A Contraction Method to Decide MSO Theories of Deterministic Trees

Angelo Montanari Gabriele Puppis
Dipartimento di Matematica e Informatica, Università di Udine
{*montana,puppis*}*@dimi.uniud.it*

Abstract

In this paper we generalize the contraction method, originally proposed by Elgot and Rabin and later extended by Carton and Thomas, from labeled linear orderings to colored deterministic trees. The method we propose rests on a suitable notion of indistinguishability of trees with respect to tree automata that allows us to reduce a number of instances of the acceptance problem for tree automata to decidable instances involving regular trees. We prove that such a method works effectively for a large class of trees, which is closed under noticeable operations and includes all the deterministic trees of the Caucal hierarchy obtained via unfoldings and inverse finite mappings as well as several trees outside such a hierarchy.

1 Introduction

Monadic Second-Order (MSO) logic has been commonly used as a specification language, because it is powerful enough to express relevant properties of graph structures such as reachability and confluency properties. Unfortunately, the model-checking problem for the MSO logic (i.e., the problem of deciding whether a given MSO formula holds in a relational structure or not) turns out to be highly undecidable for many structures. Decidability has been proved for the linear order $(\omega, <)$ (resp., the infinite complete binary tree) by Büchi (resp., Rabin) by reducing the model checking problem to the acceptance problem for Büchi (resp., Rabin tree) automata [12]. Büchi's result has been exploited to deal with extensions of $(\omega, <)$ with unary predicates. The acceptance problem for an expanded structure $(\omega, <, P)$, with $P \subseteq \omega$, is the problem of determining, for any given Büchi automaton M, whether M accepts (the infinite word that characterizes) $(\omega, <, P)$. Elgot and Rabin gave a positive answer to this problem for various relevant predicates, e.g., the factorial one [7]. Intuitively, their approach consists in defining a transformation of a given infinite word u into another infinite word u' and a transformation of a Büchi automaton M into another automaton M' in such a way that M accepts u iff M' accepts u'. If u' happens to be ultimately periodic, the latter condition (and hence the original question) can be checked effectively. In [3] Elgot and Rabin's method is generalized to deal with the class of profinitely ultimately periodic words, whose acceptance problem can be traced back to the case of ultimately periodic words.

In this paper, we show that a similar approach can be followed to deal with expanded tree structures. Here the role of ultimately periodic words is taken by regular colored trees and the notion of factorization for infinite words is accordingly generalized to branching structures. More precisely, we show that the model-checking problem for MSO logic interpreted over deterministic vertex-colored trees can be reduced to the problem of deciding whether a Rabin/Muller tree automaton accepts a given relational structure viewed as a deterministic vertex-colored tree (acceptance problem). Such a problem is easily decidable in the case of regular trees. By exploiting a suitable notion of tree equivalence (indistinguishability) with respect to Rabin/Muller tree automata, we show that it is decidable over a large class of non-regular trees as well. We also prove that such a class is closed with respect to natural operations on trees. Finally, we show that it includes meaningful relational structures inside and outside the so-called Caucal hierarchy [4].

The paper extends and refines previous results presented in [8] by introducing a more general notion of factorization, which allows reductions towards branching structures, instead of linear ones, and by identifying a much larger class of reducible trees, which is closed under several natural transformations on trees. The proposed approach is somehow related to Shelah's composition method, which directly exploits indistiguishability of relational structures with respect to MSO formulas [11]. However, unlike our automaton-based one, Shelah's composition method finds it difficult to manage different valuations of a given variable over distinct copies of the same factor. This is a problem, in particular, when one needs to reduce a branching structure to a linear one (see the example in Section 3.2). An extended version of the work can be found in [10].

2 Basic notation and terminology

We use the term *label* (resp., *color*) to identify a symbol, usually taken from a finite set A (resp., C), marking an edge (resp., a vertex) of a graph. An A-*labeled* graph is a tuple $G = (D, (E_a)_{a\in A})$, where D (also denoted $\mathcal{D}om(G)$) is a set of vertices and $(E_a)_{a\in A}$ are binary relations defining the edges and their labels. An *expanded graph* is a graph equipped with some unary predicates, namely, a structure $(G, \bar{V})$, where G is a graph and $\bar{V} = (V_1, ..., V_m)$, with $V_i \subseteq \mathcal{D}om(G)$ for all $1 \leq i \leq m$. Any expanded graph $(G, \bar{V})$ is canonically represented by a C-*colored graph* $G_{\bar{V}} = (G, \Omega)$, where $C = \mathscr{P}(\{1, \ldots, m\})$ and $\Omega : \mathcal{D}om(G) \rightarrow C$ is a coloring function mapping a vertex $v \in \mathcal{D}om(G)$ to the set of all indices $1 \leq i \leq m$ such that $v \in V_i$. A *tree* is a graph such that, for every vertex v, there exists a unique path, called *access path*, from a designated source vertex, called *root*, to v. We identify each vertex $v \in D$ of a tree (resp., deterministic tree) with its access path (resp., with the sequence of labels in its access path). In particular, we view a deterministic A-labeled C-colored tree T as a partial function from $D \subseteq A^*$ to C, where D is a *prefix-closed* language over A. Accordingly, we denote the color of a vertex v of T by $T(v)$ and the a-successor of v in T by va. Unless otherwise stated, we assume that trees are deterministic, labeled over a finite set A, and colored over a finite alphabet C. A *full* tree is a deterministic tree such that, whenever $(u, ua) \in E_a$ holds for some $a \in A$, then $(u, ua') \in E_{a'}$ holds for every $a' \in A$. An important role is played by the so-called (C-colored) B-*augmented* trees. These trees have internal nodes colored over C and leaves colored over B.

As for tree automata, we make use of Muller acceptance conditions, rather than Rabin ones. A *(Muller) tree automaton* is a tuple $M = (S, A, C, \Delta, \mathcal{I}, \mathcal{F})$, where S is a finite set of states, A is a finite set of labels, C is a finite set of colors, $\Delta \subseteq S \times C \times S^A$ is a transition relation, $\mathcal{I} \subseteq S$ is a set of initial states, and $\mathcal{F} \subseteq \mathscr{P}(S)$ is a family of sets of final states. Given an *infinite complete* tree T, a run of M on T is an infinite complete S-colored tree P such that for every $v \in \mathcal{D}om(P)$, $\big(P(v), T(v), (P(va))_{a\in A}\big) \in \Delta$. We say that P is successful, and hence T is accepted by M (shortly $T \in \mathscr{L}(M)$), if $P(\varepsilon) \in \mathcal{I}$ and for every infinite path π in P, $\mathcal{I}nf(P|\pi) \in \mathcal{F}$, where $P|\pi$ denotes the sequence of states along π and $\mathcal{I}nf(P|\pi)$ denotes the set of all states that occur infinitely often along π.

3 The automaton-based decision method

In this section we develop an automaton-based method to decide MSO theories of deterministic trees. The method can be viewed as a generalization of the contraction method, originally proposed by Elgot and Rabin and later extended by Carton and Thomas, which exploits noticeable properties of an 'indistinguishability' relation for Büchi automata to decide MSO theories of expanded linear orders [3, 7].

