The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-order Pushdown Automata

Arnaud Carayol[1] and Stefan Wöhrle[2]

[1] IRISA Rennes, France
arnaud.carayol@irisa.fr
[2] Lehrstuhl für Informatik 7
RWTH Aachen, Germany
woehrle@informatik.rwth-aachen.de

Abstract. In this paper we give two equivalent characterizations of the Caucal hierarchy, a hierarchy of infinite graphs with a decidable monadic second-order (MSO) theory. It is obtained by iterating the graph transformations of unfolding and inverse rational mapping. The first characterization sticks to this hierarchical approach, replacing the language-theoretic operation of a rational mapping by an MSO-transduction and the unfolding by the treegraph operation. The second characterization is non-iterative. We show that the family of graphs of the Caucal hierarchy coincides with the family of graphs obtained as the ε-closure of configuration graphs of higher-order pushdown automata.
While the different characterizations of the graph family show their robustness and thus also their importance, the characterization in terms of higher-order pushdown automata additionally yields that the graph hierarchy is indeed strict.

1 Introduction

Classes of finitely generated infinite graphs enjoying a decidable theory are a strong subject of current research. Interest arises from applications in model checking of infinite structures (e.g. transition systems, unfoldings of Kripke structures) as well as from a theoretical point of view since the border to undecidability is very close and even for very regular structures many properties become undecidable.

We are interested in a hierarchy of infinite graphs with a decidable monadic second-order (MSO) theory which was introduced by D. Caucal in [7]. Starting from the class of finite graphs two operations preserving the decidability of the MSO-theory are applied, the unfolding [9] and inverse rational mappings [6]. Iterating these operations we obtain the hierarchy $(\mathrm{Graph}(n))_{n\in\mathbb{N}}$ where $\mathrm{Graph}(n)$ is the class of all graphs which can be obtained from some finite graph by an n-fold iteration of unfolding followed by an inverse rational mapping. This hierarchy of infinite graphs contains several interesting families of graphs (see [7, 17]) and has already been subject to further studies [2].

The first level contains exactly the prefix-recognizable graphs [6], which can in turn be characterized as the graphs definable in Δ_2 (the infinite binary tree) by an MSO-transduction [1], or as the ε-closure of configuration graphs of pushdown automata [16] (see [1] for an overview). We extend these characterizations to higher levels.

In Sect. 3 we show that the iteration of the treegraph operation, a variant of the tree-iteration [18], and MSO-transductions generates exactly the Caucal hierarchy. These two operations are to our knowledge the strongest graph transformations which preserve the decidability of the MSO-theory. In [17] the hierarchy was defined starting from the infinite binary tree by iterating MSO-interpretations (particular MSO-transductions) and unfolding. Since the unfolding is definable inside the graph obtained by the treegraph operation, it follows from our result these definitions are indeed equivalent.

Pushdown automata can also be seen as the first level of a hierarchy of higher-order pushdown automata, whose stack entries are not only single letters (as for level 1), but words (level 2), words of words (level 3) .... Similar hierarchies have already been considered in [14, 11, 15].

In Sect. 4 we show that a graph is on level n of the Caucal hierarchy iff it is the ε-closure of a configuration graph of a higher-order pushdown automaton of level n. This result is incorrectly attributed to [2, 7] in [17]. In [2], in the context of game simulation, only the easier direction from higher-order pushdown graphs to graphs in the hierarchy is shown. All the proofs in Sections 3 and 4 are effective.

In Sect. 5 we use the characterization of the hierarchy in terms of higher-order pushdown automata to show that it is strict. Moreover we exhibit a generator for every level, i.e. every graph on this level can be obtained from the generator by applying a rational marking and an inverse rational mapping, or an MSO-interpretation. Finally we give an example of graph with a decidable MSO-theory which is not in the hierarchy.

2 Preliminaries

2.1 Operations on Graphs and the Caucal Hierarchy

We fix a countable set A, also called *alphabet*. Let $\Sigma \subseteq A$ be a finite subset of edge labels. A *Σ-labeled graph* G is a tuple $(V^G, (E_a^G)_{a \in \Sigma})$ where V^G is a set of vertices and for $a \in \Sigma$ we denote by $E_a^G \subseteq V^G \times V^G$ the set of a-labeled edges of G. We assume that V^G is at most countable, and that there are no isolated vertices in G, i.e. for every $v \in V^G$ there exists an $w \in V^G$ such that $(v, w) \in E_a^G$ or $(w, v) \in E_a^G$ for some $a \in \Sigma$. If the graph G and the set of edge labels Σ is clear from the context we drop the superscript ^{G} and speak just of a labeled graph. A graph is called *deterministic* if $(v, w) \in E_a$ and $(v, w') \in E_a$ implies $w = w'$ for all $v, w, w' \in V$ and $a \in \Sigma$.

