A Contraction Method to Decide MSO Theory of Deterministic Trees

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. First, we reduce the model checking problem for MSO logic over deterministic colored trees to the acceptance problem for tree automata. Then, we exploit a suitable notion of indistinguishability of trees with respect to tree automata to reduce a number of instances of the latter problem to decidable instances over 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 [14], 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 [14]. 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, 11] 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 acceptance problem for Rabin/Muller tree automata, that is, to the problem of deciding whether a Rabin/Muller tree automaton accepts a given relational structure viewed as a deterministic vertex-colored tree. Such a problem is 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 [12, 13]. 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 neeeds to reduce a branching structure to a linear one (see the example in Section 3.2). Detailed definitions and proofs can be found in [10] (some of them are reported in the appendix).

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* (directed simple) graph is a tuple $G = (D, (E_a)_{a\in A})$, where D (also denoted $\mathcal{D}om(G)$) is a countable 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 a tuple $\bar{V}$ of unary predicates, namely, a structure of the form $(G, \bar{V})$, where $\bar{V} = (V_1, ..., V_m)$ and $V_i \subseteq \mathcal{D}om(G)$ 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 $v \in V_i$. We focus our attention on graphs such that for every vertex v, there exists a unique path, called *access path*, from a designated source vertex, called *root*, to v (unranked rooted trees). 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 as a partial function T 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 tree automata, rather than Rabin ones, to simplify definitions and proofs. 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 states, defining the acceptance condition. 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 [14] establishes a strong correspondence between MSO formulas satisfied by an expanded tree structure $(T, \bar{V})$ 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 $T \models \varphi[\bar{V}]$ iff $T_{\bar{V}} \in \mathscr{L}(M)$. Let us call *acceptance problem* of a given 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 tree structures, including non-regular trees.

3.1 Indistinguishability of trees

Here we introduce the basic ingredients of the automaton-based approach to the decidability of MSO theories of trees. 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 non-complete, 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 $\mathcal{B}ch(P)$ denotes the set of infinite paths in P originating from the root and $\mathcal{F}r(P)$ denotes the set of P leaves.

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 third component identifies, for each 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 cope with empty or non-full trees, we extend the input alphabet of the automaton M with a fresh symbol $\perp \notin B \cup C$ and we assume that M reads $\perp$ on the missing successors, as well as on their descendants, of an internal vertex. More precisely, we introduce a suitable operation, called *completion*, which extends a B-augmented tree T with $\perp$-colored vertices. 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 $v \in F \cdot A^*$. Intuitively, $T_\perp$ is obtained from T by appending infinite complete $\perp$-colored trees to every missing successor of an internal vertex. Notice that $T_\perp$ is a non-empty, full, B-augmented tree. Such an operation 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

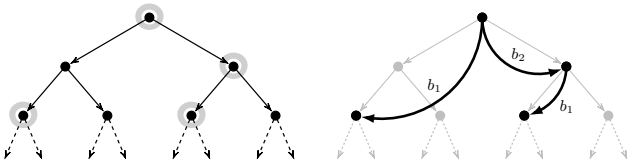

Figure 1. An example of factorization.

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.

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) $\mathcal{D}om(\Pi) \subseteq \mathcal{D}om(T)$, (ii) $\varepsilon \in \mathcal{D}om(\Pi)$, (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.

Note that a B-augmented tree T can be viewed as a B'-augmented tree, for any $B' \supseteq B$. This allows one to use arbitrarily large sets B' for the edge labels of a factorization. From now on, if not explicitly mentioned, we shall assume that the set of edge labels of a factorization of a B-augmented tree T is exactly the set B. We can graphically represent a factorization of a tree by first identifying its vertices (look at those nodes on the left-hand side tree of Figure 1 which are surrounded by gray circles) and then drawing the resulting edges, together with the corresponding labels (look at 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 in $T_\perp$ of u, but not proper descendants in $T_\perp$ of any $u' \in Succ(u)$. 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 and, moreover, it avoids trivial factorizations 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 identify a retraction R with a suitable infinite complete deterministic tree. For instance, this is possible 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 and by extending R with $\perp$-colored vertices, one can obtain an infinite complete 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. This encoding of a retraction is unique up to isomorphisms and it can be provided in input to a suitable tree automaton. Moreover, by definition, retractions depend on automata, but, as a matter a fact, for all the considered tree structures, we shall provide a single factorization from which we will be able to generate a corresponding retraction for every 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. The automaton $\overrightarrow{M}$ 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 information processed by M during its computations up to a certain point; its transitions compute the new states which extend information provided by the current state with information provided by the input symbol (i.e., the M-type of a marked factor or $\perp$). The automaton $\overrightarrow{M}$ 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 $(q_b)_{b \in B} \in Q^B$, $\big(q, \perp, (q_b)_{b \in B}\big) \in \delta'$ 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, J are suitable sets of indices, for 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 set of indices L, G, and for every tuple $(q_b)_{b \in B} \in Q^B$, $\big(q, t, (q_b)_{b \in B}\big) \in \delta'$ iff for all $g \in G$, there exists $h_g \in H$ such that (i) $r_g = s_{h_g}$ and (ii) $q_b = (b, 0, \mathcal{U}_b, \mathcal{V}_b)$, with $\mathcal{U}_b = \mathcal{U} \cup \{X_{h_g,i}\}_{g \in G, i \in I}$ and $\mathcal{V}_b = \{(s_{h_l,i}, Y_{h_g,j}, W_l \cup Y_{h_g,j})\}_{g \in G, j \in J, b_j = b}$;