Rabin's Theorem [12] establishes a strong correspondence between MSO formulas satisfied by an expanded tree $(T, \bar{V})$ and Rabin (equivalently, Muller) tree automata accepting its canonical representation $T_{\bar{V}}$: for every formula $\varphi(\bar{X})$, one can compute a tree automaton M (and, conversely, for every tree automaton M, one can compute a formula $\varphi(\bar{X})$) such that $T \vDash \varphi[\bar{V}]$ iff $T_{\bar{V}} \in \mathscr{L}(M)$. Let us call *acceptance problem* for a tree $T_{\bar{V}}$, denoted $Acc(T_{\bar{V}})$, the problem of deciding, for any tree automaton M, whether M accepts $T_{\bar{V}}$. We have that the MSO theory of an expanded tree structure $(T, \bar{V})$ is decidable iff $Acc(T_{\bar{V}})$ is decidable. For the sake of simplicity, hereafter we shall omit the subscript $\bar{V}$, thus writing T for $T_{\bar{V}}$. Given a *regular* tree T, by viewing T as a tree automaton recognizing the singleton $\{T\}$ and by exploiting the closure of tree automata with respect to intersection and the decidability of their emptiness problem, one can easily show that the problem $Acc(T)$ is decidable. In the following, we extend such a result to a large class of trees, including non-regular ones.

3.1 Indistinguishability of trees

Here we introduce the basic ingredients of the automaton-based approach to the decidability of MSO theories of tree structures we propose. It is worth pointing out that the definitions given in this section can be easily tailored to different types of automata, such as, for instance, Rabin tree automata and parity tree automata.

First of all, we slightly generalize the notion of tree automaton in order to allow computations over incomplete non-empty full B-augmented trees. To this end, it suffices to extend the acceptance condition of the automaton: we define a B-*augmented (Muller) tree automaton* as a tuple $M = (S, A, B, C, \Delta, \mathcal{I}, \mathcal{F}, \mathcal{B})$, where B is the set of colors for the leaves of the input tree and $\mathcal{B} \subseteq B \times S \times \mathscr{P}(S)$ specifies the acceptance condition for the leaves of a B-augmented tree. A run of M on a *non-empty full* B-*augmented* tree T is an S-colored tree P such that (i) $\mathcal{D}om(P) = \mathcal{D}om(T)$ and (ii) for every internal vertex v of P, $\big(P(v), T(v), (P(va))_{a\in A}\big) \in \Delta$ (note that only the colors of the internal nodes of T are involved in the definition of run of M). We say that the run P is successful, and hence T is accepted by M, if (i) $P(\varepsilon) \in \mathcal{I}$, (ii) for every infinite path π in P, $\mathcal{I}nf(P|\pi) \in \mathcal{F}$, and (iii) for every leaf v in P, $(T(v), P(v), \mathcal{I}mg(P|\pi_v)) \in \mathcal{B}$, where π_v denotes the access path of v in P and $\mathcal{I}mg(P|\pi_v)$ denotes the set of states that occur along π_v.

Relevant information about a run of a given automaton on a tree is collected in a suitable data structure, called *feature* (of the run of the automaton on the tree).

Definition 1. Let T be a *non-empty full* B-augmented tree and P a run of a B-augmented tree automaton M on T. We define the *feature* $[T, P]$ as the triple

$$\begin{pmatrix} P(\varepsilon), \\ \{\mathcal{I}nf(P|\pi) \,:\, \pi \in \mathcal{B}ch(P)\}, \\ \{(T(v), P(v), \mathcal{I}mg(P|\pi_v) \,:\, v \in \mathcal{F}r(P)\} \end{pmatrix}$$

where $\mathcal{B}ch(P)$ is the set of infinite paths in P originating from the root and $\mathcal{F}r(P)$ is the set of all leaves of P.

The above definition accounts for the occurrences of states along finite and infinite paths in the run P. In particular, the first component of the feature $[T, P]$ identifies the state at the root of the run, the second component identifies, for every infinite path π in T, the set of states that occur infinitely often along π, and the third component identifies, for every leaf v of T, the color of v, the state at v, and the set of states that occur at least once along the access path of v.

In order to generalize the notions of run and feature to empty and non-full trees, we introduce a fresh symbol $\perp$ and we assume that automata read $\perp$ on the missing successors, as well as on their descendants, of any internal vertex of a tree. More precisely, we introduce a suitable operation, called *completion*, which extends a B-augmented tree T by appending infinite complete $\perp$-colored trees to every missing successor of an internal vertex. If T is the empty tree, then the completion of T, denoted $T_\perp$, is the infinite complete $\perp$-colored tree. If T is a non-empty tree, then the completion $T_\perp$ of T is defined as follows: we set $F = \{va \,:\, v \in \mathcal{D}om(T) \setminus \mathcal{F}r(T), a \in A, va \notin \mathcal{D}om(T)\}$ and we let $\mathcal{D}om(T_\perp) = \mathcal{D}om(T) \cup (F \cdot A^*)$, $T_\perp(v) = T(v)$, for every $v \in \mathcal{D}om(T)$, and $T_\perp(v) = \perp$, for every $v \in F \cdot A^*$. Note that $T_\perp$ is a non-empty, full, B-augmented tree. This makes it possible to apply the notions of run and feature to any given B-augmented tree, under the proviso that the input alphabet of the automaton contains the dummy symbol $\perp$.

Given a B-augmented tree T and a B-augmented tree automaton M, in order to decide whether $T \in \mathscr{L}(M)$, we introduce the notion of M-type of T, which is a collection of features of the form $[T_\perp, P]$, where P ranges over a suitable set $\mathcal{P}$ of runs of M on $T_\perp$ (different choices for $\mathcal{P}$ may result into different M-types of T). We allow $\mathcal{P}$ to be a *proper subset* of the set of all runs of M on $T_\perp$, because there can be runs which are subsumed by other ones and thus can be 'forgotten'. The notion of subsumed run is as follows. Given two runs P, P' of M on $T_\perp$, we say that P' is *subsumed* by P, and we write $P \preceq P'$, iff (i) $P(\varepsilon) = P'(\varepsilon)$, (ii) for every infinite path π in P, there is an infinite path π' in P' such that $\mathcal{I}nf(P|\pi) = \mathcal{I}nf(P'|\pi')$, and (iii) for every leaf v in P, there is a leaf v' in P' such that $T(v) = T(v')$, $P(v) = P'(v')$, and $\mathcal{I}mg(P|\pi_v) = \mathcal{I}mg(P'|\pi_{v'})$. Notice that the relation $\preceq$ is a *quasi-order*. Moreover, if P and P' are two runs of M on $T_\perp$ and if $P \preceq P'$, then P is successful whenever P' is successful. Given a set $\mathcal{P}$ of partial runs of M on $T_\perp$, we say that $\mathcal{P}$ is *complete* if for every partial run P' of M on $T_\perp$, there is $P \in \mathcal{P}$ such that $P \preceq P'$.

Definition 2. Given a B-augmented tree automaton M and a B-augmented tree T, an *M-type* of T is a set of features of the form $[T_\perp, P]$, where P ranges over a *complete* set $\mathcal{P}$ of runs of M on $T_\perp$. The *basic* M-type of T is the (unique) set of features of the form $[T_\perp, P]$, where P ranges over *all* runs of M on $T_\perp$.

Note that the notion of M-type is independent from the acceptance condition of M, that is, from $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{B}$. We denote by $\mathscr{T}_M$ the set of all possible M-types of a B-augmented tree automaton M. Since $\mathscr{T}_M$ is included in the *finite* set $\mathscr{P}(S \times \mathscr{P}(\mathscr{P}(S)) \times \mathscr{P}(B \times S \times \mathscr{P}(S)))$, for any M there exist only finitely many different M-types.