A *path* from a vertex u to a vertex v labeled by $w = a_1 \dots a_{n-1}$ is a sequence $v_1 a_1 \dots a_{n-1} v_n \in V(\Sigma V)^*$ such that $v_1 = u$, $v_n = v$ and $(v_i, v_{i+1}) \in E_{a_i}$ for every $i \in \{1, \dots, n-1\}$. In this case we will also write $u \xrightarrow{w} v$. A *tree* T is a graph containing a vertex r called the *root* such that for any vertex $v \in V^T$ there exists a unique path from r to v.

The *unfolding* $\mathrm{Unf}(G, r)$ 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 starting from r in G and for all $a \in \Sigma$, $(w, w') \in E_a^T$ iff $w' = w \cdot a \cdot v$ for some $v \in V^G$.

The *treegraph* $\mathrm{Treegraph}(G, \sharp)$ of a graph $G = (V, (E_a)_{a\in\Sigma})$ by a symbol $\sharp \notin \Sigma$ is the graph $G' = (V^+, (E'_a)_{a\in\Sigma\cup\{\sharp\}})$ where V^+ designates the set of all finite non-empty sequences of elements of V, for all $a \in \Sigma$ and all $w \in V^*$, $(wu, wv) \in E'_a$ iff $(u, v) \in E_a$, and $E'_\sharp = \{(wu, wuu) \mid w \in V^*,\ u \in V\}$. The tree-iteration as defined in [18] also contains a son-relation given by $\{(w, wu) \mid w \in V^* \text{ and } u \in V\}$. If G is connected then the son-relation can be defined in the treegraph.

Let $\bar{\Sigma}$ be a set of symbols disjoint from but in bijection with Σ. We extend every Σ-labeled graph G to a $(\Sigma \cup \bar{\Sigma})$-labeled graph $\bar{G}$ by adding reverse edges $E_{\bar{a}} := \{(u, v) \mid (v, u) \in E_a\}$. Let $\Gamma \subseteq A$ be a set of edge labels. A *rational mapping* is a mapping $h : \Gamma \to \mathcal{P}(\Sigma \cup \bar{\Sigma})^*$ which associates to every symbol from Γ a regular subset of $(\Sigma \cup \bar{\Sigma})^*$. If $h(a)$ is finite for every $a \in \Gamma$ we also speak of a *finite mapping*. We apply a rational mapping h to a Σ-labeled graph G by the inverse to obtain a Γ-labeled graph $h^{-1}(G)$ with $(u, v) \in E_b$ iff there is a path from u to v in $\bar{G}$ labeled by a word in $h(b)$. The set of vertices of $h^{-1}(G)$ is given implicitly by the edge relations. We also speak of $h^{-1}(G)$ as the graph obtained from G by the *inverse rational mapping* h^{-1}.

The *marking* $\mathcal{M}_\sharp(G, X)$ of a graph $G = (V, (E_a)_{a\in\Sigma})$ on a set of vertices $X \subseteq V$ by a symbol $\sharp \notin \Sigma$ is the graph $G' = (V', (E'_a)_{a\in\Sigma\cup\{\sharp\}})$ where $V' = \{(x, 0) \mid x \in V\} \cup \{(x, 1) \mid x \in X\}$, $E'_a = \{((x, 0), (y, 0)) \mid (x, y) \in E_a\}$ for $a \in \Sigma$, and $E_\sharp = \{((x, 0), (x, 1)) \mid x \in X\}$. A *rational marking* of a graph $G = (V, (E_a)_{a\in\Sigma})$ by a symbol $\sharp \notin \Sigma$ with a rational subset R over $\Sigma \cup \bar{\Sigma}$ from a vertex $r \in V$ is the graph $\mathcal{M}_\sharp\big(G, \{x \in V \mid r \xrightarrow{w}_{\bar{G}} x,\ w \in R\}\big)$. An *MSO-marking* of a graph G by a symbol $\sharp$ with an MSO-formula $\varphi(x)$ is the graph $\mathcal{M}_\sharp\left(G, \{v \in V^G \mid G \models \varphi(v)\}\right)$.

Following [7], we define $\mathrm{Graph}(0)$ to be the class containing for every finite subset $\Sigma \subseteq A$ all finite Σ-labeled graphs, and for all $n \geq 0$

$$\begin{aligned}
\mathrm{Tree}(n+1) &:= \left\{\mathrm{Unf}(G, r) \mid G \in \mathrm{Graph}(n),\ r \in V^G\right\}, \\
\mathrm{Graph}(n+1) &:= \left\{h^{-1}(T) \mid T \in \mathrm{Tree}(n+1), h^{-1} \text{ an inverse rational mapping}\right\},
\end{aligned}$$

where we do not distinguish between isomorphic graphs.

2.2 Monadic Second-order Logic and Transductions

We define the *monadic second-order logic* over Σ-labeled graphs as usual, (see e.g. [13]), i.e. we view a graph as a relational structure over the signature consisting of the binary relation symbols $(E_a)_{a\in\Sigma}$.

