The Structure of Trees in the Pushdown Hierarchy

Arnaud Carayol
Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France

Lucien Charamond
Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France

Abstract

In this article, we investigate the structure of the trees in the pushdown hierarchy, a hierarchy of infinite graphs having a decidable MSO-theory. We show that a binary complete tree in the pushdown hierarchy must contain at least two different subtrees which are isomorphic. We extend this property to any tree with no leaves and with chains of unary vertices of bounded length. We provided two applications of this result. A first application in formal language theory, gives a simple argument to show that some languages are not deterministic higher-order indexed languages. A second application in number theory shows that the real numbers defined by deterministic higher-order pushdown automata are either rational or transcendental.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Verification by model checking

Keywords and phrases Pushdown hierarchy, Monadic second-order logic, Automatic numbers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.131

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Acknowledgements The authors are indebted to Didier Caucal for pointing out the notion of (ℓ, b)-tree implicit in the work [3] and starting this work. The authors would like to thanks the reviewers for their work.

1 Introduction

The pushdown hierarchy (also called the Caucal hierarchy) is a robust hierarchy of infinite directed graphs for which monadic second-order logic (MSO) is decidable. These graphs have a countable set of vertices but their edges and nodes are labeled and colored by finite sets. Such infinite graphs with decidable MSO-theories play an important role in automated program verification as they provide a framework in which the model-checking problem for many relevant properties such as termination and safety is decidable. The robustness of the pushdown hierarchy is witnessed by its numerous characterizations and closure properties (we refer the reader to [21] for a survey).

A first characterization of the pushdown hierarchy is via graph transformations following the original idea of Caucal [10]. Every graph in the pushdown hierarchy can be constructed starting from a finite tree by combining two graph transformations that preserve the decidability of MSO-theories namely MSO-interpretations [12] and graph unfolding [13]. As shown in Figure 1, the pushdown hierarchy consists of two intertwined hierarchies: one of classes of trees $(\mathrm{Tree}_n)_{n\geqslant 0}$ and one of classes of graphs $(\mathrm{Graph}_n)_{n\geqslant 0}$. The class Tree_0 contains all finite trees and for $n \geqslant 0$, Graph_n contains all graphs that can be MSO-interpreted in a tree of Tree_n. The trees in Tree_{n+1} are the unfoldings of the graphs in Graph_n. In particular, Graph_0 contains all finite graphs, Tree_1 contains the regular trees and the graphs in Graph_1 are the prefix-recognizable graphs [9]. This hierarchy is closed under most if not all transformations known to preserve the decidability of MSO-theories [8]. It is in particular closed under MSO-transductions [13] and the Muchnik's iteration [22]. More recently, the pushdown hierarchy was shown to be closed under MSO+$\mathrm{U}^{\mathrm{fin}}$-interpretations in [20].

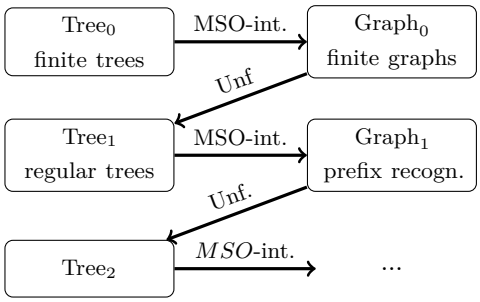

Figure 1 The construction of the pushdown hierarchy using MSO-interpretations and the unfolding operation.

The graphs and trees in the pushdown hierarchy admit several alternative characterizations. The graphs in Graph_n are (up-to isomorphism) the transition graphs of higher-order pushdown automata of order n (see [8]). Higher-order pushdown automata [16] are a generalization of the standard model of pushdown automata which manipulates stacks of stacks at order 2, stacks of stacks of stacks at order 3, ... The deterministic terms in Tree_{n+1} are the solutions of higher-order safe recursion schemes of order-n [10, 15].

These different characterizations make it easy to show that a graph belongs to the pushdown hierarchy. It is rather more complicated to show that a graph does not belong to the pushdown hierarchy. Still the question of characterizing the graphs in this hierarchy has received a lot of attention. There are two main approaches. One approach is to work with higher-order pushdown automata and to develop pumping lemmas for these automata [5, 6, 18, 19]. A second approach is to focus on structural properties of the graphs and to work by induction of the level in the hierarchy using graph transformations [1, 7]. The most involded separation result, namely the separation between trees produced by safe and unsafe recursion schemes was obtained with the first approach [19]. However this approach is much more technically involved and arguably the technical results obtained are less likely to be reusable. In this work, we follow the second approach.

Our starting point is a question asked to the first author by Wolfgang Thomas. He asked whether the pushdown hierarchy contains irrational algebraic numbers such as $\sqrt{2}$. Meaning, does there exists an infinite word (i.e., a unary tree) in the pushdown hierarchy that encodes the expansion of $\sqrt{2}$ in some base $\ell \geqslant 2$? Sadly with this precise statement, the question seems still far out of reach[1]. But the recent work of [3] shows that if we choose to represent a real number in $[0, 1]$ not by its expansion but by a particular tree encoding this expansion, the trees associated with irrational algebraic numbers have strong structural properties: they are deterministic, mostly-complete and all their subtrees are non-isomorphic. The trees representing expansions of real numbers in $[0, 1]$ are implicit in the work of [3] and generalize the definition of automatic real numbers [4]. Indeed they coincide with a generalization of automatic real numbers in which deterministic finite automata are replaced by deterministic higher-order pushdown automata [11].

[1] In this sense, the pushdown hierarchy is known to contain all the rational numbers and all the morphic numbers (and hence all the automatic numbers) in $[0, 1]$. Indeed morphic sequences have been shown to belong to Tree_2 in [10, Proposition 3.2]. Hence the pushdown hierarchy contains expansions of transcendental numbers (see [2]) but is not known to contain the expansions of any irrational algebraic number.

In Section 4 and Section 5, we show that any tree with no leaves and with chains of unary vertices of bounded length in the pushdown hierarchy must contain two different isomorphic subtrees. The main technical ingredient is a precise description of the MSO-interpretations constructing a complete binary tree from a complete binary tree. In Section 6, we give two applications of this result. First, we show how it can be used to show that certain languages such as the language $L_{ww} = \{ww \mid w \in \{0,1\}^*\}$ cannot be accepted by any deterministic higher-order pushdown automaton. Second, we show that a real number in $[0,1]$ represented by a tree in the pushdown hierarchy is either rational or transcendental (i.e., not algebraic). As the real numbers with a tree in Tree_1 are the automatic reals numbers, the result was proved for level 1 in [2]. It was also shown for level 2 in [3].

2 Preliminaries

Notations. Let Σ^* denote the set of all words over the alphabet Σ. We write $u \sqsubseteq v$ if u is a prefix of v and $u \sqsubsetneq v$ if u is a strict-prefix of v. If Σ is equipped with a total order relation, we denote by $<_{\mathrm{lex}}$ the resulting lexicographic ordering on words over Σ.

Infinite graphs and trees. In this article, we consider graphs with countably many vertices with labeled edges and colored vertices. Let Σ be a finite set of edge labels and Θ be a finite set of vertex colors, a graph G labeled by Σ and colored by Θ is a tuple (V, E, C) where V is a countable set of vertices, $E \subseteq V \times \Sigma \times V$ is a set of labeled edges, $C \subseteq \Theta \times V$ is the set of colors. A graph G is deterministic if for all $\sigma \in \Sigma$ and all vertices u, v and v', $u \xrightarrow[G]{\sigma} v$ and $u \xrightarrow[G]{\sigma} v'$ then $v = v'$.

A path π in a graph G from u to v is a sequence $u_0\sigma_0u_1 \cdots \sigma_{n-1}u_n \in V(\Sigma V)^*$ such that $u_0 = u$, $u_n = v$ and $(u_i, \sigma_i, u_{i+1}) \in E$ for $i \in [0, n-1]$. This path is labeled by the word $w = \sigma_0 \cdots \sigma_{n-1}$. We write $u \xrightarrow[G]{w} v$ (or simply $u \xrightarrow{w} v$ if G is clear from the context) to denote the existence of such a path. We extend this notation to a language L over Σ by taking $u \xrightarrow{L} v$ if and only if $u \xrightarrow{w} v$ for some $w \in L$. To improve readability, we write $\longrightarrow$ instead of $\xrightarrow{\Sigma}$, $\longrightarrow^*$ instead of $\xrightarrow{\Sigma^*}$ and $\longrightarrow^+$ instead of $\xrightarrow{\Sigma^+}$.

For two graphs G_1 and G_2, we write $G_1 \sim G_2$ to denote the existence of an isomorphism between G_1 and G_2.