We have that the acceptance problem of a B-augmented tree T is equivalent to the problem of computing (and checking) one of its M-type, for any given B-augmented tree automaton M. On the one hand, given an automaton $M = (S, A, B, C \cup \{\perp\}, \Delta, \mathcal{I}, \mathcal{F}, \mathcal{B})$ and an M-type t of a B-augmented tree T, we have that T is accepted by M iff t contains a feature of the form $\big(s, \{X_i\}_{i\in I}, \{(b_j, s_j, Y_j)\}_{j\in J}\big)$, with $s \in \mathcal{I}$, $X_i \in \mathcal{F}$ for all $i \in I$, and $(b_j, s_j, Y_j) \in \mathcal{B}$ for all $j \in J$. Such a condition can be easily checked on the given M-type t (as a matter of fact, this implies that pairs of B-augmented trees T, T' having a common M-type are indistinguishable by the automaton M, that is, $T \in \mathscr{L}(M)$ iff $T' \in \mathscr{L}(M)$). On the other hand, if $Acc(T)$ is decidable (this is the case, for instance, with regular trees), then, for every automaton $M = (S, A, B, C \cup \{\perp\}, \Delta, \mathcal{I}, \mathcal{F}, \mathcal{B})$, one can compute an M-type of T by exploiting an automaton-driven selection of the features of $T_\perp$. Such a selection can be done by building, for any candidate feature $f = \big(s, \{X_i\}_{i\in I}, \{(b_j, s_j, Y_j)\}_{j\in J}\big)$, an automaton $M_f = \big(S, A, B, C \cup \{\perp\}, \Delta, \{s\}, \{X_i\}_{i\in I}, \{(b_j, s_j, Y_j)\}_{j\in J}\big)$, which differs from M at most in the acceptance condition, that accepts $T_\perp$ iff f belongs to an M-type of T.

3.2 From trees to their retractions

We now show how M-types can actually be exploited to solve non-trivial instances of the acceptance problem. To this end, we introduce the notion of factorization, which allows us to decompose a tree T into its basic components. Each component, called *factor*, is obtained by selecting the elements of T that lie in between some distinguished vertices. Taking advantage of the notion of factorization, we define a *retraction* of T, which is a tree-shaped arrangement of M-types corresponding to the factors of T; then, we prove that the acceptance problem for T can be reduced to the acceptance problem for a retraction of it.

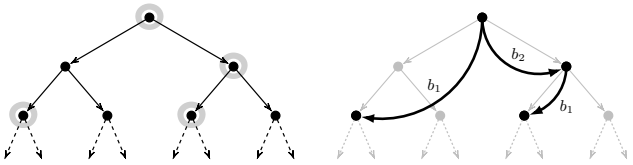

Figure 1. An example of a factorization.

Definition 3. Given a B-augmented tree T, a *factorization* Π of T is a (possibly non-deterministic) *B-labeled uncolored* tree such that (i) $\varepsilon \in \mathcal{D}om(\Pi)$, (ii) $\mathcal{D}om(\Pi) \subseteq \mathcal{D}om(T)$, (iii) for all $u, u' \in \mathcal{D}om(\Pi)$, with $u \neq u'$, (u, u') is an edge of Π iff u' is a descendant of u in T and there exists no other $u'' \in \mathcal{D}om(\Pi)$ that occurs along the path from u to u' in T, and (iv) the edge labels of Π are arbitrarily chosen in B.

Since a B-augmented tree T can be viewed as a B'-augmented tree, for any $B' \supseteq B$, one can use arbitrarily large sets B' to label the edges of a factorization. From now on, if not explicitly mentioned, we shall assume the set of edge labels of a factorization of a B-augmented tree T to be exactly B. We can graphically represent a factorization of a tree by first identifying its vertices (the nodes of the left-hand side tree of Figure 1 which are surrounded by gray circles) and then drawing the resulting edges, together with their labels (the right-hand side tree of Figure 1).

The (marked) factors of a B-augmented tree T with respect to a factorization Π are defined as follows. Let $u \in \mathcal{D}om(\Pi)$ and $Succ(u) = \{u' : (u, u') \text{ is an edge of } \Pi\}$. The *unmarked factor* of T rooted at u, denoted by $T_\Pi[u]$, is the tree obtained by selecting the vertices of $T_\perp$ which are descendants of u, but not proper descendants of any $u' \in Succ(u)$, in $T_\perp$. For any vertex u ($\neq \varepsilon$) in Π, we define the *marker* of u as the label $b \in B$ of the (unique) edge in Π having target u, and we denote it by $m_\Pi[u]$. The *marked factor* of T rooted at u, denoted $T^+_\Pi[u]$, is the tree obtained from $T_\Pi[u]$ by recoloring each leaf u' with the corresponding marker $m_\Pi[u']$. Note that every marked factor of T is a *non-empty full* B-augmented tree.

Hereafter, we assume that all marked factors are *regular* trees. Such an assumption guarantees that M-types of marked factors can be computed; moreover, it excludes trivial factorizations, e.g., those consisting of a single factor.

Definition 4. Let T be a B-augmented tree, M a B-augmented tree automaton, and Π a factorization of T. A *retraction* R of T with respect to M and Π is a *B-labeled $\mathscr{T}_M$-colored* tree such that (i) $\mathcal{D}om(R) = \mathcal{D}om(\Pi)$, (ii) for every $b \in B$, (u, u') is a b-labeled edge in R iff (u, u') is a b-labeled edge in Π, and (iii) each vertex u of R is colored with an M-type (in $\mathscr{T}_M$) of the corresponding marked factor $T^+_\Pi[u]$.

In general, a retraction R, as well as a factorization Π, may be a nondeterministic tree, possibly having vertices with unbounded (or even infinite) out-degree. Since tree automata operate on *deterministic* trees, we have to identify a retraction R with a suitable infinite complete deterministic tree. This is possible, for instance, if for every pair of edges (u, u') and (u, u'') in R labeled with the same symbol, the subtrees of R rooted at u' and u'' are isomorphic. In such a case, by collapsing isomorphic subtrees issued from edges labeled with the same symbol, one can obtain a deterministic B-labeled tree bisimilar to R. From now on, we restrict ourselves to retractions which are *bisimilar* to deterministic trees. Formally, we say that a tree $\vec{R}$ *encodes* a retraction R if $\vec{R}$ is an infinite complete deterministic tree bisimilar to the infinite complete tree obtained from R by adding $\perp$-colored vertices. The encoding of a retraction is unique up to isomorphisms and it can be provided in input to a suitable tree automaton.

We now show how, given a tree T, an automaton M, a factorization Π of T, and a retraction R of T with respect to M and Π, one can build a suitable tree automaton $\overrightarrow{M}$ such that M accepts T iff $\overrightarrow{M}$ accepts the encoding $\vec{R}$ of R. $\overrightarrow{M}$ mimics the behavior of M at a 'coarser' level and it can be effectively computed from M. Its input alphabet is the set $\mathscr{T}_M$ of all M-types plus the additional symbol $\perp$; its states encode the finite amount of information processed by M during its computations up to a certain point; its transitions compute the new states which extend information given by the current state with information provided by the input symbol, i.e., the M-type of a marked factor or $\perp$. $\overrightarrow{M}$ is formally defined as follows.