A formula $\varphi(X_1, \ldots, X_k)$ containing at most the free variables $X_1, \ldots, X_k$ is evaluated in $(G, \mathcal{V})$ where $G = (V, (E_a)_{a\in\Sigma})$ is a Σ-labeled graph and $\mathcal{V} : V \to \mathcal{P}(\{1, \ldots, k\})$ is a function which assigns to every vertex v of G a set $\mathcal{V}(v)$ such that $v \in X_i$ iff $i \in \mathcal{V}(v)$. We write $(G, \mathcal{V}) \models \varphi(X_1, \ldots, X_k)$, or equivalently

$G \models \varphi[V_1, \ldots, V_k]$ where $V_i := \{v \in V \mid i \in \mathcal{V}(v)\}$, if φ holds in G under the given valuation $\mathcal{V}$.

An *MSO-interpretation* of Γ in Σ is a family $\mathcal{I} = (\varphi_a(x,y))_{a \in \Gamma}$ of MSO-formulas over Σ. Applying an MSO-interpretation $\mathcal{I} = (\varphi_a(x,y))_{a \in \Gamma}$ of Γ in Σ to a Σ-labeled graph G we obtain a Γ-labeled graph $\mathcal{I}(G)$ where the edge relation $E_a^{\mathcal{I}(G)}$ is given by the pairs of vertices for which $\varphi_a(x,y)$ is satisfied in G, and $V^{\mathcal{I}(G)}$ is given implicitly as the set of all vertices occurring in the relations $E_b^{\mathcal{I}(G)}$. Note that the addition of an MSO-formula $\delta(x)$ to $\mathcal{I}$ defining the vertex set explicitly does not increase the power of an interpretation if we require that there are no isolated vertices in the resulting graph.

Interpretations cannot increase the size of a structure. To overcome this weakness the notion of a transduction was introduced, cf. [8]. Let $G = (V, (E_a)_{a \in \Sigma})$ be a Σ-labeled graph and K be a finite subset of A disjoint from Σ. A *K-copying operation* for Σ associates to G a $(\Sigma \cup K)$-labeled graph $G' = (V', (E'_a)_{a \in \Sigma \cup K})$ where $V' = V \cup (V \times K)$, $E'_a := E_a$ for $a \in \Sigma$, and $E'_b := \{(v, (v,b)) \mid v \in V\}$ for $b \in K$. An *MSO-transduction* $\mathcal{T} = (K, \mathcal{I})$ from Σ to Γ is a K-copying operation for Σ followed by an MSO-interpretation $\mathcal{I}$ of Γ in $\Sigma \cup K$.

Note that an inverse rational mapping is a special case of an MSO-interpretation and an MSO-marking is a special case of an MSO-transduction.

2.3 Higher-order Pushdown Automata

We follow the definition of [15]. Let Γ be a finite set of stack symbols. A *level 1 pushdown stack* over Γ is a word $w \in \Gamma^*$ in reversed order, i.e. if $w = a_1 \ldots a_m$ the corresponding stack is denoted by $[a_m, \ldots, a_1]$. For $n \geq 2$ a level n pushdown stack over Γ is inductively defined as a sequence $[s_r, \ldots, s_1]$ of level $n-1$ pushdown stacks s_i for $1 \leq i \leq r$. $[\varepsilon]$ denotes the empty level 1 stack, the empty level n stack, denoted by $[\varepsilon]^n$, is a stack which contains for $1 \leq i < n$ only a single empty level i stack.

The following *instructions* can be executed on a level 1 stack $[a_m, \ldots, a_1]$:

$$\begin{aligned} \mathrm{push}_1^a([a_m, \ldots, a_1]) &:= [a, a_m, \ldots, a_1] \text{ for every } a \in \Gamma \\ \mathrm{pop}_1([a_m, a_{m-1} \ldots, a_1]) &:= [a_{m-1}, \ldots, a_1] \end{aligned}$$

Furthermore we define the following function which does not change the content of a stack:

$$\mathrm{top}([\varepsilon]) := \varepsilon \text{ and } \mathrm{top}([a_m, \ldots, a_1]) := a_m \text{ for } m \geq 1.$$

For a stack $[s_r, \ldots, s_1]$ of level $n \geq 2$ we define the following instructions

$$\begin{aligned} \mathrm{push}_1^a([s_r, \ldots, s_1]) &:= [\mathrm{push}_1^a(s_r), s_{r-1}, \ldots s_1] \text{ for every } a \in \Gamma \\ \mathrm{push}_n([s_r, \ldots, s_1]) &:= [s_r, s_r, \ldots, s_1] \\ \mathrm{push}_k([s_r, \ldots, s_1]) &:= [\mathrm{push}_k(s_r), s_{r-1}, \ldots, s_1] \text{ for } 2 \leq k < n \\ \mathrm{pop}_n([s_r, \ldots, s_1]) &:= [s_{r-1}, \ldots, s_1] \\ \mathrm{pop}_k([s_r, \ldots, s_1]) &:= [\mathrm{pop}_k(s_r), s_{r-1}, \ldots, s_1] \text{ for } 1 \leq k < n \end{aligned}$$

and extend top to a level n stack by setting $\mathrm{top}([s_r,\ldots,s_1]) := \mathrm{top}(s_r)$.

We denote by Instr_n the set of instructions that can be applied to a level n stack (without the top function). For the sake of easiness we also add an identity function denoted by $-$ which does not change the stack at all.