A graph T is a tree if there exists a vertex r called the *root* of T such that there exists a unique path from the root to any vertex. Vertices in a tree are called nodes. A node v is a child of a node u if $u \xrightarrow{\Sigma} v$. In this case, we say that u is the parent of v. A node v is a descendant of u if $u \longrightarrow^* v$ which we also denote by $u \sqsubseteq_T v$. The subtree of a tree T rooted at a node u, denoted by $T_{|u}$, is the tree obtained by restricting T to u and its descendants.

Every node in a deterministic tree is uniquely identified by the label of the unique path from the root to this vertex. As a result, a deterministic tree T labeled by Σ and colored by Θ is determined up-to isomorphism by a mapping from a prefix-closed subset of Σ^* to the subsets of Θ. When reasoning up to isomorphism, we will not distinguish between a deterministic tree and the associated mapping. We always assume that Σ comes with a fixed arbitrary order and hence that the nodes of a deterministic tree can be compared using the lexicographic order.

A complete binary tree is a deterministic tree labeled by $\{0,1\}$ in which every node has two children. The 0-child is called the left-child and the 1-child is called the right-child. For a direction $\gamma \in \{\uparrow, \swarrow, \searrow\}$, we say that v is the γ-successor of u, if v is the parent of u and

$\gamma = \uparrow$ or if v is the left-child (resp. right-child) of u and $\gamma = \swarrow$ (resp. $\gamma = \searrow$). We say that v is in direction γ relative to u, if the γ-successor of u is on the minimal path (ignoring the orientations of the edges) from u to v.

Monadic-second order logic on graphs. We define *monadic-second order logic* (MSO) over graphs with labeled edges and colored nodes as usual. We use lowercase letters $x, y, z, \ldots$ for first order variables and uppercase letters $X, Y, Z, \ldots$ for second order variables. The atomic formulas are $x = y$, $x \in X$, $x \xrightarrow{\sigma} y$ and $\theta(x)$ for σ an edge label and θ a color. MSO-formulas are obtained by applying boolean operators ($\neg$ and $\vee$) and existential quantifiers ($\exists$) over both first and second order variables. To improve readability, we will freely use any definable notion as syntactic sugar: $\forall$, $\Rightarrow$, $X \subseteq Y$, ...

The notion of free variables is defined as usual. We write $\varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ when the free variables of a formula φ are among $x_1, \ldots, x_n, X_1, \ldots, X_m$. A closed formula does not have any free variables. For a graph G and a formula $\varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$, we write $G \models \varphi[u_1, \ldots, u_n, U_1, \ldots, U_m]$ when the graph satisfies the formula if the free variables $x_1, \ldots, x_n, X_1, \ldots, X_m$ are respectively interpreted as $u_1, \ldots, u_n, U_1, \ldots, U_m$ where the u_i's are vertices and U_i's are sets of vertices. The MSO-theory of a graph G is a set of closed formulas satisfied by G. A vertex u of a graph G is MSO-definable in G if there exists a formula $\varphi(x)$ such that u is the only vertex such that $G \models \varphi[u]$. The notion of an MSO-definable set of vertices is defined similarly.

Graph transformations. An *MSO-interpretation* $\mathcal{I}$ (on graphs) is given by a tuple of MSO-formulas $(\delta(x), (\varphi_\sigma(x, y))_{\sigma \in \Sigma}, (\varphi_\theta(x))_{\theta \in \Theta})$ where Σ and Θ are finite sets of labels and colors respectively. An *MSO-recoloring* is a special case of MSO-interpretation which does not erase any vertices (i.e., $\delta(x) = \mathbf{true}$) and preserves all edges (i.e., $\varphi_\sigma(x, y) = x \xrightarrow{\sigma} y$ for $\sigma \in \Sigma$).

An *MSO-transduction* (see [12]) is the composition of a K-copying operation followed by an MSO-interpretation. For a finite set of labels $K = \{k_1, \ldots, k_n\}$, the K-copying operation adds for every vertex u of the graph, fresh vertices $u_1, \ldots, u_n$ as well as edges from u to u_i labeled by k_i for each $i \in [1, n]$.

The *unfolding* of a graph G from a vertex s is the tree denoted by $\mathrm{Unf}(G, s)$ whose vertices consists of all paths in G starting from s and with an edge labeled by a from a path π to a path π' if $\pi' = \pi a t$ for some vertex t. Furthermore a path π, ending at a vertex t of G, is colored in $\mathrm{Unf}(G, s)$ with the same colors as t in G.

3 The pushdown hierarchy

The pushdown hierarchy contains the (possibly infinite) graphs which can be constructed using MSO-interpretations combined with the unfolding operation starting from a finite tree. The pushdown hierarchy consists of two intertwined hierarchies of classes of graphs[2]: one containing trees $(\mathrm{Tree}_n)_{n \in \mathbb{N}}$ and one containing graphs $(\mathrm{Graph}_n)_{n \in \mathbb{N}}$ such that:

- Tree_0 is the class of all finite trees;
- for $n \geqslant 0$, Graph_n is the class of all graphs G such that there exists an MSO-interpretation $\mathcal{I}$ and a tree $T \in Tree_n$ with $G \sim \mathcal{I}(T)$;
- for $n \geqslant 1$, Tree_n is the class of trees such that there exists a graph $G \in \mathrm{Graph}_{n-1}$ and a vertex $u \in G$ such that $T \sim \mathrm{Unf}(G, u)$.

[2] All the graphs we consider are labeled and colored by finite sets: only the set of vertices is infinite.

As both MSO-interpretations and the unfolding (from an MSO-definable vertex) preserve the decidability of MSO-theories [12, 13], it follows that all graphs in the pushdown hierarchy have a decidable MSO-theory [8]. Our main contribution only uses the following closure properties which follow from [8]:

▶ **Theorem 1** ([8]). *The following properties hold:*

1. *For $n \geqslant 0$, for a deterministic tree T in Tree_n and an MSO-recoloring μ, the tree $\mu(T)$ belongs to Tree_n.*
2. *For $n \geqslant 0$, the class Graph_n is closed under MSO-transductions and under restriction to reachable vertices from a given vertex (not necessarily MSO-definable) and Tree_n is closed under taking subtrees.*
3. *For $n \geqslant 1$, every graph $G \in \mathrm{Graph}_n$ is isomorphic to $\mathcal{I}(T)$ for some complete binary tree $T \in \mathrm{Tree}_n$ and some MSO-interpretation $\mathcal{I}$. Furthermore, we can assume that for all nodes $s \neq s'$ of $\mathcal{I}(T)$, s and s' are incomparable for $\sqsubseteq_T$.*

3.1 MSO-interpretations and tree-walking automata

By Property 1 of Theorem 1, every graph in Graph_n can be MSO-interpreted in a complete binary tree. In [7, 8], it is shown that MSO-interpretations applied to deterministic trees can be described using only MSO-recolorings and tree walking automata. To simplify the presentation, we tailor our definitions to complete binary trees (cf. Remark 4).

A *tree-walking automaton* on complete binary trees colored by Θ is a tuple $A = (Q, q_A, F, \Delta)$ where Q is the finite set of states, $q_A \in Q$ is the initial state, $F \subseteq Q \times 2^\Theta$ is the set of accepting conditions and $\Delta \subseteq Q \times 2^\Theta \times \{\uparrow, \swarrow, \searrow\} \times Q$ is the set of transitions. Intuitively, a transition (p, c, γ, q) expresses that if the automaton is in state p on a node u colored by the colors of c, it can move in state q to the γ-successor of u.

A run of a tree-walking automaton A on a complete binary tree T starting from a node u_0 in state q is a finite sequence $q_0 u_0 q_1 u_1 \cdots q_n u_n \in (QV_T)^+$ with $q_0 = q$ and for $i \in [0, n-1]$, there exists a transition $\delta = (q_i, T(u_i), \gamma, q_{i+1}) \in \Delta$ with u_{i+1} the γ-successor of u_i. A run is accepting if $(q_n, T(u_n))$ belongs to F. A tree-walking automaton A *accepts a pair of nodes* (u, v) if there exists an accepting run of A on T from the initial state q_A starting at u and ending in v.

On complete binary trees, tree-walking automata can accept any MSO-definable binary relation provided that the trees are recolored with a suitable MSO-recoloring.

▶ **Proposition 2** ([7, 8]). *For every binary complete tree T and every MSO-formula $\varphi(x, y)$, there exists an MSO-recoloring μ and a tree-walking automaton A_φ which accepts on $\mu(T)$ the pairs of nodes (u, v) such that $T \models \varphi[u, v]$.*

In this article, we need a stronger result in the case when the MSO-formula $\varphi(x, y)$ defines a functional relation (i.e., for each node u, there exists at most one node v such that $T \models \varphi[u, v]$). Under this restriction, we will show that the tree-walking automaton can be chosen to be *deterministic* and *non-backtracking* on T.