Definition 5. Given a B-augmented tree automaton $M = (S, A, B, C \cup \{\perp\}, \Delta, \mathcal{I}, \mathcal{F}, \mathcal{B})$, we define the automaton $\overrightarrow{M} = (Q, B, \mathscr{T}_M \cup \{\perp\}, \Delta', \mathcal{I}', \mathcal{F}')$, where:

- $Q = \big\{(d, x, \mathcal{U}, \mathcal{V}) : d \in B, x \in \{0, 1\}, \mathcal{U} \subseteq \mathscr{P}(S), \mathcal{V} \subseteq S \times \mathscr{P}(S) \times \mathscr{P}(S)\big\}$;
- for every state $q = (d, x, \mathcal{U}, \mathcal{V})$ and every tuple of states $(q_b)_{b\in B}$, $\big(q, \perp, (q_b)_{b\in B}\big) \in \Delta'$ iff for all $b \in B$, $q_b = (d, 1, \mathcal{U}, \mathcal{V})$;
- for every M-type t of the form $\big\{\big(s_h, \{X_{h,i}\}_{i\in I}, \{(b_j, s_{h,j}, Y_{h,j})\}_{j\in J}\big)\big\}_{h\in H}$, where H, I, and J are suitable index sets, every state $q = (d, 0, \mathcal{U}, \mathcal{V})$, where $\mathcal{U} = \{U_l\}_{l\in L}$ and $\mathcal{V} = \{(r_g, V_g, W_g)\}_{g\in G}$ for suitable index sets L and G, and every tuple of states $(q_b)_{b\in B}$, $\big(q, t, (q_b)_{b\in B}\big) \in \Delta'$ iff there exists a mapping $h : G \to H$ such that (i) for all $g \in G$, $r_g = s_{h(g)}$ and (ii) for all $b \in B$, $q_b = (b, 0, \mathcal{U}_b, \mathcal{V}_b)$, with $\mathcal{U}_b = \mathcal{U} \cup \{X_{h(g),i}\}_{g\in G, i\in I}$ and $\mathcal{V}_b = \{(s_{h(g),j}, Y_{h(g),j}, W_g \cup Y_{h(g),j})\}_{g\in G, j\in J \text{ with } b_j = b}$;
- $\mathcal{I}'$ consists of all states of the form $(d, 0, \emptyset, \{(s, \emptyset, \emptyset)\})$, with $d \in B$ and $s \in \mathcal{I}$;
- $\mathcal{F}'$ consists of all sets of states $\{q_1, ..., q_n\}$ such that

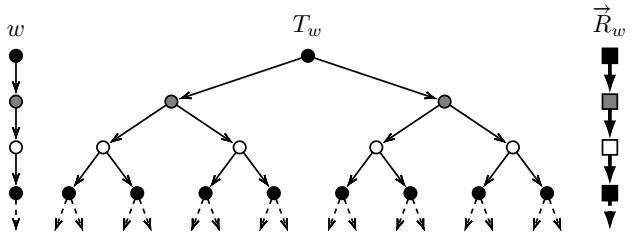

Figure 2. An application of Theorem 1.

(i) $q_1 = (d_1, x, \mathcal{U}, \mathcal{V}_1)$, ..., $q_n = (d_n, x, \mathcal{U}, \mathcal{V}_n)$, (ii) $\mathcal{U} \subseteq \mathcal{F}$, (iii) if $x = 1$, then $n = 1$, $\mathcal{V}_1 = \{(r_g, V_g, W_g)\}_{g \in G}$, where G is a suitable index set, and for all $g \in G$, $(d_1, r_g, W_g) \in \mathcal{B}$, and (iv) if $x = 0$ and $\mathcal{V}_k = \{(r_{k,g}, V_{k,g}, W_k)\}_{g \in G_k}$, where G_k is a suitable index set, for $k = 1, ..., n$, then $\emptyset \subsetneq G'_1 \subseteq G_1$, ..., $\emptyset \subsetneq G'_n \subseteq G_n$ implies $\bigcup_{1 \leq k \leq n, g \in G'_k} V_{k,g} \in \mathcal{F}$.

The following theorem provides a way to compute an M-type t of a tree T from a given $\overrightarrow{M}$-type t' of the encoding $\vec{R}$ of a retraction of T[1]. As a corollary of Theorem 1, we have that the acceptance problem for M over T can be effectively reduced to that for $\overrightarrow{M}$ over $\vec{R}$. The upshot of such a result is that, given an automaton M running on a tree T, if we are able to compute (a representation of) the encoding $\vec{R}$ of a retraction of T with respect to M, then the decidability of $Acc(\vec{R})$ implies that of $Acc(T)$. Moreover, if two trees T, T' have bisimilar retractions R, R' with respect to a given automaton M, then $T \in \mathscr{L}(M)$ iff $T' \in \mathscr{L}(M')$.

Theorem 1. *Let T be a B-augmented tree, M a B-augmented tree automaton, Π a factorization of T, R a retraction of T with respect to M and Π, and $\vec{R}$ the encoding of R. One can compute an M-type of T from a given $\overrightarrow{M}$-type of $\vec{R}$ and thus we have that $T \in \mathscr{L}(M)$ iff $\vec{R} \in \mathscr{L}(\overrightarrow{M})$.*

The proof is rather involved (see [10] for details). It exploits a suitable two-way correspondence between (the features of) the runs of M on T and (the features of) the runs of $\overrightarrow{M}$ on $\vec{R}$. Roughly speaking, from the definition of the automaton $\overrightarrow{M}$, we have that, for every run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$, one can build a corresponding run P of M on T, and, conversely, for every run P of M on T, one can build a run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ and a corresponding run P' of M on T such that P is subsumed by P'. This allows one to define a complete set of runs of M on T on the grounds of a complete set of runs of $\overrightarrow{M}$ on $\vec{R}$.

We conclude the section with a simple example of application of Theorem 1. Let w be an infinite word over an alphabet C. It can be thought of as an expanded linear structure $(\mathbb{N}^+, E, (V_c)_{c \in C})$, where $(i, j) \in E$ iff $j = i + 1$ and $i \in V_c$ iff $w[i] = c$. We denote by T_w the infinite complete $\{1, 2\}$-labeled C-colored tree obtained by assigning color $w[i]$ to every vertex that belongs to the i-th level of the tree. Formally, $T_w(v) = w[|v| + 1]$ for every $v \in \{1, 2\}^*$ (see Figure 2). If the MSO theory of w is decidable, then that of T_w is decidable as well. This can be proved by showing that T_w is nothing but the unfolding of the graph G, labeled over a binary alphabet $\{1, 2\}$, which is obtained from w via the MSO-definable interpretation that introduces a new copy of each edge of w. By exploiting the MSO-compatibility of the unfolding operation, one can reduce the model checking problem for T_w to that for G, which can in its turn be reduced to that of w. Our method provides an alternative proof of the decidability of the MSO theory of T_w, which is independent from the MSO-compatibility of the unfolding operation. Let $B = \{b\}$ and let Π be the factorization of T_w with edges labeled over B (recall that T_w can be viewed as a B-augmented tree) such that $\mathcal{D}om(\Pi) = \mathcal{D}om(T_w)$. Given a tree automaton M running on T_w, we can turn it into an equivalent B-augmented tree automaton by letting the third component of the acceptance condition be empty (i.e., $\mathcal{B} = \emptyset$). We then define a retraction R_w of T_w with respect to M and Π as follows. Let $\mathcal{D}om(R_w) = \mathcal{D}om(\Pi)$ and, for every $v \in \mathcal{D}om(R_w)$, let $R_w(v)$ be the basic M-type of the marked factor of T_w in v with respect to Π (i.e., the B-augmented tree $c\langle b, b\rangle$, where $c = w[|v| + 1]$). Let Ω be the (computable) function that maps each symbol $c \in C$ to the basic M-type of the B-augmented tree $c\langle b, b\rangle$. It can be easily shown that R_w is bisimilar to the linear structure $\vec{R}_w = \Omega(w)$. As a matter of fact, the structure $\vec{R}_w$ can be obtained from w via an MSO-definable interpretation that replaces every color $c \in C$ with the corresponding basic M-type $\Omega(c)$. Thus, by Theorem 1, the decidability of the acceptance problem for T_w (and hence the decidability of the MSO theory of T_w) follows immediately from the decidability of $Acc(w)$ (hence from the decidability of the MSO theory of w).