- $\mathcal{I}'$ 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 of the form $\{q_1, ..., q_n\}$ such that (i) $q_1 = (d_1, x, \mathcal{U}, \mathcal{V}_1)$, ..., $q_n = (d_n, x, \mathcal{U}, \mathcal{V}_n)$, (ii) if $x = 1$, then $n = 1$, $\mathcal{V}_1 = \big\{(r_g, V_g, W_g)\big\}_{g \in G}$, and for all $g \in G$, $(d_1, r_g, W_g) \in \mathcal{B}$, (iii) if $x = 0$ and $\mathcal{V}_k = \{(r_{k,g}, V_{k,g}, W_k)\}_{g \in G_k}$ for $k = 1, ..., n$ and for suitable sets of indices G_k, then $\emptyset \subsetneq G'_1 \subseteq G_1$, ..., $\emptyset \subsetneq G'_n \subseteq G_n$ implies $\bigcup_{1 \leq k \leq n, g \in G'_k} V_{k,g} \in \mathcal{F}$, and (iv) $\mathcal{W} \subseteq \mathcal{F}$.

The following main 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 (a sketch of the proof is given in the Appendix). It is worth to remark that, 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. As a corollary of Theorem 1, it turns out 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 $\vec{R} \in \mathscr{L}(\overrightarrow{M})$.*

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

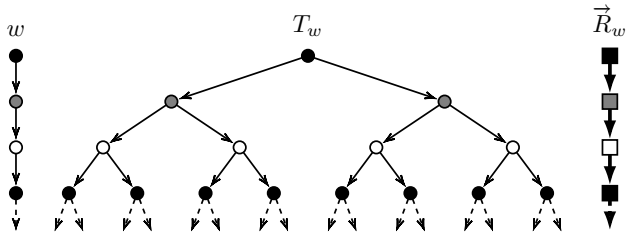

Figure 2. An application of Theorem 1.

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 $c = w[|v| + 1]$). If we denote by Ω the (computable) function that maps each symbol $c \in C$ to the basic M-type of the B-augmented tree $c\langle b, b\rangle$, then we easily see 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 $c \in C$ by 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 [12, 13]. 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 el-

ements enjoy decidable MSO theories. Roughly speaking, the decidability of the MSO theories of reducible trees follows 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 $\overrightarrow{R}$ 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. On the one hand, establishing whether or not such a hierarchy is *strictly increasing*, namely, whether, for every $n > 0$, the class of rank n trees properly includes the class of rank $n-1$ trees, is an open problem. On the other hand, it is possible to show that the class of reducible trees properly includes that of residually ultimately periodic trees introduced in [8]. To give an intuition, such an inclusion reflects the more general notion of factorization proposed in this paper, which allows reductions towards branching structures, instead of linear ones. We distinguish between the two notions of factorizations by defining *linear factorization* any factorization that

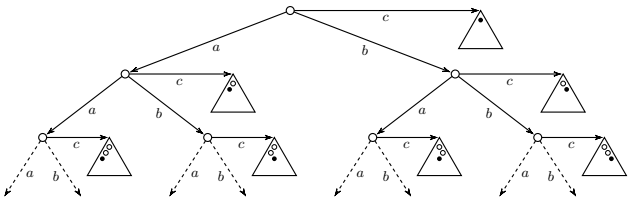

Figure 3. A reducible tree which is not residually ultimately periodic.

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 (this is needed to prove the decidability of the acceptance problem for the class of residually ultimately periodic trees). 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 $u \in \{0, 1\}^*$ are two-by-two non-isomorphic. Due to such a particular structure, T cannot have a linear factorization 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.