A tree-walking automaton is said to be *deterministic* if for all state q and set of colors $c \in 2^\Theta$, $(q, c, \gamma, p) \in \Delta$ and $(q, c, \gamma', p') \in \Delta$ implies that $\gamma = \gamma'$ and $p = p'$. This notion of determinism guarantees that there is at most one run starting from a given node in a given state but it does not forbid the tree-walking automaton from visiting the same node twice. A tree-walking automaton is said to be *non-backtracking* on T if none of its runs on T visits the same node twice. In particular a non-backtracking automaton, when going from u to v, will always follow the shortest path from u to v (ignoring the orientations of the edges).

▶ **Proposition 3.** *For every complete binary tree T and every MSO-formula $\varphi(x, y)$ functional on T, there exist an MSO-recoloring μ and a deterministic and non-backtracking tree-walking automaton A_φ which accepts on $\mu(T)$ the pairs of nodes (u, v) such that $T \models \varphi[u, v]$.*

Proof Sketch. Let A be the non-deterministic tree-walking automaton and μ_A be the MSO-recoloring obtained for $\varphi(x, y)$ in Proposition 2. Our aim is to define a new coloring μ_B that captures the functional behavior of A on T. For this we define, for every node u and every state p of A, $\text{target}(u, p)$ to be the unique node v such that all accepting runs of A on $\mu_A(T)$ starting in state p at u end in v. If no such node v exists, $\text{target}(u, p)$ is undefined. As A accepts a functional relation, if A accepts a pair (u, v) then for the initial state q_A of A, $\text{target}(q_A, u)$ is defined and equal to v.

To define the coloring μ_B, we fix an arbitrary order on the states of A. For each state p of A, we color a node u by the tuple (p, γ, q) if $\text{target}(u, p)$ is defined and equal to v, γ is the direction in $\{\uparrow, \swarrow, \searrow\}$ of v relative to u and q is the smallest state of A such that $\text{target}(u_\gamma, q) = v$ with u_γ the γ-successor of u. Such a state q must exists, as every run from u to v must go through u_γ. This coloring can be defined in MSO as μ_A is MSO-definable.

The deterministic and non-backtracking automaton B has the same states, initial state and acceptance conditions as A. In a state p at a node u colored with a tuple (p, γ, q), the automaton moves in the direction γ to the state q. It is easy to show that B accepts (u, v) on $\mu_B(T)$ if and only if A accepts (u, v) on $\mu_A(T)$ which concludes the proof. ◀

▶ Remark 4. To ease the presentation, we only defined tree-walking automata on binary complete trees but they can be defined to work on general deterministic trees and the results of both Proposition 2 and Proposition 3 generalize to this setting.

As a spin-off result, we obtain a simple proof to an open question of [10, Question b] which asks (when reformulated in the setting of this article) if all deterministic trees in Tree_n can be obtained by replacing general MSO-interpretations by a restricted sub-class called deterministic rational inverse mapping. A *deterministic rational inverse mapping* is an MSO-interpretation in which edges are defined by deterministic tree-walking automata working on deterministic trees (cf. Remark 4) and the colors are obtained by renaming or erasing the existing colors. Only vertices that are source or target of an edge are kept (i.e. $\delta(x) := \exists y, \bigvee_{\sigma \in \Sigma} \varphi_\sigma(x, y) \vee \varphi_\sigma(y, x)$).

By a direct induction on the level of the pushdown hierarchy and using Proposition 3, we obtain the following proposition.

▶ **Proposition 5.** *Every deterministic tree in Tree_n is obtained by a n-fold application of a deterministic inverse rational mapping followed by an unfolding starting with a finite tree.*

4 Trees with no self-similarities in the pushdown hierarchy

A tree is said to have a self-similarity if it contains two different subtrees which are isomorphic. The main contribution of this article is the following theorem.

▶ **Theorem 6.** *Every complete binary tree in the pushdown hierarchy has self-similarities.*

Trees with no self-similarities are called *pure* in this article. To prove Theorem 6, we need to show that the pushdown hierarchy does not contain any pure binary complete tree. Assume toward a contradiction that it does. Let $n_0 \geqslant 0$ denote the smallest level such that either Graph_{n_0} or Tree_{n_0} contain such a tree.

Consider the case where Tree_{n_0} contains some pure complete binary tree T_{pure}. As T_{pure} is infinite, we must have $n_0 \geqslant 1$ and by definition of Tree_{n_0}, the tree T_{pure} is (up-to isomorphism) the unfolding of a graph $G \in \mathrm{Graph}_{n_0-1}$ from some vertex r_G. By Property 2 of Theorem 1, we can assume w.l.o.g. that all the vertices of G are reachable from r_G. As all subtrees of the unfolding of G from r_G are non-isomorphic, every vertex of G must be reachable by exactly one path from r_G. Indeed, if a node s was reachable by two different paths $\pi_1 \neq \pi_2$ from r_G, then the subtrees rooted at π_1 and at π_2 in $\mathrm{Unf}(G, r_G)$ would be isomorphic. This implies that G is a tree. As every tree is isomorphic to its unfolding from its root, the graph G must be isomorphic to T_{pure}. Hence $G \sim T_{\mathrm{pure}}$ is a pure tree in Graph_{n_0-1} which contradicts the definition of n_0.

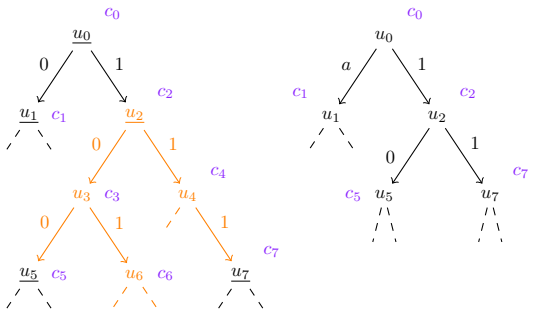

Figure 2 A tree T with a distinguished set of nodes $U = \{u_0, u_1, u_2, u_5, u_7\}$ (on the left) and the induced tree T_U (on the right). The chunk C_{u_2} is highlighted in orange.

It only remains to consider the case where Graph_{n_0} contains a pure binary tree T_{pure} but Tree_{n_0} does not contain any pure complete binary trees. By definition of Graph_{n_0} and by Property 3 of Theorem 1, there must exist a complete binary tree T and an MSO-interpretation $\mathcal{I}$ such that T_{pure} is isomorphic to $\mathcal{I}(T)$. We will show that if an MSO-interpretation is able to produce a pure complete binary tree when applied to some complete binary tree T, then the tree T must be a pure tree *in disguise*.

To formalize what we mean, we recall the notion of embedding in a tree which is illustrated in Figure 2. A set of nodes U in a tree T which contains a unique minimal element induces a tree denoted by T_U. Intuitively this tree is obtained by restricting the tree T to the vertices in U while preserving their colors and inheriting the ancestor relation from T. The label of an edge (u, v) in T_U is the label of the first edge in the unique path from u to v in T. Remark that we present the notion in its most general form but we will mainly work with embedding defining binary complete trees.

▶ **Definition 7.** *An* embedding *in a tree T labeled by Σ is a set of nodes U which contains a unique minimal element for the ancestor relation $\sqsubseteq_T$. This element is called the root of the embedding. This embedding* induces *the tree T_U whose nodes are the elements of U and such that $u \xrightarrow{x} v \in T_U$ if and only if $u \xrightarrow{xw} v \in T$ for some $w \in \Sigma^*$ and there are no $v' \in U$ such that $u \sqsubsetneq v' \sqsubsetneq v$. Moreover the nodes of U have the same colors in T and T_U.*

In Proposition 8, we show that, after applying a suitable MSO-recoloring to the tree T, the resulting tree embeds a pure complete binary tree. This is the main technical contribution of this paper.

▶ **Proposition 8.** *Let T_{pure} and T be two complete binary trees and let $\mathcal{I}$ be an MSO-interpretation such that $T = \mathcal{I}(T)$. If T_{pure} is pure then there exists an MSO-recoloring μ and an embedding $\mathcal{S}_{\mathcal{I}}$ MSO-definable in $\mu(T)$ which induces a pure complete binary tree.*

We defer the proof of this proposition to Section 5, to first show how it can be used to conclude the proof of Theorem 6. By applying Proposition 8, we obtain an MSO-recoloring μ and an embedding $\mathcal{S}_\mathcal{I}$ MSO-definable in $\mu(T)$ which induces a pure binary complete tree. The tree $\mu(T)$ belongs to Tree_{n_0} by Property 1 of Theorem 1. To reach a contradiction, we will show that the tree induced by $\mathcal{S}_\mathcal{I}$ in $\mu(T)$ belongs to Tree_{n_0}.