The instruction push_1^a adds the symbol a to the topmost level 1 stack, while push_k duplicates the topmost level $k-1$ stack completely. Similarly pop_1 removes the top symbol of the topmost level 1 stack, while pop_k for $1 < k \leq n$ removes the corresponding level $k-1$ stack completely. Note that the instruction pop_k for $2 \leq k \leq n$ can only be applied if the resulting stack is again a level n stack, i.e. it does not remove a bottom level $k-1$ stack.

A *higher-order pushdown automaton of level* n is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \Delta)$ where Q is a finite set of states, Σ is an input alphabet, Γ is a stack alphabet, $q_0 \in Q$ is an initial state, and $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q \times \mathrm{Instr}_n$ is a transition relation. A *configuration* of $\mathcal{A}$ is a pair $(q, [s_r,\ldots,s_1])$ where q is a state of $\mathcal{A}$ and $[s_r,\ldots,s_1]$ is a stack of level n. The initial configuration $(q_0, [\varepsilon]^n)$ consists of the initial state q_0 and the empty level n stack $[\varepsilon]^n$. $\mathcal{A}$ can reach a configuration $(q', [s'_{r'},\ldots,s'_1])$ from $(q, [s_r,\ldots,s_1])$ by reading $a \in \Sigma \cup \{\varepsilon\}$ if there is a transition $(q, a, \mathrm{top}([s_r,\ldots,s_1]), q', i) \in \Delta$ such that $i([s_r,\ldots,s_1]) = [s'_{r'},\ldots,s'_1]$. The automaton $\mathcal{A}$ accepts a word $w \in \Sigma^*$ if $\mathcal{A}$ reaches from the initial configuration the empty level n stack after reading w. We denote by $\mathrm{HOPDA}(n)$ the class of all higher-order pushdown automata of level n.

3 Closure Properties

In this part, we prove that the hierarchy is closed under MSO-transductions and the treegraph operation. We first consider the case of deterministic trees.

3.1 The Deterministic Case

We consider a sub-hierarchy obtained by unfolding only *deterministic* graphs. $\mathrm{Graph_d}(0)$ is equal to $\mathrm{Graph}(0)$. $\mathrm{Tree_d}(n+1)$ contains the unfoldings of every deterministic graph $G \in \mathrm{Graph_d}(n)$ from a vertex in V^G. $\mathrm{Graph_d}(n)$ is defined in the same way as $\mathrm{Graph}(n)$. Note that $\mathrm{Graph_d}(n)$ also contains non-deterministic graphs.

Closure under MSO-transductions Using results from [3], we prove that for all $n \in \mathbb{N}$, $\mathrm{Graph_d}(n)$ is closed under MSO-transductions. This result was obtained for the first level by A. Blumensath in [1]. Obviously, $\mathrm{Tree_d}(n)$ is not closed under MSO-transductions but if we consider only MSO-markings, we obtain also a closure property for $\mathrm{Tree_d}(n)$.

Proposition 1. *For all* $n \geq 0$*, all tree* $T \in \mathrm{Tree_d}(n)$ *and all graph* $G \in \mathrm{Graph_d}(n)$*, we have that:*

1. $\mathcal{M}(T)$ *also belongs to* $\mathrm{Tree_d}(n)$*, for any MSO-marking* $\mathcal{M}$*,*

2. $\mathcal{T}(G)$ *also belongs to* $\mathrm{Graph_d}(n)$*, for any MSO-transduction* $\mathcal{T}$*.*

Proof (sketch): These results are proved by induction on the level using partial commutation results of MSO-transductions and unfolding obtained in [3].

1. For every deterministic graph G and every MSO-marking $\mathcal{M}$, there exists an MSO-transduction $\mathcal{T}'$ and a vertex r' such that $\mathcal{M}(\mathrm{Unf}(G,r)) \approx \mathrm{Unf}(\mathcal{T}'(G), r')$.
2. For every deterministic graph G and every MSO-transduction $\mathcal{T}$, there exists an MSO-transduction $\mathcal{T}'$, a rational mapping h and a vertex r' such that $\mathcal{T}(\mathrm{Unf}(G,r)) \approx h^{-1}(\mathrm{Unf}(\mathcal{T}'(G), r'))$.

Note that in both cases $\mathcal{T}'$ preserves determinism. $\square$

Closure Under the Treegraph Operation The unfolding is a particular case of the treegraph operation in the sense that for any graph G the unfolding from a definable vertex r, $\mathrm{Unf}(G,r)$, can be obtained by an MSO-interpretation from $\mathrm{Treegraph}(G,\sharp)$ (see [9]). In the case of deterministic trees, we show a converse result: how to obtain treegraph using MSO-interpretations and unfolding. This construction is due to T. Colcombet.