4.1 Closure properties of reducible trees

In this section we prove several closure properties for the class of reducible trees, which, roughly speaking, 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 automa-

ton 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. We can view any second-order tree substitution 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 latter 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.*

Roughly speaking, the proof of closure properties rests on the possibility of transferring the complexity of the tree resulting from the application of a transformation back to the automaton running on the original tree (a sketch of the proof is given in the Appendix). We expect that the proof of the closure property for rank n trees with respect to finite-state recolorings with bounded lookahead can be generalized to 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 applying to T first 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 $n = 1$, then we can denote by $\mathscr{A}_M$ the set of all abstractions of regular tree insertions and we can endow it with the operation $\circ$ of functional composition. It turns out that $(\mathscr{A}_M, \circ)$ is a *finite monoid*. 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 to regular trees), and (iv) there exists an abstraction id_M playing the role of the identity in $\mathscr{A}_M$.

We conclude by showing 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 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 $\mathcal{B}ack\mathcal{U}nf$.*

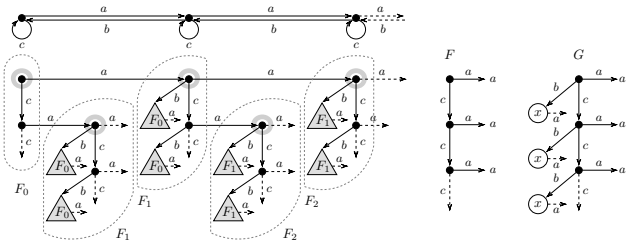

Figure 4. Unfolding of the semi-infinite line.

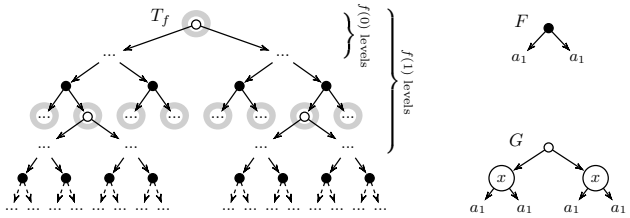

Figure 5. An example of reducible tree outside the Caucal hierarchy.

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 enlighten the reader on the use of previously disclosed closure properties. 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-alabeled 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 substituted 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 also 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 also viewed as a trivial implication of Theorem 5: indeed, 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 reducible tree outside the Caucal hierarchy. Let f be a *strictly monotone* function over $\mathbb{N}$, namely, such that $i < j$ implies $f(i) < f(j)$ for every $i, j \in \mathbb{N}$, and let g be the function defined by $g(0) = f(0)$ and $g(i+1) = f(i+1) - f(i) - 1$, for all $i \in \mathbb{N}$. We set $A = \{a_1, a_2\}$ and $C = \{0, 1\}$ and we denote by T_f the A-labeled C-colored tree obtained from the infinite complete A-labeled $\{0\}$-colored tree by recoloring with 1 every vertex at distance $f(i)$ from the root, for $i \in \mathbb{N}$ (see the left part of Figure 5, where we represent 0-colored vertices with white nodes and 1-colored vertices with black nodes). We now define a factorization Π of T_f with respect to $B = \{a_1\}$ by letting $\mathcal{D}om(\Pi)$ be the set consisting of the root plus 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 Fig-

ure 5 and let γ be the regular tree insertion specified by G (the vertices of G that must be substituted 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 if 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. Moreover, 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 non-deterministic) 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 preserving 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. 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. First, we reduced the model-checking problem to the acceptance problem for Rabin/Muller tree automata and we showed that the latter is easily decidable for the class of regular colored trees. Then, by exploiting a suitable notion of tree indistinguishability with respect to tree automata, we showed that this reduction can actually be used to decide the class of 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 working at a generalization of these closure properties to cope with the case of finite-state recolorings with rational lookahead (its 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 [12, 13]. We believe it possible to generalize both of them to deal with non-deterministic colored trees as well as with generic relational structures.

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 (STACS)*, 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 (FSTTCS)*, 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 (LICS)*, 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. Available on: `http://www.dimi.uniud.it/~puppis/PhDThesis.pdf`.

[11] A. Rabinovich and W. Thomas. Decidable theories of the ordering of natural numbers with unary predicates. In *Proc. of the 15th European Conference on Computer Science Logic (CSL)*, volume 4207, pages 562–574. Springer, 2006.

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

[13] W. Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science*, volume 1261 of *LNCS*, pages 118–143. Springer, 1997.