▶ **Proposition 9.** *Let $n \geqslant 0$, let T be a deterministic tree in Tree_n and let U be an MSO-definable embedding in T inducing a deterministic tree T_U. The tree T_U belongs to Tree_n.*

Proof Sketch. For $n = 0$, the result is immediate. Hence we assume that $n \geqslant 1$ and furthermore using the closure properties of Theorem 1, we can assume that the root of the embedding is the root of the tree T. Thanks to Property 1 of Theorem 1, we can color the nodes of U in T with a fresh color \$ to obtain a tree $T_\$$ also in Tree_n. Consider the following MSO-interpretation $\mathcal{I}$ that produces T_U from $T_\$$: it only keeps the vertices colored by \$ and for $x \in \Sigma$, it defines an x-labeled edge between two such vertices u and v if and only if v can be reached from u by a path labeled by a word in $x\Sigma^*$ which does not visit any vertices colored by \$ (except u or v). Furthermore, $\mathcal{I}$ erases the color \$ and preserves all other colors. Clearly, the tree induced by U on T is isomorphic to $\mathcal{I}(T_\$)$. By definition of Tree_n, the tree $T_\$$ is the unfolding of some graph $G \in \mathrm{Graph}_{n-1}$ from some vertex r. The key ingredient is that because $\mathcal{I}(T_\$)$ is deterministic, the MSO-interpretation $\mathcal{I}$ commutes with the unfolding. It follows that $T_U \sim \mathcal{I}(T_\$) \sim \mathrm{Unf}(\mathcal{I}(G), r)$. Hence T_U is isomorphic to the unfolding of the graph $\mathcal{I}(G)$ in Graph_{n-1} and belongs to Tree_n. ◀

By Proposition 9, the tree induced by $\mathcal{S}_\mathcal{I}$ in $\mu(T)$ belongs to Tree_{n_0} and by Proposition 8 it is pure which brings the contradiction and conclude the proof of Theorem 6.

Clearly, Theorem 6 does not hold for all trees in the pushdown hierarchy. For instance, the pushdown hierarchy contains infinite unary trees corresponding to non-ultimately-periodic infinite words[3] which are therefore pure. However, Theorem 6 can be extended to any tree that does not contain arbitrary long chains of unary vertices.

▶ **Corollary 10.** *Every tree in the pushdown hierarchy with no leaves and in which the length of all chains of unary vertices is bounded has self-similarities.*

Remark that if we simply ask that there is no infinite chains of unary vertices, Corollary 10 no longer holds. It is possible to construct a pure binary tree in Graph_2 in which all chains of unary vertices are finite. As illustrated in Figure 3, an example of such a tree can be obtained by starting with a copy of the complete binary tree and replacing each node $u = u_1 \cdots u_n \in \{0,1\}^+$ by the finite chain $\bullet \xrightarrow{u_1} \bullet \xrightarrow{u_2} \bullet \cdots \xrightarrow{u_n} \bullet$.

5 Proof of Proposition 8

This section is devoted to proving Proposition 8. For the rest of the section, we fix a complete binary tree T and an MSO-interpretation $\mathcal{I}$ such that $\mathcal{I}(T)$ is a pure complete binary tree. Furthermore for each label $x \in \{0,1\}$, we fix a deterministic and non-backtracking tree-walking automaton A_x with states in Q_x and an MSO-recoloring μ_x such that A_x accepts the pair (u,v) on $\mu_x(T)$ if and only if $u \xrightarrow{x} v$ in $\mathcal{I}(T)$. We assume Q_0 and Q_1 are disjoint and take $Q = Q_0 \uplus Q_1$.

[3] For example, the unary trees representing the morphic words belong to Tree_2 (see [10, Proposition 3.2]).

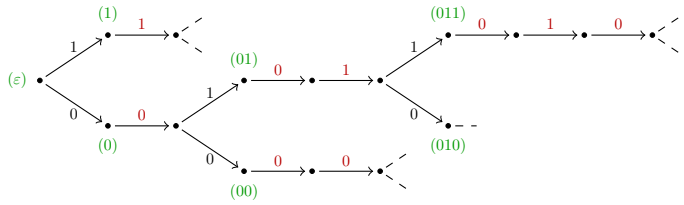

Figure 3 An example of a pure non-complete binary tree in Graph_2. This tree has its edges labeled by $\{0,1\}$. It is obtained by replacing each node $u = u_1 \cdots u_n \in \{0,1\}^+$ of the complete binary tree by a finite line of length $|u|$ in which the i-th edge is labeled by u_i. The corresponding node of the complete binary tree is in green between brackets.

In Subsection 5.1, we define an MSO-definable embedding $\mathcal{S}_\mathcal{I}$ inducing in T a complete binary tree. In Subsection 5.2, we define a MSO-recoloring μ of T. In Subsection 5.3, we show that the binary complete tree induced by $\mathcal{S}_\mathcal{I}$ in $\mu(T)$ is pure.

5.1 Definition of the embedding $\mathcal{S}_\mathcal{I}$

Let us start by remarking that any embedding S in a tree T induces a partition of the nodes of the tree T in regions, we call them chunks. A *chunk* of T is a set of nodes obtained by removing a finite number of subtrees from a subtree of T. Hence a chunk C is described by its *boundary* $(u_0, \ldots, u_n)$ which is a sequence of nodes of T with u_0 the root of the chunk and the $u_1, \ldots, u_n$ are descendants of u_0 which are the pairwise incomparable roots of the subtrees that are removed from $T_{|u_0}$. In other terms, $C = T_{|u_0} \setminus \bigcup_{i\in[1,n]} T_{|u_i}$. We call n the degree of the chunk C.

To every node s in an embedding S, we associate the chunk C_s rooted at s defined by the boundary $(s, s_1, \ldots, s_n)$ where $s_1, \ldots, s_n$ are the children of s in the tree T_S in lexicographical order. This notion is illustrated in Figure 2. The chunks $(C_s)_{s\in S}$ form a partition of the subtree of T rooted at r_S, the root of the embedding S. For our purpose, it is more convenient to obtain a partition of the whole tree T. Hence, we define C_{r_S} by the boundary $(\varepsilon, s_1, \ldots, s_n)$ instead of $(r_S, s_1, \ldots, s_n)$.

The content of a chunk C is the set of nodes kept by the interpretation $\mathcal{I}$ (i.e., $C \cap \mathcal{I}(T)$). We can leverage the fact that $\mathcal{I}(T)$ is a complete binary tree to show that for any chunk with a finite content, the size of the content is bounded by a constant that only depends on the degree of the chunk and on the interpretation $\mathcal{I}$.

▶ **Lemma 11.** *Under the assumptions of this section, for all $m \geqslant 0$, there exists a constant $d \geqslant 0$ depending only on m and on the MSO-interpretation $\mathcal{I}$ such that for each chunk C of degree m, if the content of C is finite, then the size of this content is bounded by d.*

Proof. Let C be a chunk of degree $m \geqslant 0$ with a boundary $(u_0, u_1, \ldots, u_m)$ and let $d := 2(m+1)\max(|Q_0|, |Q_1|)$.

Assume toward a contradiction that C has a finite content U of size $k > d$. As $\mathcal{I}(T)$ is a tree, $\mathcal{I}(T)$ restricted to the content U of C is a forest F with k vertices and hence at most $k-1$ edges. As $\mathcal{I}(T)$ is a complete binary tree, there are at least $2k-(k-1) = k+1$ edges of $\mathcal{I}(T)$ starting in U and ending outside of C. By the pigeonhole principle, at least $\ell > \frac{k+1}{2}$ edges share the same label $x \in \{0,1\}$. Let $(v_1, w_1), \ldots, (v_l, w_l)$ be an enumeration of these edges. As v_i belongs to C and w_i does not, the accepting run of A_x on $\mu_x(T)$ for the pair (v_i, w_i) must cross the boundary of C. By the pigeonhole principle, at least $\frac{\ell}{m+1}$ of these runs cross the boundary of C at the same u_{i_0} with $i_0 \in [0,n]$. As $\frac{\ell}{m+1} > \max(|Q_0|, |Q_1|)$,

there exists two different runs (corresponding to two different edges) reaching u_{i_0} in the same state. This implies that these two different edges must have the same target which contradicts the fact that $\mathcal{I}(T)$ is a tree. ◀

▶ **Definition 12.** *The embedding $\mathcal{S}_\mathcal{I}$ is composed of all nodes u for which both their left and right subtrees contain infinitely many nodes of $\mathcal{I}(T)$.*

Using Lemma 11, we can show that the content of the chunks associated with $\mathcal{S}_\mathcal{I}$ are finite and hence bounded.