Lemma 1. *For any finite set of labels* Σ*, there exist two finite mappings* h_1*,*h_2 *and a rational marking* $\mathcal{M}$ *such that for any deterministic tree* T *with root* r*:*

$$\mathrm{Treegraph}(T,\sharp) \approx h_2^{-1}\left(\mathcal{M}\left(\mathrm{Unf}\left(h_1^{-1}(T), r\right)\right)\right).$$

Proof (sketch): The finite mapping h_1 adds backward edges labeled by elements of $\tilde{\Sigma}$ and a loop labeled by $\sharp$ to every vertex. Thus, for all $a \in \Sigma$, h_1 is defined by $h_1(a) = \{a\}$, $h_1(\tilde{a}) = \{\bar{a}\}$ and $h_1(\sharp) = \{\varepsilon\}$.

Let H be the deterministic tree equal to $\mathrm{Unf}\left(h_1^{-1}(T), r\right)$, every node x of H is uniquely characterized by a word in $(\Sigma \cup \tilde{\Sigma} \cup \{\sharp\})^*$. The rational marking $\mathcal{M}_\$$ marks all the vertices corresponding to a word which does not contain $x\tilde{x}$ or $\tilde{x}x$ for $x \in \Sigma$. Finally, h_2 is used to erase unmarked vertices and to reverse the remaining edges with labels in $\tilde{\Sigma}$. h_2 is given by $h_2(\sharp) = \{\sharp\}$ and $h_2(a) = \left\{\$\bar{\$}a\$\bar{\$}, \$\bar{\$}\bar{\tilde{a}}\$\bar{\$}\right\}$ for $a \in \Sigma$. $\square$

Figure 1 illustrates the construction above on the semi-infinite line. The filled dots represent the vertices marked by $\mathcal{M}_\$$. The closure of the deterministic hierarchy under the treegraph operation is obtained from Lem. 1 and Prop. 1, using the fact that for all trees T and all rational mappings h which do not contain $\sharp$, $\mathrm{Treegraph}(h^{-1}(T),\sharp) = h_\sharp^{-1}(\mathrm{Treegraph}(T,\sharp))$ where $h_\sharp$ designates the rational mapping that extends h with $h_\sharp(\sharp) = \{\sharp\}$.

Proposition 2. *For all* $n \geq 0$*, if* $G \in \mathrm{Graph}_d(n)$ *then* $\mathrm{Treegraph}(G,\sharp) \in \mathrm{Graph}_d(n+1)$*.*

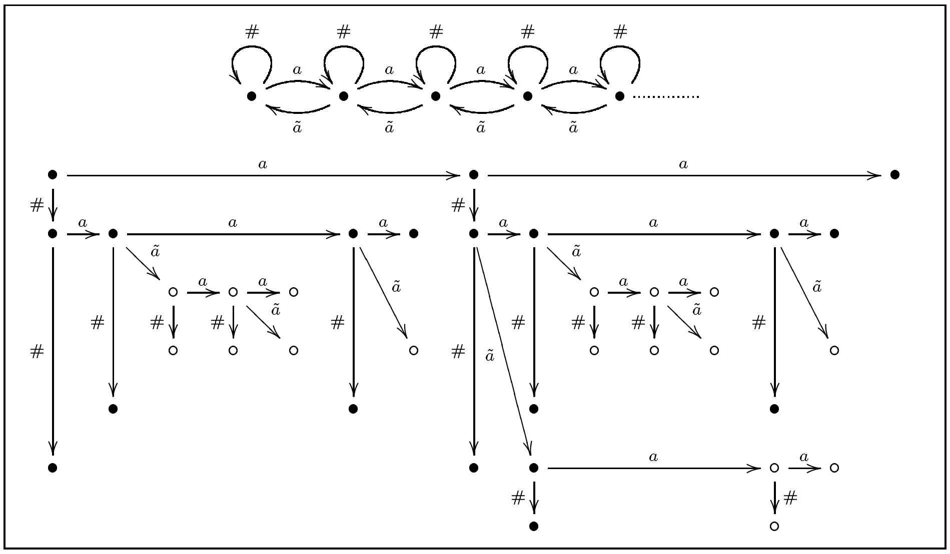

Fig. 1. The semi-infinite line after applying h_1 and its unfolding

3.2 Deterministic Trees are Enough

We now prove that for all n, $\mathrm{Graph}(n)$ is equal to $\mathrm{Graph_d}(n)$. From the technical point of view, this means that even if the hierarchy contains non-deterministic graphs and even graphs of infinite out-degree, we can always work with an "underlying" deterministic tree.

Lemma 2. *For all* $n > 0$, *if* $G \in \mathrm{Graph}(n)$, *then there exists a deterministic tree* $T \in \mathrm{Tree_d}(n)$ *and a rational mapping* h *such that* $G = h^{-1}(T)$.