Even though tree automata are intimately related to MSO formulas evaluated over deterministic trees, we do not know whether the above result can be obtained by means of Shelah's composition method [11]. Precisely, we found it difficult to directly map (without using MSO-compatibility of the unfolding operation) formulas over T_w to equivalent formulas over a memoryless recoloring of w.

4 The class of reducible trees

In this section we define the class of reducible trees, which properly includes that of regular trees and whose elements enjoy decidable MSO theories. Intuitively, the decidability of the MSO theories of reducible trees follows

[1] Even in the case in which t' is the *basic* $\overrightarrow{M}$-type of $\vec{R}$, we cannot guarantee the computed t to be the *basic* M-type of T. This further explains the need for the notions of *subsumed* run and *complete* set of runs.

from the possibility of recursively reducing their acceptance problem to that for retractions of them. In addition, we prove that the class of reducible trees is closed with respect to various natural operations. These results, besides showing the robustness of the class of reducible trees, provide a neat framework to reason on retractions of trees and to easily transfer decidability results.

Reducible trees are inductively defined as follows.

Definition 6. Any regular tree is a *rank* 0 *tree*. Given a tree T and a natural number $n > 0$, T is a (B-augmented) *rank* n *tree* if, for every (B-augmented) tree automaton M, there exist a factorization Π of T and a retraction R of T with respect to M and Π such that the encoding of R is a rank $n-1$ tree. A *reducible tree* is a *rank* n *tree* for some $n \geq 0$.

According to Definition 6, the decidability of the MSO theory of a reducible tree T follows from Theorem 1 and from the decidability of the acceptance problem for regular trees, provided that there exists an effective way to compute (a representation) of the encoding of a retraction of T with respect to any given automaton M. Let the *footprint* of a tree T be the minimum amount of information that should be provided to make the reduction from T to its retraction feasible. Such a footprint can be inductively defined as follows. Given a B-augmented rank 0 tree T, a footprint of T is any finite rooted $B \cup C$-colored graph, whose unfolding is isomorphic to T. Given a B-augmented rank $n > 0$ tree T, a footprint of T is any computable function ξ that maps a B-augmented tree automaton M to a set $B' \supseteq B$ and a footprint of a rank $n-1$ B'-labeled tree $\vec{R}$ which encodes a retraction of T with respect to M. Hereafter, we restrict ourselves to reducible trees which are modeled according to any suitable (internal or external) representation system that allows the computation of their footprints. Under such a restriction, we have the following result.

Theorem 2. *Reducible trees enjoy a decidable acceptance problem.*

The inductive definition of rank n tree induces a hierarchical structure on the class of reducible trees. Establishing whether or not such a hierarchy is *strictly increasing*, namely, whether or not, for every $n > 0$, the class of rank n trees properly includes the class of rank $n-1$ trees, is an open problem.

It is possible to prove that the class of reducible trees properly includes that of residually ultimately periodic trees introduced in [8]. Such an inclusion reflects the generality of the notion of factorization proposed in this paper, which allows reductions towards branching structures, instead of linear ones. We may distinguish between the two notions of factorizations by defining *linear factorization* any factorization that satisfies the definition given in [8]. One can view a linear factorization of a tree T as a factorization Π of T (according to Definition 3), which satisfies the following property: for every pair of paths π and π' in Π, labeled, respectively, by words u and u', having the same length and terminating with the same symbol, the marked factor of T in u and the marked factor of T in u' coincide. Moreover, the definition proposed in [8] associates with each linear factorization an ordinal n, called *level*, which takes into account the form of its factors. For instance, all factors in a level 1 linear factorization must be regular trees, while those in a level 2 linear factorization must be decomposable into level 1 linear factorizations. Consider now the A-labeled C-colored tree T depicted in Figure 3, where $A = \{a, b, c\}$, $C = \{0, 1\}$, $\mathcal{D}om(T) = \{a, b\}^* \cup \{a, b\}^* \cdot \{c\} \cdot \{a, b\}^*$, and, for every $v \in \mathcal{D}om(T)$, $T(v) = 1$ iff $v = u \cdot c \cdot u$ for some $u \in \{a, b\}^*$. Clearly, all subtrees of T rooted at the vertices $u \in \{a, b\}^*$ are two-by-two non-isomorphic. Due to such a particular structure, there exists not a linear factorization of T of any level. However, by exploiting closure properties of rank n trees (see Section 4.1), one can easily prove that T is a rank 1 tree.

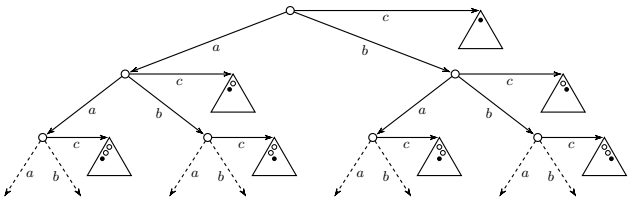

Figure 3. A reducible tree which is not residually ultimately periodic.

4.1 Closure properties of reducible trees

In this section we prove several closure properties for the class of reducible trees, which are based on compositional properties of M-types. We say that the class of rank n trees (resp., reducible trees) is effectively closed under a family $\mathscr{F}$ of operations on trees whenever the application of any transformation $t \in \mathscr{F}$ results in a tree whose footprint is computable on the grounds of the footprint of the input tree. In the following, we show that reducible trees are closed under (suitable variants of) three powerful operations on trees, namely, *finite-state recolorings*, *second-order tree substitutions*, and *top-down deterministic tree transducers*.

As for the first operation, we distinguish among three different notions of recoloring: finite-state recoloring *without lookahead* (i.e., the output of a Mealy tree automaton working in top-down fashion on an input colored tree), finite-state recoloring *with bounded lookahead* (which allows the inspection of the subtree rooted at the current position up to a bounded depth and makes transitions dependent on that portion of the subtree), and finite-state recoloring

with rational lookahead (which allows the inspection of the whole subtree rooted at the current position and makes transitions dependent on the subtree classification induced by a given finite class of rational tree languages).

A second-order tree substitution of the form $T[\![F_c]\!]_{c\in C}$ replaces all c-colored vertices in the tree T by a new tree F_c, simultaneously for all colors $c \in C$. The subtrees rooted at the 1-st, 2-nd, ..., k-th successor of a replaced c-colored vertex are possibly attached to the replacing tree F_c as follows: we mark the leaves of F_c with elements from A, which act as placeholders for the subtrees to be attached, making every replacing tree an A-augmented tree. A second-order tree substitution can be viewed either as a function σ, specified by an m-tuple of replacing trees $F_1, ..., F_m$, with $C = \{c_1, ..., c_m\}$, which maps a tree T to the tree $T[\![F_1/c_1, ..., F_m/c_m]\!]$, or as a function γ, specified by a tree T and by an n-tuple of colors c_{i_1}, ..., c_{i_n}, with $1 \leq n \leq m$, which maps the n-tuple $(F_1, ..., F_n)$ to $T[\![F_1/c_{i_1}, ..., F_n/c_{i_n}]\!]$. These two views of a second-order tree substitution give rise to the notions of tree morphism and tree insertion, respectively. We say that a tree morphism (resp., a tree insertion) is regular if the trees $F_1, ..., F_m$ are regular (resp., the tree T is regular).

