trees are easy. In: Nielsen, M., Engberg, U. (eds.) FoSSaCS 2002. LNCS, vol. 2303, pp. 205–222. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45931-6_15
20. Ong, C.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS, pp. 81–90 (2006). https://doi.org/10.1109/LICS.2006.38
21. Parys, P.: Variants of collapsible pushdown systems. In: CSL, pp. 500–515 (2012). https://doi.org/10.4230/LIPIcs.CSL.2012.500
22. Parys, P.: Recursion schemes, the MSO logic, and the U quantifier (submitted). https://arxiv.org/abs/1810.04763
23. Parys, P.: A type system describing unboundedness (submitted). https://hal.archives-ouvertes.fr/hal-01850934
24. Parys, P.: On the significance of the collapse operation. In: LICS, pp. 521–530 (2012). https://doi.org/10.1109/LICS.2012.62
25. Penelle, V.: Rewriting higher-order stack trees. Theory Comput. Syst. **61**(2), 536–580 (2017). https://doi.org/10.1007/s00224-017-9769-6
26. Walukiewicz, I.: Monadic second-order logic on tree-like structures. Theoret. Comput. Sci. **275**(1–2), 311–346 (2002). https://doi.org/10.1016/S0304-3975(01)00185-2