Proof (sketch): The proof proceeds by induction on the level n. Let $T \in \mathrm{Tree}(n+1)$, we want to prove that T belongs to $\mathrm{Graph_d}(n+1)$. By definition of $\mathrm{Tree}(n+1)$ and by induction hypothesis, we have $T \approx \mathrm{Unf}(h^{-1}(T_d), s)$ for some deterministic tree $T_d \in \mathrm{Tree_d}(n)$ and some rational mapping h. Using the fact that the unfolding can be defined in the treegraph operation (see [9]), we have $T \approx \mathcal{T}(\mathrm{Treegraph}(h^{-1}(T_d), \sharp))$ for some MSO-transduction $\mathcal{T}$. If $h_\sharp$ denotes the rational mapping obtained by extending h with $h_\sharp(\sharp) = \{\sharp\}$, we have $T = \mathcal{T}(h_\sharp^{-1}(\mathrm{Treegraph}(T_d, \sharp))$. Applying Lem. 1, we have $T = \mathcal{T}'(\mathrm{Unf}(h_1^{-1}(T_d), r))$ where $\mathcal{T}' = \mathcal{M} \circ h_2^{-1} \circ h_{\#}^{-1} \circ \mathcal{T}$. It is easy to check that $\mathrm{Unf}\left(h_1^{-1}(T_d), r\right)$ belongs to $\mathrm{Tree_d}(n+1)$. Using Prop. 1, we prove that T belongs to $\mathrm{Graph_d}(n+1)$. The case of $G \in \mathrm{Graph}(n+1)$ is easily derived from this. $\square$

We can now prove that every graph of the hierarchy has a decidable MSO-theory. Note that this does not follow directly from the definition because unfolding from an arbitrary (i.e. not necessarily MSO-definable) vertex does not preserve the decidability of MSO-logic. However, using Lem. 2 we can always come back to the case where we unfold from a MSO-definable vertex (see [4] for more details).

Theorem 1. *Each graph of the hierarchy has decidable MSO-theory and this remains true if we add to MSO-logic the predicates* $|X| < \infty$, $|X| = k \mod p$ *for all* k *and* $p \in \mathbb{N}$ *which are interpreted as* X *is finite respectively* X *has size equal to* k *modulo* p *for* $k, p \in \mathbb{N}$.

Combining Prop. 1, Prop. 2 and Lem. 2, we now have two equivalent characterizations of the hierarchy: one "minimal" in terms of unfolding and inverse rational mappings and one "maximal" in terms of the treegraph operation and MSO-transductions. The maximal characterization shows the robustness of the hierarchy and its interest because it combines the two, to our knowledge, most powerful MSO-preserving operations. On the other side, the minimal characterization allows us to make the link between the hierarchy and the graphs of higher-order pushdown automata.

Theorem 2. *The Caucal hierarchy is equal to the hierarchy obtained by iterating the treegraph operation and MSO-transductions.*

4 Higher-order Pushdown Graphs vs. Caucal Graphs

In this section we give an automata-theoretic characterization of the classes of the Caucal hierarchy. This characterization provides us with a "flat" model for describing a graph of any level, i.e. we do not have to refer to a sequence of operations. Furthermore it extends the characterization of the first level of the hierarchy as the ε-closure of configuration graphs of pushdown automata given in [16] to any level. We recall some definitions.

The *configuration graph* $C(\mathcal{A})$ of $\mathcal{A} \in \mathrm{HOPDA}(n)$ is the graph of all configurations of $\mathcal{A}$ reachable from the initial configuration, with an edge labeled by $a \in \Sigma \cup \{\varepsilon\}$ from (q, s) to (q', s') iff there is a transition $(q, a, \mathrm{top}(s), q', i) \in \Delta$ such that $i(s) = s'$.

Let $C(\mathcal{A})$ be the configuration graph of $\mathcal{A} \in \mathrm{HOPDA}(n)$. We will assume for the remainder of the paper that for every pair (q, α) of state q and top stack symbol α only ε-transitions or only non-ε-transitions are possible. The ε-*closure* of $C(\mathcal{A})$ is the graph G obtained from $C(\mathcal{A})$ by removing all vertices with only outgoing ε-transitions and adding an a-labeled edge between v and w if there is an a-labeled path from v to w in $C(\mathcal{A})$.

A *higher-order pushdown graph* G *of level* n is the ε-closure of the configuration graph of some $\mathcal{A} \in \mathrm{HOPDA}(n)$. We call G the higher-order pushdown graph *generated* by $\mathcal{A}$ and denote by $\mathrm{HOPDG}(n)$ the class of all higher-order pushdown graphs of level n (up to isomorphism).

This notion of ε-closure was used in [16] to show that the class $\mathrm{HOPDG}(1)$ coincides with the class of prefix recognizable graphs, i.e. with the the graphs on the first level of the hierarchy. We extend this result to every level of the hierarchy.

The easier part of the equivalence is to show that every HOPDG of level n is a graph on the same level of the hierarchy. The main idea is to find a graph in $\mathrm{Graph}(n)$ such that every node of this graph can be identified with a configuration of a higher-order pushdown automaton, and to construct an inverse rational mapping which generates the edges of the configuration graph of the automaton. Such a construction is already contained in [2] in a slightly different setting. We propose here to use the family Δ_m^n of graphs obtained by an $(n-1)$-fold application of the treegraph operation to the infinite m-ary tree Δ_m. This has the advantage that there is almost a one-to-one correspondence between configurations of the higher-order pushdown automaton and the vertices of the graph. Using the fact that $\Delta_m^n \in \mathrm{Graph}(n)$ we obtain:

Lemma 3. *If $G \in \mathrm{HOPDG}(n)$ then $G \in \mathrm{Graph}(n)$.*

We now turn to the converse direction: every graph G on level n of the hierarchy is indeed the ε-closure of a configuration graph of a higher-order pushdown automaton of level n. We show this using the following two Lemmas.