▶ **Proposition 13.** *The set $\mathcal{S}_\mathcal{I}$ is an MSO-definable embedding in T defining a complete binary tree. Moreover, for all $s \in \mathcal{S}_\mathcal{I}$, the content of the chunk C_s has size at most d where d is a constant that only depends on $\mathcal{I}$.*

Proof Sketch. For a node u of T, we write $\mathrm{Inf}(u)$ if there are infinitely many nodes of $\mathcal{I}(T)$ below u.

We start by showing that for all node u satisfying $\mathrm{Inf}(u)$, there exist two incomparable descendants v and v' of u such that $\mathrm{Inf}(v)$ and $\mathrm{Inf}(v')$. Assume towards a contradiction that it is not the case for some $u \in T$. Let $v_1, v_2, \dots$ be an enumeration of the descendants v of u for which $\mathrm{Inf}(v)$ holds. For all $i \geqslant 1$, the content of the chunk C_i with boundary (u, v_i) must be finite otherwise it would contain a descendant v of u satisfying $\mathrm{Inf}(v)$ which is incomparable with v_i. By assumption, the v_i's belong to the same infinite branch and hence $T_{|u} = \bigcup_{i \geqslant 1} C_i$. Therefore the size of the content of the C_i's must be unbounded which contradicts Lemma 11.

As the least common ancestor of v and v' belongs to $\mathcal{S}_\mathcal{I}$, we have shown that below every node u satisfying $\mathrm{Inf}(u)$ there exists an element of $\mathcal{S}_\mathcal{I}$. It immediately follows that $\mathcal{S}_\mathcal{I}$ is non-empty and that below each of the two children of an element of $\mathcal{S}_\mathcal{I}$, there exists a unique minimal element in $\mathcal{S}_\mathcal{I}$ (as $\mathcal{S}_\mathcal{I}$ is closed under least common ancestor). This shows that $\mathcal{S}_\mathcal{I}$ induces a complete binary tree. As on deterministic trees the predicate $\mathrm{Inf}(u)$ is MSO-definable, the embedding $\mathcal{S}_\mathcal{I}$ is also MSO-definable.

Using Lemma 11, it only remains to show that content of every chunk C_s for $s \in \mathcal{S}_\mathcal{I}$ is finite. Toward a contradiction, assume that for some $s \in \mathcal{S}_\mathcal{I}$ which is not the root of the embedding, the chunk C_s with boundary (s, s_0, s_1) has an infinite content. There must exist some s' in C_s satisfying $\mathrm{Inf}(s')$ and which is incomparable with both s_0 and s_1. By symmetry, we can assume that s' is in the left subtree of s. In this case, the least common ancestor of s' and s_0 would belong to $\mathcal{S}_\mathcal{I}$ and would be strictly between s and s_0 which would contradict the definition of s_0. The case of the chunk of root of the embedding is treated with similar arguments. ◀

5.2 Definition of the MSO-recoloring μ of the nodes of the embedding

The MSO-recoloring μ will color each vertex $s \in \mathcal{S}_\mathcal{I}$ with a tuple (F_s, π_s, ψ_s) where F_s is the forest obtained by restricting $\mathcal{I}(T)$ to the chunk C_s and π_s and ψ_s are two finite functions describing how to reconnect the forest F_s to the other forests to obtain $\mathcal{I}(T)$.

The forest F_s. For a node $s \in \mathcal{S}_\mathcal{I}$, let $v_1, \dots, v_n$ with $0 \leqslant n \leqslant d$, be an enumeration in lexicographic order of the content of C_s. The nodes of the forest F_s are in $[1, n]$ and for $x \in \{0, 1\}$, $i \xrightarrow{x} j$ in F_s if and only if $v_i \xrightarrow{x} v_j$ in $\mathcal{I}(T)$ and for a color c, i is colored by c in F_s if and only if v_i is colored by c in $\mathcal{I}(T)$.

The partial function π_s. The partial function π_s provides information on the edges that connect the forest F_s to the rest of $\mathcal{I}(T)$. Consider a vertex $i \in [1, n]$ of F_s and a label $x \in \{0, 1\}$ such that i has no outgoing x-labeled edge in F_s. Because $\mathcal{I}(T)$ is complete, there is an edge $v_i \xrightarrow{x} v$ in $\mathcal{I}(T)$. The node v is outside of the chunk C_s and hence the unique run of A_x accepting the pair (v_i, v) on $\mu_x(T)$ must exit the chunk through one of its boundaries[4] $\gamma \in \{\text{root}, \text{left}, \text{right}\}$ in a state $q_\gamma \in Q_x$. As A_x follows the minimal path from v_i to v, it crosses the boundary exactly once. The partial function $\pi_s : [1, d] \times \{0, 1\} \to \{\text{root}, \text{left}, \text{right}\} \times Q$ associates (i, x) to (γ, q_γ) if the x-labeled edge outgoing from i is missing in F_s and is undefined otherwise.

The partial function ψ_s. The partial function ψ_s provides information on the behavior of the automata A_0 and A_1 when entering the chunk C_s from one of its boundaries. Let $\gamma \in \{\text{root}, \text{left}, \text{right}\}$ be a boundary symbol and let s_γ be the corresponding node in the boundary of C_s. The value of $\psi_s(\gamma, q)$ will provide information on the unique accepting run ρ of A_x on $\mu_x(T)$ starting in state q at s_γ (if it exists). If the run ρ enters C_s and ends at a vertex v_i of the content of C_s, then $\psi_s(\gamma, q)$ is defined to be i. If the run ρ enters C_s and exits through a boundary $\gamma \neq \gamma' \in \{\text{root}, \text{left}, \text{right}\}$ in some state q', then $\psi_s(\gamma, q)$ is defined to be (γ', q'). In all other cases, ψ_s is undefined. As a result ψ_s is a partial mapping $\{\text{root}, \text{left}, \text{right}\} \times Q \to [1, d] \cup (\{\text{root}, \text{left}, \text{right}\} \times Q)$.

▶ **Lemma 14.** *The recoloring μ which colors every $s \in \mathcal{S}_\mathcal{I}$ with the color (F_s, π_s, ψ_s) and leave all other nodes uncolored is MSO-definable.*

5.3 The tree induced by $\mathcal{S}_\mathcal{I}$ on $\mu(T)$ is pure

We start by showing how $\mathcal{I}(T)$ can be reconstructed from the complete binary tree induced by the embedding $\mathcal{S}_\mathcal{I}$ in $\mu(T)$. This tree, denoted E in the following, is essentially the mapping associating to every node $s \in \mathcal{S}_\mathcal{I}$ the tuple (F_s, ψ_s, π_s).

Every node in $\mathcal{I}(T)$ corresponds to a unique node in some forest coloring of E. More precisely, for a node $u \in \mathcal{I}(T)$, the address of u is the unique pair (s, i) such that u is the i-th node in lexicographic order of the content of the chunk C_s. Let $\text{Addr} = \{(s, i) \in \mathcal{S}_\mathcal{I} \times [1, d] \mid E(s) = (F_s, \pi_s, \psi_s) \text{ and } i \in F_s\}$ be the set of all valid addresses.

We now define a deterministic tree R whose nodes are the addresses in Addr and which we will prove to be isomorphic to $\mathcal{I}(T)$ in Proposition 16.

In the tree R, an address $(s, i) \in \text{Addr}$ inherits the colors of the corresponding vertex in F_s. Hence, we define $\mathsf{Colors}(s, i)$ to be $\{\theta \in \Theta \mid E(s) = (F_s, \pi_s, \psi_s) \text{ and } (\theta, i) \in F_s\}$. To define the edges of the tree R, we introduce, for each label $x \in \{0, 1\}$, a function Target_x which defines the target of all the x-labeled outgoing edges in R. To define Target_x, we need two auxiliary functions Travel and Follow. These definitions are illustrated in Figure 4.

For $(s, \gamma) \in \mathcal{S}_\mathcal{I} \times \{\text{root}, \text{left}, \text{right}\}$, we define $\mathsf{Travel}(s, \gamma)$ to be equal to (s', γ') if upon leaving C_s by the boundary γ, we enter $C_{s'}$ by its boundary γ'.

For all $s \in \mathcal{S}_\mathcal{I}$, $\gamma \in \{\text{root}, \text{left}, \text{right}\}$ and $q \in Q_x$, $\mathsf{Follow}(s, \gamma, q)$ is recursively defined by (s, j) if $\psi_s(\gamma, q) = j \in [1, d]$ and by $\mathsf{Follow}(s', \gamma'', q')$ if $\psi_s(\gamma, q) = (\gamma', q')$ and $\mathsf{Travel}(s, \gamma') = (s', \gamma'')$. Intuitively, $\mathsf{Follow}(s, \gamma, q)$ gives the address of the end of the run of A_x starting in q at the boundary γ of C_s.