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

Appendix

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 $\vec{R}$ (in particular, we have that $T \in \mathscr{L}(M)$ iff $\vec{R} \in \mathscr{L}(\overrightarrow{M})$).*

Proof sketch. The proof is rather involved and it requires some technical definitions and preliminary results about tree decompositions. Here we give an account of the basic ingredients of the proof and we refer the reader to [10] for further details. Let $M = (S, A, B, C \cup \{\bot\}, \delta, \mathcal{I}, \mathcal{F}, \mathcal{B})$ and let $\overrightarrow{M} = (Q, B, \mathscr{T}_M \cup \{\bot\}, \delta', \mathcal{I}', \mathcal{F}')$, as specified in Definition 5. The proof is based on a suitable two-way correspondence between (the features of) the runs of M on T (here for simplicity we assume that T is a non-empty full B-augmented tree, otherwise replace T by $T_\bot$ everywhere) and (the features of) the runs of $\overrightarrow{M}$ on the encoding $\vec{R}$ of R. More precisely, we say that a run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ *mimics* a run P of M on T iff the following conditions are satisfied:

C1) $\vec{P}(\varepsilon) = (c, 0, \mathcal{U}, \mathcal{V})$ with $c \in B, \mathcal{U} = \emptyset$, and $\mathcal{V} = \big\{(P(\varepsilon), \emptyset, \emptyset)\big\}$;

C2) for every leaf v in P, there is an infinite path π in $\vec{P}$, with $\mathcal{I}nf(\vec{P}|\pi) = \big\{(b, 1, \mathcal{U}, \mathcal{V})\big\}$ and $\mathcal{V} = \big\{(r_g, V_g, W_g)\big\}_{g \in G}$, and there is an index $g \in G$ such that $T(v) = b$, $P(v) = r_g$, and $\mathcal{I}mg(P|v) = W_g$;

C3) for every infinite path τ in P, one of the following conditions holds
   1. there is an infinite path π in $\vec{P}$ such that $\mathcal{I}nf(\vec{P}|\pi) = \big\{(b_k, 0, \mathcal{U}, \mathcal{V}_k)\big\}_{1 \leq k \leq n}$ and $\mathcal{V}_k = \big\{(r_{k,g}, V_{k,g}, W_k)\big\}_{g \in G_k}$ for all $1 \leq k \leq n$ (where $G_1, ..., G_k$ are finite sets of indices), and there are $\emptyset \subsetneq G'_1 \subseteq G_1$, ..., $\emptyset \subsetneq G'_n \subseteq G_n$ such that $\mathcal{I}nf(P|\tau) = \bigcup_{1 \leq k \leq n} \bigcup_{g \in G'_k} V_{k,g}$;
   2. there is an infinite path π in $\vec{P}$ such that $\mathcal{I}nf(\vec{P}|\pi) = \big\{(b_k, x, \mathcal{U}, \mathcal{V}_k)\big\}_{1 \leq k \leq n}$, $x \in \{0, 1\}$, and $\mathcal{U} = \big\{U_l\big\}_{l \in L}$, and there is an index $l \in L$ such that $\mathcal{I}nf(P|\tau) = U_l$;

C4) for every infinite path π in $\vec{P}$, one of the following conditions holds
   1. there are $b \in B, \mathcal{U} \subseteq \mathscr{P}(S)$, and $\mathcal{V} = \big\{(r_g, V_g, W_g)\big\}_{g \in G}$ (where G is a finite set of indices) such that $\mathcal{I}nf(\vec{P}|\pi) = \big\{(b, 1, \mathcal{U}, \mathcal{V})\big\}$ and for every index $g \in G$, there is a leaf v in P such that $T(v) = b$, $P(v) = r_g$, and $\mathcal{I}mg(P|v) = W_g$;
   2. there are $n > 0$, $b_1, ..., b_n \in B$, $\mathcal{U} \subseteq \mathscr{P}(S)$, $\mathcal{V}_1, ..., \mathcal{V}_n \subseteq S \times \mathscr{P}(S) \times \mathscr{P}(S)$ such that $\mathcal{V}_k = \big\{(r_{k,g}, V_{k,g}, W_k)\big\}_{g \in G_k}$ for all $1 \leq k \leq n$ (where $G_1, ..., G_k$ are finite sets of indices), $\mathcal{I}nf(\vec{P}|\pi) = \big\{(b_k, 0, \mathcal{U}, \mathcal{V}_k)\big\}_{1 \leq k \leq n}$, and, for every $\emptyset \subsetneq G'_1 \subseteq G_1$, ..., $\emptyset \subsetneq G'_n \subseteq G_n$, there is an infinite path τ in P such that $\mathcal{I}nf(P|\tau) = \bigcup_{1 \leq k \leq n} \bigcup_{g \in G'_k} V_{k,g}$;

C5) for every infinite path π in $\vec{P}$, if $\mathcal{I}nf(\vec{P}|\pi)$ is of the form $\big\{(b_k, x, \mathcal{U}, \mathcal{V}_k)\big\}_{1 \leq k \leq n}$, with $\mathcal{U} = \big\{U_l\big\}_{l \in L}$, then, for every $l \in L$, there is an infinite path τ in P such that $\mathcal{I}nf(P|\tau) = U_l$.