Lemma 4. *If $G \in \mathrm{HOPDG}(n)$ and $r \in V^G$, then $\mathrm{Unf}(G, r) \in \mathrm{HOPDG}(n+1)$.*

Lemma 5. *If $G \in \mathrm{HOPDG}(n)$, $r \in V^G$ and h is a rational mapping, then $h^{-1}(\mathrm{Unf}(G, r)) \in \mathrm{HOPDG}(n+1)$.*

While the proof of Lem. 4 consists of a straightforward modification of the HOPDA for G, the proof of Lem. 5 requires some technical preparation. We need to show that for an automaton as constructed in the proof of Lem. 4 there exists a higher-order pushdown automaton which generates exactly the graph $\mathrm{Unf}(G, r)$ extended by reverse edges, i.e. for all $v, w \in \mathrm{Unf}(G, r)$, $v \xrightarrow{a} w$ in $\mathrm{Unf}(G, r)$ iff $w \xrightarrow{\bar{a}} v$ in the extended graph.

To show that such an automaton exists we introduce the notion of a *weak popping higher-order pushdown automaton.* A weak popping automaton is only allowed to execute a pop instruction of level $j \geq 2$ if the two top level j stacks coincide. We skip a formal definition of a weak popping higher-order pushdown automaton and just mention that even though this automaton model is equipped with a built-in test on the equality of two stacks of the same level, it is equivalent to the usual model. All proofs are given in the full version of this article [4].

Theorem 3. *For every $n \in \mathbb{N}$, $\quad G \in \mathrm{HOPDG}(n)$ iff $G \in \mathrm{Graph}(n)$.*

5 More Properties of the Caucal Hierarchy

We give a generator for each level of the hierarchy. Then we use the traces of the graphs of the hierarchy to prove its strictness and to exhibit a graph having a decidable MSO-theory which is not in the hierarchy.

5.1 Generators

For the first level of the hierarchy, the infinite binary tree Δ_2 is a generator for rational markings (without backward edges) from the root and inverse rational mappings. As hinted by the proof of Lem. 3, a similar result can be obtained at any level. Recall that Δ_2^n is the graph obtained from Δ_2 by an $(n-1)$-fold application of the treegraph operation.

Proposition 3. *Every graph $G \in \text{Graph}(n)$ can be obtained from Δ_2^n by applying a rational marking (with backward edges) from its source and an inverse rational mapping.*

5.2 On Traces — The Strictness of the Hierarchy

A direct consequence of Theo. 3 is that the traces of the graphs of level n are recognized by a higher-order pushdown automaton of level n. These families of languages have been studied by W. Damm and form the OI-hierarchy [11]. The equivalence between the OI-hierarchy and the traces of higher-order pushdown automata is proved in [12]. In [10, 14], this hierarchy is proved to be strict at each level.

Theorem 4. *For all $n \geq 1$,*

(a) for all $T \in \text{Tree}(n)$ the branch language of T (i.e. the set of all words labeling a path from the root to a leaf) is recognized by a HOPDA of level $n-1$.
(b) for all $G \in \text{Graph}(n)$ and $u, v \in V_G$, $\mathcal{L}(u,v,G) = \{w \in \Gamma^* \mid u \xrightarrow{w} v\}$ is recognized by a HOPDA of level n.

According to Theo. 4, the strictness level-by-level of the OI-hierarchy implies the strictness of the Caucal hierarchy. An obvious example of a graph which is at level n but not at level $n-1$ is the generator Δ_2^n. To obtain more natural graphs that separate the hierarchy, we consider the trees associated to monotonically increasing mappings $f : \mathbb{N} \to \mathbb{N}$. The tree $\mathcal{T}_f$ associated to f is defined by the following set of edges: $E_a = \{((i,0),(i+1,0)) \mid i \in \mathbb{N}\}$ and $E_b = \{((i,j),(i,j+1)) \mid i \in \mathbb{N} \text{ and } j+1 \leq f(i)\}$. The branch language of $\mathcal{T}_f$ is $\{a^n b^{f(n)} \mid n \geq 0\}$.

Using a property of rational indexes of k-OI languages (see [10]), we obtain the following proposition.

Proposition 4. *If $\{a^n b^{f(n)} | n \in \mathbb{N}\}$ is recognized by a higher-order pushdown automaton of level k then $f \in \mathcal{O}(2{\uparrow}^{k-1}(p(n)))$ for some polynomial p where $2{\uparrow}^0(n) = n$ and $2\uparrow^{k+1}(n) = 2^{2\uparrow^k(n)}$.*

Let us consider the mapping $exp_k(n) = 2{\uparrow}^k(n)$. It has been proved in [7] that $\mathcal{T}_{exp_k}$ belongs to Graph$(k+1)$. Note that using Theo. 3 the construction given in [7] can be avoided by providing a deterministic higher-order pushdown automaton of level $k+1$ that recognizes $\{a^n b^{exp_k(n)} | n \in \mathbb{N}\}$.