[4] If the boundary of C_s is (s, s_0, s_1), then s correspond to root, s_0 to left and s_1 to right.

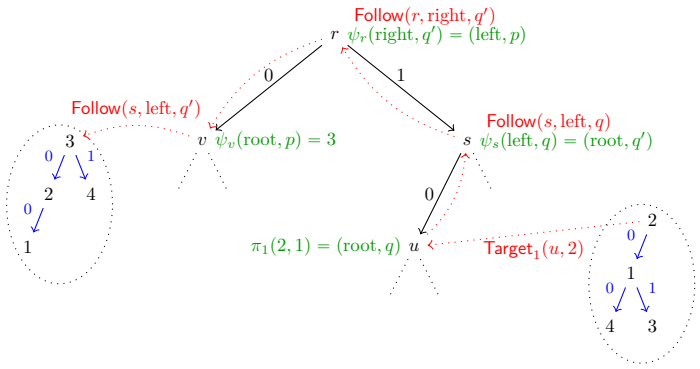

Figure 4 An example of the computation of the function Target. The red path represents the successive recursive calls made when computing $\mathsf{Target}_1(u, 2) = (v, 3)$. As there is no edge from 2 labeled by 1 in F_u, $\mathsf{Target}_1(u, 2)$ calls $\mathsf{Follow}(s, \text{left}, q)$, as $\mathsf{Travel}(u, \text{root}) = (s, \text{left})$ following the function $\pi_1(2, 1) = (\text{root}, q)$. In turn, $\mathsf{Follow}(s, \text{left}, q)$ calls $\mathsf{Follow}(r, \text{right}, q')$ as $\psi_s(\text{left}, q) = (\text{root}, q)$ and $\mathsf{Travel}(s, \text{root}) = (r, \text{right})$. It then proceeds to call $\mathsf{Follow}(s, \text{left}, q')$ as before, which follows $\psi_v(\text{root}, p) = 3$ ending the run on the vertex 3 of the forest F_v.

For $x \in \{0, 1\}$, $\mathsf{Target}_x(s, i)$ is defined to be (s, j) if $i \xrightarrow{x} j$ belongs to F_s otherwise, the x-labeled outgoing edge from i is missing from the forest F_s and $\pi_s(i, x)$ is defined and equal to some $(\gamma, q) \in \{\text{root}, \text{left}, \text{right}\} \times Q_x$ and $\mathsf{Target}_x(s, i)$ is taken to be $\mathsf{Follow}(s', \gamma', q)$ where $(s', \gamma') = \mathsf{Travel}(s, \gamma)$.

▶ **Lemma 15.** *For two addresses (s_u, i_u) and $(s_v, i_v) \in$ Addr respectively corresponding to nodes u and v in $\mathcal{I}(T)$, we have $\mathsf{Target}_x(s_u, i_u) = (s_v, i_v)$ if and only if $u \xrightarrow{x} v$ in $\mathcal{I}(T)$. Furthermore, as the tree-walking automaton A_x is non-backtracking in T, for all $s \in \mathcal{S}_\mathcal{I}$ such that $s \sqsubseteq s_u$ and $s \sqsubseteq s_v$, the recursive calls to* Follow *made when computing $\mathsf{Target}_x(s_u, i_u)$ will all stay below s.*

From this lemma, it follows that the tree R is isomorphic to $\mathcal{I}(T)$.

▶ **Proposition 16.** *The deterministic binary tree R with nodes in* Addr *and defined by* Colors, Target_0 *and* Target_1 *is isomorphic to $\mathcal{I}(T)$.*

We use this reconstruction of $\mathcal{I}(T)$ in E presented above to show that any self-similarity in E would lead to a self-similarity in $\mathcal{I}(T)$ which is impossible by assumption.

▶ **Proposition 17.** *Under the assumptions of this section, for the MSO-recoloring μ and the MSO-definable embedding $\mathcal{S}_\mathcal{I}$ defined previously, the deterministic binary tree E induced by $\mathcal{S}_\mathcal{I}$ on $\mu(T)$ does not have any self-similarities.*

Proof. Assume toward a contradiction that E has a self-similarity and let $s_1 \neq s_2$ be two nodes of E such that $E_{|s_1} \sim E_{|s_2}$. Recall that the nodes of E are the elements of $\mathcal{S}_\mathcal{I}$ and that the ancestor relation $\sqsubseteq_E$ coincide with the ancestor relation $\sqsubseteq_T$ of T on $\mathcal{S}_\mathcal{I} \times \mathcal{S}_\mathcal{I}$. In the following, we say that an address $(t, i) \in$ Addr is below a node s of E if t is descendant of s in E (or equivalently in T).

▷ Claim 18. There exists an address $(t, j) \in$ Addr below s_1 such that all the descendants of (t, j) in the tree R of Proposition 16 have an address below s_1.

Proof. For a label $x \in \{0,1\}$, consider the set X_x of addresses (s,i) below s_1 such that $\mathsf{Target}_x(s,i)$ is not below s_1. By definition of Target_x, there exists $\gamma \in \{\text{left}, \text{right}\}$ such that for all $(s,i) \in X_x$, $\mathsf{Target}_x(s,i)$ must be equal to $\mathsf{Follow}(s_1,\gamma,q)$ for some q. By the pigeonhole principle, X_x is of size at most $|Q_x|$ as otherwise we would have two addresses in X_x with the same value of Target_x which would contradict the fact that R is a tree.

The set Y of addresses (s,i) having a descendant (in R) in the finite set $X_0 \cup X_1$ is itself finite. Let k be the maximal depth of an $s \in E$ such that $(s,i) \in Y$ for some i. As s_1 belongs to $\mathcal{S}_{\mathcal{I}}$, we have by definition of $\mathcal{S}_{\mathcal{I}}$ that there are infinitely many vertices of $\mathcal{I}(T)$ below s_1 in T. Let Z be the corresponding set of addresses. Remark that all these addresses are necessarily below s_1 as if a node $r \in \mathcal{I}(T)$ has an address (s,i) then s is the greatest ancestor of r which belongs to $\mathcal{S}_{\mathcal{I}}$. Hence as Z is infinite, it must contain an element (t,j) with t at depth greater than k in T. By definition of the depth k, this implies that all descendants of (t,j) in R are below s_1. ◁

As $E_{|s_1}$ and $E_{|s_2}$ are isomorphic, there exists a (unique) bijection h that maps every node s of $E_{|s_1}$ to the corresponding node $h(s)$ in $E_{|s_2}$. Let $t' = h(t)$ be the node corresponding to t in $E_{|s_2}$. In particular, t and t' have the same color in R and hence (t',j) is an address in Addr. We claim that $T_1 = R_{|(t,j)}$ is isomorphic to $T_2 = R_{|(t',j)}$ which will bring the contradiction as $R \sim \mathcal{I}(T)$ is assumed to be pure. To see this, consider two descendants (t_1,j_1) and (t_2,j_2) of (t,j) such that $(t_1,j_1) \xrightarrow[R]{x} (t_2,j_2)$ and hence $\mathsf{Target}_x(t_1,j_1) = (t_2,j_2)$. By definition of (t,j), we have that both t_1 and t_2 are below s_1. The recursive calls to Follow made when computing $\mathsf{Target}_x(t_1,j_1)$ stay inside the subtree $E_{|s_1}$ (cf. Lemma 15). As $E_{|s_2}$ is isomorphic to $E_{|s_1}$, we have that $\mathsf{Target}_x(h(t_1),j_1) = (h(t_2),j_2)$. This implies our claim and conclude the proof. ◀

6 Applications

In this section, we leverage the well-known connection between the deterministic trees in the pushdown hierarchy and the trees defined by deterministic higher-order pushdown automata (presented in Subsection 6.1) to give two applications of our main results: one in formal language theory in Subsection 6.2 and one in number theory in Subsection 6.3.

6.1 Higher-order pushdown automata

Higher-order pushdown automata are a generalization of the standard model of pushdown automata. To simplify the presentation, we only define formally higher-order pushdown automata of order 2. We refer the reader to [15] for a definition at all orders.

An order-2 pushdown automaton works on a stack of stacks, called an order-2 stack. We start by defining order-1 and order-2 stacks and the operation to manipulate them. Let Γ be a stack alphabet and let $\perp \notin \Gamma$ be a distinguished bottom of stack symbol. An order-1 stack is a sequence $\perp\gamma_1 \dots \gamma_n \in \perp\Gamma^*$, denoted by $[\,\gamma_1 \dots \gamma_n\,]_1$. The symbol γ_n is the top-most symbol of the stack and $[\,]_1$ is called the empty order-1 stack. An order-2 stack is a non-empty sequence $s_1, \dots, s_n$ of order-1 stacks denoted by $[\,s_1, \dots, s_n\,]_2$. The order-1 stack s_n is the top-most order-1 stack and $[\,[\,]_1\,]_2$ is the empty order-2 stack.