Intuitively, the above conditions guarantee that the feature $[T, P]$ is uniquely determined by the feature $[\vec{R}, \vec{P}]$.

Given the above notion of correspondence, one can prove the following properties:

P1) *Every run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ such that $\vec{P}(\varepsilon) = \big(d, 0, \{r, \emptyset, \emptyset\}, \emptyset\big)$ mimics a run P of M on T.*

   This can be proved by first extracting from $\vec{R}$ and $\vec{P}$ a tree-shaped arrangement of runs of M on the factors of T and then combining those runs to form a valid run of M on T which satisfies Conditions C1)–C5).

P2) *For every run P of M on T, there are (i) a run P' of M on T such that $P' \preceq P$ and (ii) a run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ that mimics P'.*

(Notice that there may not exists a run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ that directly mimics P.)

This requires a two-step construction. First, given the run P of M on T, one builds a run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ by simply choosing suitable transitions of $\overrightarrow{M}$ that match those of M on P. Then, by exploiting Property P1), one builds a new run P' of M on T which is mimicked by $\vec{P}$. Finally, one verifies that $P' \preceq P$.

P3) *Let P_1 and P_2 be runs of M on T and let $\vec{P}_1$ and $\vec{P}_2$ be runs of $\overrightarrow{M}$ on $\vec{R}$. If $\vec{P}_i$ mimics P_i for both $i = 1$ and for $i = 2$ and if $\vec{P}_1 \preceq \vec{P}_2$, then $P_1 \preceq P_2$.*

This easily follows from the definition of mimicking run and from the definition of the transitions of $\overrightarrow{M}$.

Finally, by exploiting Properties P1), P2), and P3), one can prove the main theorem. Let t' be an $\overrightarrow{M}$-type of $\vec{R}$ of the form

$$\left\{ \begin{pmatrix} \vec{P}(\varepsilon), \\ \{\mathcal{I}nf(\vec{P}|\pi) \,:\, \pi \in \mathcal{B}ch(\vec{P})\}, \\ \emptyset \end{pmatrix} \,:\, \vec{P} \in \vec{\mathcal{P}} \right\}$$

where $\vec{\mathcal{P}}$ is a complete set of runs of $\overrightarrow{M}$ on $\vec{R}$. By Property P1), we know that every run $\vec{P}$ of $\overrightarrow{M}$ on $\vec{R}$ such that $\vec{P}(\varepsilon) = \big(d, 0, \emptyset, \{r, \emptyset, \emptyset\}\big)$ mimics some run P of M on T. Let $\mathcal{P}$ be the set of all runs P obtained from some $\vec{P} \in \vec{\mathcal{P}}$. We show that $\mathcal{P}$ is a complete set of runs of M on T. Let P_2 be a generic run of M on T. By Property P2), there is a run P_2' of M on T such that $P_2' \preceq P_2$ and there is a run $\vec{P}_2$ of $\overrightarrow{M}$ on $\vec{R}$ that mimics P_2'. Now, since $\vec{\mathcal{P}}$ is a complete set of runs, there exists a run $\vec{P}_1 \in \vec{\mathcal{P}}$ such that $\vec{P}_1 \preceq \vec{P}_2$. Again, by Property P1), there exists a run P_1 of M on T which is mimicked by $\vec{P}_1$. Summing up, we have

i) $\vec{P}_1$ mimics P_1,
ii) $\vec{P}_2$ mimics P_2,
iii) $\vec{P}_1 \preceq \vec{P}_2$

By Property P3), this implies that $P_1 \preceq P_2'$ and hence, by transitivity (since $P_2' \preceq P_2$), we obtain $P_1 \preceq P_2$. This shows that $\mathcal{P}$ is a complete set of runs of M on T. We now write t' as $\big\{\big(q_h, \{X_{h,i}\}_{i\in I}, \emptyset\big) \,:\, h \in H\big\}$, where, for all $h \in H$ and for all $i \in I$,