Top-down deterministic tree transducers are finite-state machines that process a tree in a top-down fashion and replace the vertex in the current position with a regular tree, which may depend on the current state and on the color of the current vertex. At each computation step, different states can be spread among different (copies of the) successors of the current vertex. Like finite-state recolorings, tree transducers can be enriched with the facility of bounded/rational lookahead.

Theorem 3. *For every $n \in \mathbb{N}$, the class of rank n trees is effectively closed under regular tree morphisms and finite-state recolorings with bounded lookahead.*

We give an intuitive account of the proof of Theorem 3 (see [10] for more details). Let $\mathscr{F}$ be the set of involved transformations. If $n = 0$, we can easily show that the class of regular trees is closed under any transformation in $\mathscr{F}$. As for the inductive step, we fix a rank n tree T, with $n > 0$, a transformation $t \in \mathscr{F}$ mapping T to $t(T)$, and a tree automaton M running on $t(T)$. Then, one can show that there are a suitable tree automaton M' running on T, a rank $n-1$ tree $\vec{R}$ encoding a retraction of T with respect to M', and a transformation $t' \in \mathscr{F}$ mapping $\vec{R}$ to a tree $t'(\vec{R})$, such that $t'(\vec{R})$ encodes a retraction of $t(T)$ with respect to M. By exploiting the inductive hypothesis, it turns out that $t'(\vec{R})$ is a rank $n-1$ tree and hence $t(T)$ is a rank n tree. Such a result allows one to interpret MSO-compatibility of several tree transformations in terms of reducibility. As a matter of fact, we believe that the proof of the above theorem can be generalized to include the case of finite-state recolorings with *rational* lookahead.

The tree transformations we described so far are not independent. Indeed, it is possible to show that finite-state recolorings without lookahead (resp., with bounded, rational lookahead) together with regular tree morphisms subsume deterministic top-down tree transducers without lookahead (resp., with bounded, rational lookahead). More precisely, given a tree transducer N without lookahead (resp., with bounded, rational lookahead), the output of N on a tree T can be obtained by first applying to T a regular tree morphism, then a finite state-recoloring without lookahead (resp., with bounded, rational lookahead), and finally another regular tree morphism. Conversely, both finite-state recolorings without lookahead (resp., with bounded, rational lookahead) and regular tree morphisms can be thought of as special cases of tree transducers without lookahead (resp., with bounded, rational lookahead). Taking advantage of such relationships, we can exploit Theorem 3 to prove that the class of reducible trees is effectively closed with respect to top-down deterministic tree transducers with bounded lookahead.

As for tree insertions, we have the following result. By a slight abuse of terminology, given an A-augmented tree automaton and an n-tuple of A-augmented trees $\bar{F} = (F_1, ..., F_n)$, we say that $\bar{t} = (t_1, ..., t_n)$ is an M-type of $\bar{F}$ if for every $1 \leq i \leq n$, t_i is an M-type of F_i.

Theorem 4. *Let M be an A-augmented tree automaton and γ a regular tree insertion. Given a tuple $\bar{t}$ of M-types, one can compute an M-type t' such that, for every tuple $\bar{F}$ of A-augmented trees, if $\bar{t}$ is an M-type of $\bar{F}$, then t' is an M-type of $\gamma(\bar{F})$.*

Theorem 4 implies that for every regular tree insertion γ and every automaton M, there is a computable function $\gamma^M : \mathscr{T}_M^n \rightarrow \mathscr{T}_M$ that maps an M-type of an n-tuple of trees $\bar{F}$ to an M-type of the tree $\gamma(\bar{F})$. The function γ^M is called an *abstraction* of the regular tree insertion γ. Moreover, if $n = 1$, the pair $(\mathscr{A}_M, \circ)$ is a *finite monoid*, where $\mathscr{A}_M$ is the set of all abstractions of regular tree insertions and $\circ$ is the operation of functional composition. Indeed, we have that (i) $\mathscr{A}_M$ is a finite set (since there are only finitely many M-types in $\mathscr{T}_M$), (ii) tree insertions are closed under functional composition (this follows from associativity of substitutions), (iii) abstractions of regular tree insertions are closed under functional composition (since regular tree insertions map regular trees into regular trees), and (iv) there exists an abstraction id_M playing the role of the identity in $\mathscr{A}_M$.

Finally, it is possible to show that reducible trees are closed under the operation of *unfolding with backward edges and loops*, denoted $\mathcal{B}ack\mathcal{U}nf$. Such an operation maps an A-labeled tree T to the unfolding of the rooted

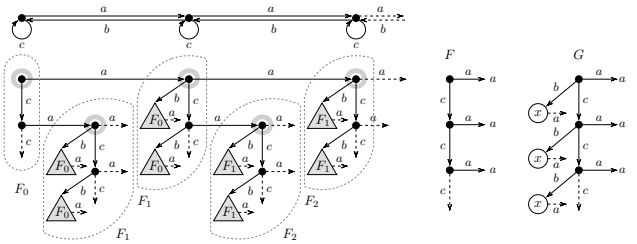

Figure 4. Unfolding of the semi-infinite line.

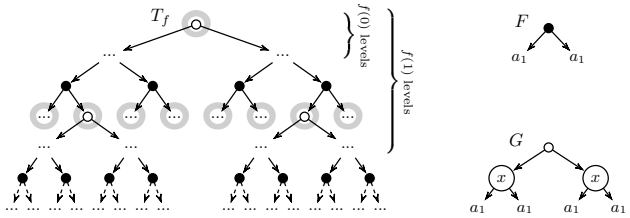

Figure 5. An example of reducible tree outside the Caucal hierarchy.

graph resulting from T after the addition of backward $\bar{A}$-labeled edges and $\#$-labeled loops (cf. [1]).

Theorem 5. *For every rank n tree T, $\mathcal{B}ack\mathcal{U}nf(T)$ is a rank $n+1$ tree, and thus the class of reducible trees is effectively closed under $\mathcal{B}ack\mathcal{U}nf$.*

4.2 Examples of reducible trees

The class of reducible trees obviously includes all regular trees; moreover, it includes a number of non-regular ones, such as, for instance, all deterministic trees in the first two levels of the Caucal hierarchy and several deterministic trees outside it. Here, we provide two noticeable examples of reducible trees, which should explain how the previously disclosed closure properties can be used. A number of other meaningful examples can be found in [10].