We now define operations on order-2 stacks. For every symbol $\gamma \in \Gamma$, the operation push_γ pushes the symbol γ on the top-most order-1 stack (i.e., $\mathrm{push}_\gamma([\,s_1, \dots, [\,\gamma_1, \dots, \gamma_m\,]_1\,]_2) = [\,s_1, \dots, [\,\gamma_1, \dots, \gamma_m, \gamma\,]_1\,]_2$). The operation pop_1 removes the top-most symbol of the top-most order-1 stack provided that $m \geqslant 1$ (i.e., $\mathrm{pop}_1([\,s_1, \dots, [\,\gamma_1, \dots, \gamma_m\,]_1\,]_2)$ is undefined if

$m = 0$ and is equal to $[\, s_1, \ldots, [\, \gamma_1, \ldots, \gamma_{m-1} \,]_1 \,]_2$ if $m \geqslant 1$). The operation copy_2 copies the top-most order-1 stack (i.e., $\mathrm{copy}_2([\, s_1, \ldots, s_n \,]_2) = [\, s_1, \ldots, s_n, s_n \,]_2$). Finally, the operation pop_2 removes the top-most order-1 stack, if the order-2 stack is not reduced to one order-1 stack (i.e., $\mathrm{pop}_2([\, s_1, \ldots, s_n \,]_2) = [\, s_1, \ldots, s_{n-1} \,]_2$).

An order-2 pushdown automaton P (with ε-transitions) is defined by a tuple (Q, q_0, Σ, Δ) where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is a finite set of input symbols, Δ is the set of transitions of the form $(p, \gamma, \sigma, \mathrm{op}, q)$ with $p, q \in Q$, $\gamma \in \Gamma \cup \{\bot\}$, $\sigma \in \Sigma \cup \{\varepsilon\}$ and op an operation in $\{\mathrm{pop}_1, \mathrm{push}_\gamma, \mathrm{pop}_2, \mathrm{copy}_2\}$. We furthermore assume that states of P can be partitioned into $Q_\Sigma \uplus Q_\varepsilon$ such that states in Q_Σ (resp. in Q_ε) are only the source of transitions labeled by Σ (resp. labeled by ε). The automaton is said to be deterministic if for all $q \in Q$, $\gamma \in \Gamma \cup \{\bot\}$ and $\sigma \in \Sigma \cup \{\varepsilon\}$, there exists at most one transition in Δ starting with (q, γ, σ) and if there exists a transition starting with (q, γ) labeled by ε then it is the only transition starting with (q, γ).

A configuration of an order-2 pushdown automaton is a pair (q, s) with $q \in Q$ and s an order-2 stack. For $\sigma \in \Sigma \cup \{\varepsilon\}$, the automaton P induces a relation $\xrightarrow[P]{\sigma}$ between the configurations of P defined by: $(p, s) \xrightarrow[P]{\sigma} (q, s')$ if there exists a transition $(p, \gamma, \sigma, \mathrm{op}, q)$ with $s' = \mathrm{op}(s)$ and γ is the top-most symbol of the top-most order-1 stack of s. From these relations, we can define the relation $\xRightarrow[P]{w}$ for each $w \in \Sigma^*$ in the usual way. If the automaton P is deterministic, we define for all $w \in \Sigma^*$, $\delta_P(w)$ to be the unique configuration (q, s) (if it exists) such that $(q_0, [\,[\,]_1\,]_2) \xRightarrow[P]{w} (q, s)$ with s a stack and $q \in Q_\Sigma$.

If we fix a set $F \subseteq Q_\Sigma$ of final states, the automaton P accepts the language $L(P)$ of words over Σ defined by $L(P) := \{w \in \Sigma^* \mid (q_0, [\,[\,]_1\,]_2) \xRightarrow[P]{w} (q, s) \wedge q \in F\}$. Figure 5 gives an example of a deterministic order-2 pushdown automaton accepting the language $\{1^n 0^n 1^n \mid n \geqslant 1\}$.

If we fix a finite set Θ of vertex colors and a mapping $\mathrm{Col} : Q_\Sigma \to 2^\Theta$, a deterministic order-2 pushdown automaton P defines a deterministic tree $T(P)$ with edges labeled by Σ and with nodes colored by Θ. This tree is given by the partial function T_P from Σ^* to 2^Θ such that for all $w \in \Sigma^*$, $T_P(w) = \mathrm{Col}(q)$ if $\delta_P(w)$ is defined and equal to (q, s) and is undefined otherwise.

The trees defined in this way are the deterministic trees in the pushdown hierarchy.

▶ **Theorem 19** ([10, 15]). *The deterministic trees in* Tree_n *are the trees defined by deterministic higher-order pushdown automata of order* $n - 1$.

6.2 Deterministic vs non-deterministic higher-order pushdown automata

Let Σ be a finite alphabet. For a language $L \subseteq \Sigma^*$ and a word $w \in \Sigma^*$, recall that the left-quotient of L by w, denoted by $w^{-1}L$, is the set $w^{-1}L := \{u \in \Sigma^* \mid wu \in L\}$. We can leverage Theorem 19 to show that if all the left-quotients of a language are different then this language cannot be accepted by any deterministic higher-order pushdown automaton.

▶ **Theorem 20.** *Let* Σ *be a finite alphabet with at least two symbols and* L *be a language over* Σ*. If for all* $w_1 \neq w_2 \in \Sigma^*$*, we have* $w_1^{-1}L \neq w_2^{-1}L$*, the language* L *cannot be accepted by a deterministic higher-order pushdown automaton of any order. In particular, this is the case for the languages:* $\{ww \mid w \in \Sigma^*\}$*,* $\{w \mid w$ *a palindrome in* $\Sigma^*\}$ *and* $\{w^{f(|w|)} \mid w \in \Sigma^*\}$ *for* $f : \mathbb{N} \to \mathbb{N}$ *strictly increasing.*

Proof Sketch. Let Σ be a finite alphabet with a least two symbols and L be a language over Σ. Assume that for all $w_1 \neq w_2 \in \Sigma^*$, $w_1^{-1}L \neq w_2^{-1}L$.

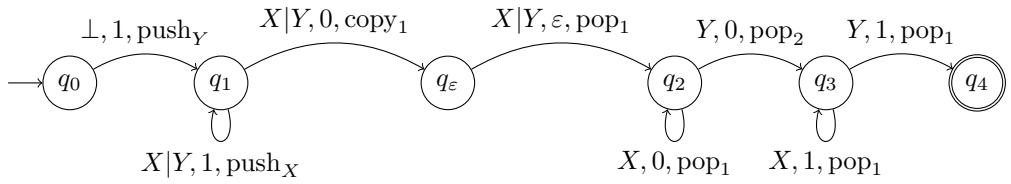

Figure 5 A deterministic higher-order pushdown automaton of order-2 recognizing the language $\{1^n 0^n 1^n \mid n \geqslant 1\}$ with q_0 its initial state and q_4 its final state. Note that the loop on the state q_1 labeled by $X|Y, 1, \text{push}_X$ is actually depicting two transitions: $(q_1, X, 1, \text{push}_X, q_1)$ and $(q_1, Y, 1, \text{push}_X, q_1)$ and similarly for the transitions going from q_1 to q_ε and from q_ε to q_2. For example, the accepting run for the word 110011 is the following $(q_0, [[\,]_1\,]_2) \xrightarrow{1} (q_1, [[\,Y\,]_1\,]_2) \xrightarrow{1} (q_1, [[\,YX\,]_1\,]_2) \xrightarrow{0} (q_\varepsilon, [[\,YX\,]_1[\,YX\,]_1\,]_2) \xrightarrow{\varepsilon} (q_2, [[\,YX\,]_1[\,Y\,]_1\,]_2) \xrightarrow{0} (q_3, [[\,YX\,]_1\,]_2) \xrightarrow{1} (q_3, [[\,Y\,]_1\,]_2) \xrightarrow{1} (q_4, [[\,]_1\,]_2)$.

Consider the deterministic complete tree $T_L : \Sigma^* \to \{0,1\}$ labeled by Σ and colored by $\{0,1\}$ defined by $T_L(w) = 1$ if and only if $w \in L$. By Theorem 19, T_L belongs to the pushdown hierarchy if and only if L is accepted by a deterministic higher-order pushdown automaton of some order. The assumption of the language L is equivalent to requiring all the subtrees of T_L to be non-isomorphic. Hence T_L does not belong to the pushdown hierarchy by Corollary 10 and our claim follows. ◀

We remark that the language $\{w \mid w \text{ a palindrome in } \Sigma^*\}$ is accepted by a non-deterministic pushdown automaton, the language $\{ww \mid w \in \Sigma^*\}$ is accepted by a non-deterministic order-2 pushdown automaton and $\{w^{|w|} \mid w \in \Sigma^*\}$ is accepted by a non-deterministic order-3 pushdown automaton.