- $q_h = (d_h, x_h, \mathcal{U}_h, \mathcal{V}_h)$, with $\mathcal{U}_h = \big\{U_{h,l}\big\}_{l\in L_h}$ and $\mathcal{V}_h = \big\{(r_{h,g}, V_{h,g}, W_{h,g})\big\}_{g\in G_h}$,
- $X_{h,i} = \big\{(d'_{h,i}, x'_{h,i}, \mathcal{U}'_{h,i,k}, \mathcal{V}'_{h,i}) \,:\, k \in K_{h,i}\big\}$, with $\mathcal{U}'_{h,i} = \big\{U'_{h,i,l}\big\}_{l\in L'_{h,i}}$ and $\mathcal{V}'_{h,i,k} = \big\{(r'_{h,i,k,g}, V'_{h,i,k,g}, W'_{h,i,k,g})\big\}_{g\in G'_{h,i,k}}$.

From the above arguments, we know an M-type t of T can be defined as the set of all triples of the form

$$\left(\begin{array}{c} r_{h,g_h}, \\ \left\{U'_{h,i,l} \,:\, i \in I,\ l \in L'_{h,i}\right\} \ \cup \\ \left\{\left(\bigcup_{k\in K_{h,i}} \bigcup_{g\in G''_k} V'_{h,i,k,g}\right) \,:\, i \in I,\ x'_{h,i} = 0,\ \forall\, k \in K_{h,i}.\ \emptyset \subsetneq G''_k \subseteq G'_{h,i,k}\right\}, \\ \left\{\left(d'_{h,i,k_{h,i}}, r'_{h,i,k_{h,i},g}, W'_{h,i,k_{h,i},g}\right) \,:\, i \in I,\ K_{h,i} = \{k_{h,i}\},\ x'_{h,i} = 1,\ g \in G_h\right\} \end{array} \right)$$

where h ranges over H and satisfies (i) $x_h = 0$, (ii) $L_h = \emptyset$, (iii) $G_h = \{g_h\}$ (namely, G_h is a set containing a single index g_h), (iv) $V_{h,g_h} = \emptyset$, and (v) $W_{h,g_h} = \emptyset$. Finally, note that such an M-type t can be effectively built from the $\overrightarrow{M}$-type t' of $\vec{R}$. The last part of the theorem, stating that $T \in \mathscr{L}(M)$ iff $\vec{R} \in \mathscr{L}(\overrightarrow{M})$ follows easily from the definition of acceptance condition of $\overrightarrow{M}$. □

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

Proof. Theorem 2 is an immediate consequence of Theorem 1, under the assumption that one can compute footprints of reducible trees. □

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 first give an intuitive account of the proof of the theorem, which is by induction on n (the rank of the considered tree). Let $\mathscr{F}$ be the set of transformations taken into account. As for the base case $n = 0$, one can easily show that the class of *regular* trees is closed under any transformation in $\mathscr{F}$. As for the inductive step, one fixes 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 over $t(T)$. Then, one has to show that there are a suitable tree automaton M' running on T, a rank $n-1$ tree $\vec{R}$ encoding a retraction R of T with respect to M', and a transformation $t' \in \mathscr{F}$ mapping $\vec{R}$ to a tree $t'(\vec{R})$ that encodes a retraction of $t(T)$ with respect to M. Finally, by exploiting the inductive hypothesis, it would turn out that $t'(\vec{R})$ is a rank $n-1$ tree and hence $t(T)$ is a rank n tree.

Hereafter, given a tree automaton $M = (S', A, C \cup \{\bot\}, \delta', \mathcal{I}, \mathcal{F})$ and an arbitrary set B of markers (this set will be used to label the edges of the outcoming retractions), we denote by M_B the B-augmented tree automaton obtained by extending M with the set $\mathcal{B} = \emptyset$, namely, the automaton $M_B = (S', A, B, C \cup \{\bot\}, \delta', \mathcal{I}, \mathcal{F}, \mathcal{B})$, where $\mathcal{B} = \emptyset$.

Before proving Theorem 3, we give some technical results. The following proposition relates the type of a B-augmented tree T and the types of some A-augmented trees $(F_c)_{c\in C}$ to the type of the B-augmented tree $T[\![F_c/c]\!]_{c\in C}$ (here we assume that $B \supseteq C$, in such a way that it is always guaranteed that second-order tree substitutions map B-augmented trees to B-augmented trees).