To start with, we consider the well-known example of the semi-infinite line. Let $L = (\mathbb{N}, E_a, E_b, E_c)$ be the semi-infinite line with a-labeled forward edges, b-labeled backward edges, and c-labeled loops (see the top part of Figure 4). Let T_L be the unfolding of L from the leftmost vertex. The bottom left part of Figure 4 depicts the tree T_L, where, for each $i \in \mathbb{N}$, F_i denotes the unfolding from the rightmost vertex of the subgraph L_i obtained by restricting L to set of vertices $\{0, ..., i-1\}$. We give an alternative proof of the decidability of the MSO theory of T_L, which exploits the closure properties of reducible trees instead of the MSO-compatibility of the unfolding operation. The idea is to give an inductive definition of the components $F_0, F_1, F_2, ...$, which allows us to prove that T_L is a rank 1 tree. By construction, every vertex v of T_L corresponds to a *unique* path π_v in L. We denote by $\overrightarrow{\pi}_v$ the last vertex of L along the path π_v and we define the factorization Π of T with respect to $B = \{a\}$ by letting $\mathcal{D}om(\Pi)$ be the set of all vertices v of T_L such that there is no a proper ancestor v' of v for which $\overrightarrow{\pi}_v = \overrightarrow{\pi}_{v'}$ (the set $\mathcal{D}om(\Pi)$ is represented in Figure 4 by circled nodes). Then, we label the resulting edges of Π with the symbol a. Notice that Π has unbounded degree. However, for every pair of vertices u, u' in Π, if the access paths of u and u' in Π have the same length, then the marked factor of T_L in u and the marked factor of T_L in u' turn out to be isomorphic. This means that we can identify access paths in Π having the same length, thus showing that, for any given B-augmented tree automaton M, there is a suitable retraction of T_L, with respect to Π and M, which is bisimilar to a deterministic B-labeled tree. Such a retraction can be obtained as follows. Let F and G be the two B-augmented trees depicted in the right part of Figure 4 and let γ be the regular tree insertion specified by G (the vertices that must be replaced with the input of γ are colored with x). Then, we set $F_i = \gamma^i(F)$, for every $i \in \mathbb{N}$. It is easy to see that, for every vertex u of Π at distance i from the root, the marked factor $T^+_\Pi[u]$ of T_L in u is isomorphic to the tree F_i. Thus, by Theorem 4, the tree $\overrightarrow{R}_L$ such that (i) $\mathcal{D}om(\overrightarrow{R}_L) = B^*$, (ii) $\overrightarrow{R}_L(\varepsilon)$ is the basic M-type of F_0, and (iii) $\overrightarrow{R}_L(a^{i+1}) = \gamma^M(R_L(a^i))$, for all $i \in \mathbb{N}$, encodes a retraction of T_L with respect to M and Π. Since $\mathscr{T}_M$ is a finite set, from the Pigeonhole Principle it follows that $\overrightarrow{R}_L$ is a regular tree. In fact, we just showed that the encoding $\overrightarrow{R}_L$ of a retraction of T_L can be obtained via a finite-state recoloring of the semi-infinite line, which is a rank 0 tree. By Theorem 3, this implies that T_L is a rank 1 tree, whose footprint can be effectively built. As a matter of fact, such a result can be viewed as a trivial implication of Theorem 5: T_L is nothing but the tree resulting from the application of $\mathcal{B}ack\mathcal{U}nf$ to the semi-infinite line. Not surprisingly, the proof technique used in this example can be applied to any rank n tree, thus proving that reducible trees are closed under $\mathcal{B}ack\mathcal{U}nf$.

We now provide an example of a reducible tree that does not belong to the Caucal hierarchy. Let f be a *strictly monotone* function over $\mathbb{N}$, that is, $i < j$ implies $f(i) < f(j)$ for every $i, j \in \mathbb{N}$, and g be the function defined by $g(0) = f(0)$ and $g(i+1) = f(i+1) - f(i) - 1$, for all $i \in \mathbb{N}$. Moreover, let $A = \{a_1, a_2\}$ and $C = \{0, 1\}$. We denote by T_f the A-labeled C-colored tree obtained from the infinite complete A-labeled $\{0\}$-colored tree by recoloring every vertex at distance $f(i)$ from the root, for $i \in \mathbb{N}$, with 1 (see the left part of Figure 5, where we represent 0-colored vertices with white nodes and 1-colored vertices with black nodes). We define a factorization Π of T_f with respect to $B = \{a_1\}$ by letting $\mathcal{D}om(\Pi)$ be the set including the root and all successors of 1-colored vertices and by

labeling the resulting edges of Π with a_1. Even though Π has unbounded degree, by identifying access paths with the same length, we can obtain, for every B-augmented tree automaton M, a retraction of T_L, with respect to Π and M, which is bisimilar to a deterministic tree. Let F and G be the two A-augmented trees depicted in the right part of Figure 5 and let γ be the regular tree insertion specified by G (the vertices of G that must be replaced with the input of γ are identified by the color x). Then, we set $F_i = \gamma^{g(i)}(F)$, for every $i \in \mathbb{N}$. It is easy to see that, for every vertex u of Π at distance $i \in \mathbb{N}$ from the root, the marked factor of T_f in u is isomorphic to the A-augmented tree F_i. Moreover, if γ^M is the abstraction of the regular tree insertion γ with respect to M and t is the basic M-type of F, then for every $i \in \mathbb{N}$, $(\gamma^M)^{g(i)}(t)$ is an M-type of F_i. This implies that the deterministic tree $\vec{R}_f$ defined by $\mathcal{D}om(\vec{R}_f) = B^*$ and $\vec{R}_f(a_1^i) = (\gamma^M)^{g(i)}(t)$, for all $i \in \mathbb{N}$, is bisimilar to a retraction of T_f with respect to Π and M. One can show that $\vec{R}_f$ is a *regular* tree, provided that g is *ultimately periodic with respect to finite monoids* (see [10] for definitions and proofs). Under such an hypothesis, the tree T_f turns out to be a rank 1 tree and hence it enjoys a decidable MSO theory. As an example, the hypothesis holds if f is the tower of exponentials tow, defined by $tow(0) = 1$ and $tow(i+1) = 2^{tow(i)}$. In such a case, the resulting tree T_f can be easily proved to be outside the Caucal hierarchy.

4.3 Relationships with Caucal hierarchy

In [2], Carayol and Wöhrle show that, for each level of the Caucal hierarchy, there exists a representative graph, called *generator*, from which all other graphs belonging to that level can be obtained via MSO-definable interpretations. For a given $n \in \mathbb{N}$, the generator G_n for the $n+1$-th level of the Caucal hierarchy is defined as the n-fold application of the treegraph operation to the infinite binary complete tree. These generators are closely related to another family of trees that Cachat introduces to simulate games on higher order pushdown systems [1]. These trees, denoted by $C_0, C_1, \ldots$, are obtained from the infinite complete binary tree via n-fold applications of $\mathcal{B}ack\mathcal{U}nf$. It can be proved that each generator G_n can be obtained from the tree C_n via a suitable MSO-definable interpretation that first restricts the domain to the vertices/words that do not contain occurrences of $a \cdot \bar{a}$ or $\bar{a} \cdot a$ (where a denotes a forward edge and $\bar{a}$ the corresponding backward edge introduced by the $\mathcal{B}ack\mathcal{U}nf$ operation) and then reverses the $\bar{a}$-labeled edges. It follows that also Cachat trees are generators of the graphs of the Caucal hierarchy via MSO-definable interpretations. Here, we define another class of tree generators, which contains all Cachat trees and that generates all *deterministic* trees in the Caucal hierarchy via inverse rational mappings, which are special cases of MSO-definable interpretations.

Definition 7. A *level* 0 *tree generator* is a regular tree. For every $n \in \mathbb{N}$, a *level* $n+1$ *tree generator* is a tree of the form $T' = \mathcal{B}ack\mathcal{U}nf(T)$, where T is a level n tree generator.

Theorem 5 immediately implies that level n tree generators are rank n trees.