It is natural to consider the "diagonal" mapping $exp_\omega(n) = exp_n(1)$. Figure 2 shows an initial segment of the tree associated to exp_ω. By Prop. 4, the associated tree $\mathcal{T}_{exp_\omega}$ is not in the hierarchy. However, using techniques from [5], we can prove that $\mathcal{T}_{exp_\omega}$ has a decidable MSO-theory.

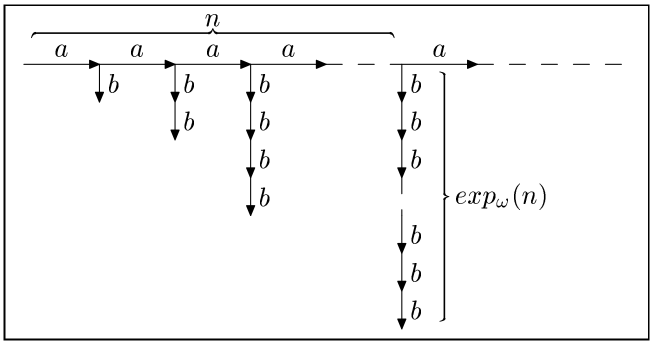

Fig. 2. The graph $\mathcal{T}_{exp_\omega}$ of the function exp_ω

Proposition 5. *There exists a graph with a decidable MSO-theory which is not in the Caucal hierarchy.*

6 Conclusion

We have given two characterizations of the Caucal hierarchy. We have shown that it coincides with the hierarchy obtained by alternating the treegraph operation and MSO-transductions, and thus have partly answered a question posed in [17]. It remains open whether one can extend this result to structures other than graphs, i.e. with symbols of higher arity.

We have also characterized the Caucal hierarchy as the ε-closure of configuration graphs of higher-order pushdown automata and have used this result to obtain that the hierarchy is indeed strict, but does not contain all graphs with a decidable MSO-theory.

Despite these characterization results we know surprisingly few about the graphs obtained on level $n \geq 2$. This deserves further study. Also a thorough comparison with other methods to generate infinite graphs with a decidable theory misses (see [17] for a more precise account on this).

Futhermore we like to mention that neither the constructions used to build the hierarchy nor Proposition 5 contradicts Seese's conjecture that every infinite graph (or every set of finite graphs) having a decidable MSO-theory is the image of a tree (or a set of trees) under an MSO-transduction.

Finally, many of the questions posed in [7] on the corresponding hierarchy of trees remained unsolved so far.

Acknowledgment

We thank D. Caucal and W. Thomas for stimulating discussions while working on this paper and for financial support enabling our visits in Aachen respectively Rennes. We also thank T. Colcombet for pointing to a solution for Lem. 1, T.

Cachat for reading a first draft of Sect. 4, and the anonymous referees for many useful remarks.

References

1. A. Blumensath. Prefix-recognisable graphs and monadic second-order logic. Technical Report AIB-2001-06, RWTH Aachen, 2001.
2. T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming*, volume 2719 of *LNCS*, pages 556–569. Springer, 2003.
3. A. Carayol and T. Colcombet. On equivalent representations of infinite structures. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming*, volume 2719 of *LNCS*, pages 599–610. Springer, 2003.
4. A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. Technical report, RWTH Aachen, 2003.
5. O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *LNCS*, pages 275–284. Springer, 2000.
6. D. Caucal. On infinite transition graphs having a decidable monadic theory. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, volume 1099 of *LNCS*, pages 194–205, 1996.
7. D. Caucal. On infinite terms having a decidable monadic theory. In *Proceedings of the 27th International Symnposium on Mathematical Foundations of Computer Science*, volume 2420 of *LNCS*, pages 165–176. Springer, 2002.
8. B. Courcelle. Monadic second-order definable graph transductions: A survey. *Theoretical Computer Science*, 126:53–75, 1994.
9. B. Courcelle and I. Walukiewicz. Monadic second-order logic, graph coverings and unfoldings of transition systems. *Annals of Pure and Applied Logic*, 92:35–62, 1998.
10. W. Damm. An algebraic extension of the Chomsky-hierarchy. In *Proceedings of the 8th International Symposium on Mathematical Foundations of Computer Science*, volume 74 of *LNCS*, pages 266–276. Springer, 1979.
11. W. Damm. The IO and OI hierarchies. *Theoretical Computer Science*, 20:95–208, 1982.
12. W. Damm and A. Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control*, 71:1–32, 1986.
13. H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
14. J. Engelfriet. Iterated stack automata and complexity classes. *Information and Computation*, 95:21–75, 1991.
15. T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pusdown trees are easy. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, volume 2303 of *LNCS*, pages 205–222. Springer, 2002.
16. C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Submitted.
17. W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science*, volume 2747 of *LNCS*. Springer, 2003.
18. I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science*, 275:311–346, 2002.