Proposition 7. *Let M be a tree automaton and let $\bar{t} = (t_c)_{c\in C}$ a tuple of M_A-types (recall that M_A is the A-augmented tree automaton obtained from M). One can compute a tree automaton $M^{\bar{t}}$ and, for any given set B of markers and for any given $M^{\bar{t}}_B$-type t, an M_B-type t' such that, for every B-augmented tree T and for every tuple of replacing trees $\bar{F} = (F_c)_{c\in C}$, if t is an $M^{\bar{t}}_B$-type of T and if $\bar{t}$ is an M_A-type of $\bar{F}$, then t' is an M_B-type of $T[\![F_c]\!]_{c\in C}$.*

In order to prove Proposition 7, we shall distinguish between two kinds of second-order tree substitutions, namely, *erasing* and *non-erasing* ones. Let assume that $A = \{a_1, ..., a_k\}$. A second-order tree substitution $T[\![F_c/c]\!]_{c\in C}$ is said to be erasing if there is $c \in C$ such that F_c is either the empty tree or a tree consisting of a single vertex marked with an element $a_i \in A$. In such a case, c (resp., F_c) is said to be an *erased color* (resp., an *erasing tree*) of the substitution. Given an erasing second-order tree substitution $T[\![F_c/c]\!]_{c\in C}$, we then denote by C^- the set of the erased colors of C and by C^+ the set of all other colors. Moreover, if $F_c = c\langle a_1, ..., a_k\rangle$ holds for every $c \in C^+$ (namely, if the substitution *preserves* non-erased colors), then we say that the substitution is *shrinking*. It is easy to see that for every (possibly erasing) second-order tree substitution $T[\![F_c/c]\!]_{c\in C}$, we have

$$T[\![F_c/c]\!]_{c\in C} = T[\![F_c/c]\!]_{c\in C^-}[\![F_c/c]\!]_{c\in C^+}$$

where the first substitution in the right-hand side of the equation is shrinking and the second one is non-erasing. Such a property allows us to prove the above proposition separately for *shrinking* substitutions (Lemma 8) and for *non-erasing* substitutions (Lemma 9).

Let us consider first the case of a shrinking substitution $T[\![F_c/c]\!]_{c\in C}$. Clearly, for every $c \in C$, one of the following conditions holds:

1. $F_c = c\langle a_1, ..., a_k\rangle$,
2. $F_c = a_j$ for some $1 \le j \le k$,
3. $F_c = \emptyset$.

In the first case ($F_c = c\langle a_1, ..., a_k\rangle$), every M_A-type of F_c is a set consisting of triples of the form $\big(s_h, \emptyset, \{(a_j, q_{h,j}, Y_{h,j})\}_{1\le j\le k}\big)$, where h ranges over a suitable finite set of indices. In the second case ($F_c = a_j$), every M_A-type of F_c is a set consisting of triples of the form $\big(s_h, \emptyset, \{(a_j, q_h, Y_h)\}\big)$, where h ranges again over a suitable finite set of indices. In the third case ($F_c = \emptyset$), every M_A-type of F_c is a set consisting of triples of the form $\big(s_h, \{X_{h,i}\}_{i\in I}, \emptyset\big)$, where h ranges over a suitable finite set of indices. This means that, for every automaton M and for every pair of shrinking substitutions $T[\![F_c/c]\!]_{c\in C}$ and $T'[\![F'_c/c]\!]_{c\in C}$, if $\bar{F} = (F_c)_{c\in C}$ and $\bar{F}'(F'_c)_{c\in C}$ have a common M_A-type, then $\bar{F} = \bar{F}'$ holds and the two substitutions are in fact the same. Therefore, in the case of a shrinking substitution, Proposition 7 can be reformulated as follows.

Lemma 8. *Let M be a tree automaton and let $\bar{F} = (F_c)_{c\in C}$ be a tuple of replacing trees of a shrinking substitution. One can compute a tree automaton $M^{\bar{F}}$ and, for any given set B of markers and for any given $M_B^{\bar{F}}$-type t, an M_B-type t' such that for every B-augmented tree T, if t is an $M_B^{\bar{F}}$-type of T, then t' is an M_B-type of $T[\![F_c]\!]_{c\in C}$.*