Consider now unfoldings of graphs obtained from trees via inverse rational mappings [4]. Intuitively, the application of an inverse rational mapping to a tree T results in a graph G, whose domain coincides with that of T and where (u, v) is an a'-labeled edge in G iff there exists a path from u to v in T labeled with a word in a designated rational language $h(a')$. In the general case, a path can be empty or can traverse edges in either direction. Given a set A of labels, we denote by ε the empty path and by a (resp., $\bar{a}$) a path that traverses a single a-labeled edge in forward (resp., backward) direction, for any $a \in A$. The application of the inverse of a rational mapping of the form $h : A' \rightarrow \mathscr{P}\big((A \cup \bar{A})^*\big)$ to an A-labeled graph (or tree) produces an A'-labeled graph. Let us consider now two special forms of rational mapping, namely, *A-flip mapping* and *A-forward mapping*. The former one is defined as the (unique) finite mapping $h_A : A' \rightarrow \mathscr{P}\big((A \cup \bar{A})^*\big)$ such that (i) $A' = A \cup \bar{A} \cup \{\#\}$, (ii) $h(a) = \{a\}$ for all $a \in A$, (iii) $h(\bar{a}) = \{\bar{a}\}$ for all $\bar{a} \in \bar{A}$, and (iv) $h(\#) = \{\varepsilon\}$. Intuitively, h_A^{-1} extends an input (A-labeled) tree T with ($\bar{A}$-labeled) backward edges and (#-labeled) loops. Clearly, for every A-labeled tree T, $\mathcal{B}ack\mathcal{U}nf(T)$ coincides with the unfolding of the rooted graph $h_A^{-1}(T)$. As for the other mapping, an A-forward mapping is any rational mapping h such that $h(a') \subseteq A^+$ for all $a' \in A'$, namely, h defines regular path expressions by using labels of forward edges only (neither backward edges nor loops are allowed). Note that, for every A-forward mapping $h : A' \rightarrow \mathscr{P}(A^+)$ and for every A-labeled tree T, $h^{-1}(T)$ is a (possibly nondeterministic) A'-labeled tree. Moreover, the operations of inverse forward mapping h^{-1} and unfolding $\mathcal{U}nf$ over rooted graphs commute up to bisimulation, namely, for every A-forward mapping h and for every A-labeled rooted graph G, the two trees $\mathcal{U}nf\big(h^{-1}(G)\big)$ and $h^{-1}\big(\mathcal{U}nf(G)\big)$ are bisimilar. Finally, it turns out that any inverse rational mapping $h : A' \rightarrow \mathscr{P}\big((A \cup \bar{A})^*\big)$ can be decomposed into an inverse A-flip mapping followed by an inverse $A \cup \bar{A} \cup \{\#\}$-forward mapping. These properties allow us to state the following result.

Proposition 6. *For every rational mapping $h : A' \rightarrow \mathscr{P}\big((A \cup \bar{A})^*\big)$, there is a rational B-forward mapping $\bar{h} : A' \rightarrow \mathscr{P}(B^+)$, with $B = A \cup \bar{A} \cup \{\#\}$, such that, for every A-labeled tree T, the two trees $\mathcal{U}nf\big(h^{-1}(T)\big)$ and $\bar{h}^{-1}\big(\mathcal{B}ack\mathcal{U}nf(T)\big)$ are bisimilar. Moreover, if both trees are deterministic, then they are isomorphic.*

In virtue of Proposition 6 and Definition 7, we have that tree generators together with inverse rational forward mappings

suffice to generate all deterministic trees in the Caucal hierarchy. Since inverse rational forward mappings are special cases of MSO-definable interpretations that preserve bisimilarity of graphs, by a result of Colcombet and Löding [5], it follows that any inverse rational forward mapping can be implemented by a tree transducer with rational lookahead. Moreover, since tree transducers with rational lookahead are subsumed by finite-state recolorings with rational lookahead and regular tree morphisms, if the class of rank n trees were closed under finite-state recolorings with rational lookahead (as we expect), then the class of reducible trees would capture *all* deterministic trees in the Caucal hierarchy. As a matter of fact, we already know that rank n trees are closed under finite-state recolorings with *bounded* lookahead. Hence, since inverse *finite* forward mapping can be implemented by tree transducers with bounded lookahead, we have that reducible trees capture all deterministic trees obtained by iterating unfoldings and inverse finite mappings, starting from regular trees. Such a class of trees is properly included, starting from level 3, in the Caucal hierarchy (as an example, the tree depicted in Figure 3 belongs to the 3-rd level of the Caucal hierarchy and it cannot be obtained via inverse finite mappings and unfoldings starting from regular trees).

5 Conclusions

In this paper, we developed an automaton-based method to decide MSO theories of deterministic tree structures. By exploiting a suitable notion of tree indistinguishability with respect to tree automata, we showed that the acceptance problem for Rabin/Muller tree automata can be used to decide a large class of deterministic trees, called reducible trees, which includes many non-regular ones. Furthermore, we proved that such a class is closed with respect to various natural operations on trees, including finite-state recolorings with bounded lookahead, regular tree morphisms, and unfoldings with backward edges and loops. We are currently investigating the closure of the class of reducible trees under finite-state recolorings with rational lookahead, whose validity would imply that the class of reducible trees includes all deterministic trees in the Caucal hierarchy. We are also trying to extend the notion of reducible tree to capture the trees generated by the so-called higher-order recursive program schemes [6] (the MSO theories of trees generated by unsafe higher-order recursive program schemes have been recently proved to be decidable by Ong [9]). Finally, we are investigating the relationships between our automaton-based method and Shelah's composition one [11]. We believe it possible to generalize both of them to deal with non-deterministic colored trees as well as with generic relational structures.

Acknowledgements

We are grateful to the anonymous referees for their useful remarks. The work has been funded by the bilateral project "Temporal logics in computer and information sciences", supported by the Italian Ministero degli Affari Esteri and the National Research Foundation of South Africa, under the Joint Italy/South Africa Science and Technology Agreement, and by the Italian PRIN project on "Constraints and preferences as a unifying formalism for system analysis and solution of real-life problems".

References

[1] T. Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Proc. of the 30th International Colloquium on Automata, Languages, and Programming*, volume 2719 of *LNCS*, pages 556–569. Springer, 2003.

[2] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *LNCS*, pages 112–123. Springer, 2003.

[3] O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. *Information and Computation*, 176(1):51–65, 2002.

[4] D. Caucal. On infinite terms having a decidable monadic theory. In *Proc. of the 27th International Symposium on Mathematical Foundations of Computer Science*, volume 2420 of *LNCS*, pages 165–176. Springer, 2002.

[5] T. Colcombet and C. Löding. On the expressiveness of deterministic transducers over infinite trees. In *Proc. of the 21st Annual Symposium on Theoretical Aspects of Computer Science*, volume 2996 of *LNCS*, pages 428–439. Springer, 2004.

[6] W. Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.

[7] C. Elgot and M. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, 1966.

[8] A. Montanari and G. Puppis. Decidability of MSO theories of tree structures. In *Proc. of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 3328 of *LNCS*, pages 430–442. Springer, 2004.

[9] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. of the 21st Symposium on Logic in Computer Science*, pages 81–90. IEEE Computer Society, 2006.

[10] G. Puppis. *Automata for Branching and Layered Temporal Structures*. PhD thesis, Dipartimento di Matematica e Informatica, Università di Udine, Italy, 2006.

[11] S. Shelah. The monadic theory of order. *Annals of Mathematics*, 102:379–419, 1975.

[12] W. Thomas. Languages, automata, and logic. In G. Rozemberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.