6.3 Real numbers defined by deterministic higher-order pushdown automata

We introduce a generalization of the notion of automatic sequence [4] by replacing the deterministic finite automata used to generate automatic sequences by deterministic higher-order pushdown automata. This generalization follows an approach initiated in [3, 11] and is explicitely mentioned in the conclusion of [11].

Recall that an automatic sequence in base $b \geqslant 2$ is an infinite sequence $\lambda_1\lambda_2\cdots \in \Lambda^\omega$ which is defined using a deterministic finite automaton A over the alphabet $\Sigma_b := \{0, 1, \ldots, b-1\}$. For all $i \geqslant 1$, the automaton outputs the symbol λ_i after having read the decomposition $\langle i \rangle_b$ of i in base b. For the automaton A to output symbols in Λ, we simply associate a symbol in Λ to every state of A using a mapping $\text{Output} : Q \to \Lambda$.

Similarly, a deterministic order-k pushdown automaton P over $\Sigma_b := \{0, \ldots, b-1\}$ equipped with an output function $\text{Output} : Q_\Sigma \to \Lambda$ defines a sequence $\lambda_1\lambda_2\cdots\Lambda^\omega$ if for all $i \geqslant 1$, $\delta_P(\langle i \rangle_b) = (q, s)$ with $\text{Output}(q) = \lambda_i$. If such an automaton exists, the sequence $\lambda_1\lambda_2\cdots \in \Lambda^\omega$ is said to be automatic of order k in base b.

For two bases ℓ and $b \geqslant 2$, a real number in $[0,1]$ is said to be (ℓ, b)-automatic of order-k if it admits an expansion $0.\alpha_1\alpha_2\cdots$ in base ℓ and the sequence $\alpha_1\alpha_2\cdots \in \{0, \ell-1\}^\omega$ is automatic of order-k in base b.

For instance, consider the real number α whose binary expansion is $0.\alpha_1\alpha_2\cdots$ with $\alpha_m = 1$ if $m = 2^{3n} - 2^{2n} + 2^n - 1$ for some $n \geqslant 1$ and $\alpha_m = 0$ otherwise. This number $\alpha = \sum_{n \geqslant 1} 2^{-2^{3n}+2^{2n}-2^n+1}$ is $(2,2)$-automatic of order-2. To see this, remark that for all $n \geqslant 1$,

the binary decomposition of $2^{3n} - 2^{2n} + 2^n - 1$ is $1^n0^n1^n$. Hence an order-2 deterministic pushdown automaton describing the sequence $\alpha_1\alpha_2\dots$ can be obtained by making the deterministic order-2 pushdown automaton presented in Figure 5 complete.

An alternative definition of (ℓ, b)-automatic numbers of order-k can be achieved by considering what we call the (ℓ, b)-tree of the real number. For a real number α with a presentation $0.\alpha_1\alpha_2\cdots$ in base $\ell \geqslant 2$, we can associate its (ℓ, b)-tree T_α which is the deterministic tree whose edges are labeled by $\{0, \dots, b-1\}$ and whose nodes are colored by a unique color in $\{0, \dots, \ell - 1\}$. The domain of the tree T_α is $\{\varepsilon\} \cup [1, \ell - 1][0, \ell - 1]^*$, for all $i \geqslant 1$, $T_\alpha(\langle i \rangle_b) = \alpha_i$ and by convention, $T_\alpha(\varepsilon)$ is taken to be 0. Thanks to Theorem 19, a real number α is (ℓ, b)-automatic of order k if and only if it admits an (ℓ, b)-tree in Tree_{k+1}.

The notion of (ℓ, b)-tree is implicit in the work of [3] where the authors use it to give a sufficient condition for a irrational number to be transcendental.

▶ **Theorem 21** ([3]). *Let $\ell \geqslant 2$, $b \geqslant 2$, $0 \leqslant \alpha \leqslant 1$ a real number and let T_α be an (ℓ, b)-tree for α. If T_α has self-similarities then α is either rational or transcendental.*

By Corollary 10, it immediately follows that:

▶ **Corollary 22.** *For all $\ell \geqslant 2$ and all $b \geqslant 2$, the (ℓ, b)-automatic real number of order k are either rational or transcendental.*

7 Conclusion

In this article, we have shown that every tree in the pushdown hierarchy with no leaves and with chains of unary vertices of bounded length must contain self-similarities. This in particular implies that the trees produced by safe recursion schemes have this property. It is an ongoing work to generalize this property to general unsafe recursion schemes [17]. This would in particular prove that collapsible pushdown automata, a generalization of higher-order pushdown automata cannot be used to generate irrational algebraic numbers [14].

References

1 Luca Aceto, Arnaud Carayol, Zoltán Ésik, and Anna Ingólfsdóttir. Algebraic synchronization trees and processes. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 30–41. Springer, 2012. doi:10.1007/978-3-642-31585-5_7.

2 Boris Adamczewski and Yann Bugeaud. On the complexity of algebraic numbers I. Expansions in integer bases. *Annals of Mathematics*, 165(2):547–565, 2007.

3 Boris Adamczewski, Julien Cassaigne, and Marion Le Gonidec. On the computational complexity of algebraic numbers : the Hartmanis-Stearns problem revisited. *Transactions of the American Mathematical Society*, 373:3085–3115., 2020.

4 Jean-Paul Allouche and Jeffrey O. Shallit. *Automatic Sequences - Theory, Applications, Generalizations*. Cambridge University Press, 2003. URL: http://www.cambridge.org/gb/knowledge/isbn/item1170556/?site_locale=en_GB.

5 Achim Blumensath. On the structure of graphs in the Caucal hierarchy. *Theor. Comput. Sci.*, 400(1-3):19–45, 2008. doi:10.1016/J.TCS.2008.01.053.

6 Achim Blumensath. Erratum to "On the structure of graphs in the Caucal hierarchy" [Theoret. Comput. Sci 400 (2008) 19-45]. *Theor. Comput. Sci.*, 475:126–127, 2013. doi:10.1016/J.TCS.2012.12.044.

7 Laurent Braud and Arnaud Carayol. Linear orders in the pushdown hierarchy. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2010. doi:10.1007/978-3-642-14162-1_8.

8 Arnaud Carayol and Stefan Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 112–123. Springer, 2003.

9 Didier Caucal. On infinite transition graphs having a decidable monadic theory. In Friedhelm Meyer auf der Heide and Burkhard Monien, editors, *Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings*, volume 1099 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1996. doi:10.1007/3-540-61440-0_128.

10 Didier Caucal. On infinite terms having a decidable monadic theory. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2002. doi:10.1007/3-540-45687-2_13.

11 Didier Caucal and Marion Le Gonidec. Context-free sequences. In Gabriel Ciobanu and Dominique Méry, editors, *Theoretical Aspects of Computing - ICTAC 2014 - 11th International Colloquium, Bucharest, Romania, September 17-19, 2014. Proceedings*, volume 8687 of *Lecture Notes in Computer Science*, pages 259–276. Springer, 2014. doi:10.1007/978-3-319-10882-7_16.

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

13 Bruno Courcelle and Igor Walukiewicz. Monadic second-order logic, graph coverings and unfoldings of transition systems. *Ann. Pure Appl. Log.*, 92(1):35–62, 1998. doi:10.1016/S0168-0072(97)00048-1.

14 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. *ACM Trans. Comput. Log.*, 18(3):25:1–25:42, 2017. doi:10.1145/3091122.

15 Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *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*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. doi:10.1007/3-540-45931-6_15.

16 A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12:38–43, 1976.

17 C.-H. Luke 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*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.

18 Paweł Parys. A pumping lemma for pushdown graphs of any level. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 54–65. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPICS.STACS.2012.54.

19 Paweł Parys. On the expressive power of higher-order pushdown systems. *Log. Methods Comput. Sci.*, 16(3), 2020. URL: https://lmcs.episciences.org/6723.

20 Paweł Parys. The Caucal hierarchy: Interpretations in the (W)MSO+U logic. *Inf. Comput.*, 286:104782, 2022. doi:10.1016/J.IC.2021.104782.

21 Wolfgang Thomas. Constructing infinite graphs with a decidable mso-theory. In Branislav Rovan and Peter Vojtás, editors, *Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings*, volume 2747 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2003. doi:10.1007/978-3-540-45138-9_6.

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