Proof. Let $M = (S, A, C \cup \{\bot\}, \delta, \mathcal{I}, \mathcal{F})$. We have to define an automaton $M^{\bar{F}}$, running on a B-augmented tree T, that mimics the transitions of M on $T' = T[\![F_c]\!]_{c\in C}$. For the sake of simplicity, we assume that both T and T' are non-empty full B-augmented trees (if this is not the case, simply replace T by $T_\bot$ and T' by $T'_\bot$). The states of $M^{\bar{F}}$ are quadruples of the form (c, x, s, V), where the first component $c \in C \cup \{\bot\}$ is used to store the last (non-erased) symbol read by the automaton M, the second component $x \in \{0, 1\}$ is used to disable the computation of $M^{\bar{B}}$ on an erased subtree of T, the third component $s \in S$ is used to store the current state of M during an enabled computation, and the fourth component $V \subseteq S$ is used to store the set of states along the access path of the current position. We thus define the automaton $M^{\bar{F}} = (Q, A, C\cup\{\bot\}, \delta', \mathcal{I}', \mathcal{F}')$ as follows (the acceptance condition is not relevant here):

- $Q = (C \cup \{\bot\}) \times \{0, 1\} \times S \times \mathscr{P}(S)$,
- for every tuple of states $q = (c, x, s, V)$, $q_{a_1} = (c_{a_1}, x_{a_1}, s_{a_1}, V_{a_1})$, ..., $q_{a_k} = (c_{a_k}, x_{a_k}, s_{a_k}, V_{a_k})$ in Q, we have $\big(q, c', (q_a)_{a\in A}\big) \in \delta'$ iff the following conditions hold
  - i) if $x = 0$, and $c' \in C$ with $F_{c'} = c'\langle a_1, ..., a_k\rangle$, then $c_{a_1} = ... = c_{a_k} = c'$, $x_{a_1} = ... = x_{a_k} = 0$, $\big(s, c', (s_a)_{a\in A}\big) \in \delta$, and $V_a = V \cup \{s_a\}$ for all $a \in A$,
  - ii) if $x = 0$ and $c' = \bot$, then $c_{a_1} = ... = c_{a_k} = \bot$, $x_{a_1} = ... = x_{a_k} = 0$, $\big(s, \bot, (s_a)_{a\in A}\big) \in \delta$, and $V_a = V \cup \{s_a\}$ for all $a \in A$,
  - iii) if $x = 0$ and $c' \in C$ with $F_{c'} = a_j$ for some $1 \le j \le k$, then $c_{a_1} = ... = c_{a_k} = c$, $x_{a_j} = 0$, $x_{a_{j'}} = 1$ for all $j' \neq j$, $s_{a_1} = ... = s_{a_k} = s$, and $V_a = V$ for all $a \in A$,
  - iv) if $x = 0$ and $c' \in C$ with $F_{c'} = \emptyset$, then $c_{a_1} = ... = c_{a_k} = c$, $x_{a_1} = ... = x_{a_k} = 1$, $s_{a_1} = ... = s_{a_k} = s$, and $V_a = V$ for all $a \in A$,
  - v) if $x = 1$, then $c_{a_1} = ... = c_{a_k} = c$, $x_{a_1} = ... = x_{a_k} = 1$, $s_{a_1} = ... = s_{a_k} = s$. $V_a = V$ for all $a \in A$.

Given the above definition, one can compute an M_B-type t' of T' from a given $M_B^{\bar{F}}$-type t of T. Precisely, we write t as $\big\{\big(q_h, \{X_{h,i}\}_{i\in I}, \{(b_j, q_{h,j}, Y_{h,j})\}_{j\in J}\big) \,:\, h \in H\big\}$, where, for all $h \in H$, for all $i \in I$, and for all $j \in J$,

- $q_h = (c_h, x_h, s_h, V_h)$,
- $X_{h,i} = \big\{q_{h,i,l} \,:\, l \in L_{h,i}\big\}$, with $q_{h,i,l} = (c'_{h,i,l}, x'_{h,i}, s'_{h,i,l}, V'_{h,i})$ for all $l \in L_{h,i}$,
- $q_{h,j} = (d'_{h,j}, y'_{h,j}, r'_{h,j}, W'_{h,j})$,
- $Y_{h,j} = \{q_{h,j,g}\}_{g\in G_{h,j}}$.

Then, it is routine to see that the M-type t' of T' can be defined as the set of all triples of the form

$$\left(\begin{array}{c} s_h, \\ \left\{\left\{s'_{h,i,l}\right\}_{l\in L_{h,i}} \,:\, i \in I,\ x'_{h,i} = 0\right\}, \\ \left\{\left(d'_{h,j}, r'_{h,j}, W'_{h,j}\right) \,:\, j \in J\right\} \ \cup \\ \left\{\left(c'_{h,i,l_{h,i}}, s'_{h,i,l_{h,i}}, V'_{h,i}\right) \,:\, i \in I, x'_{h,i} = 1, L_{h,i} = \{l_{h,i}\}\right\} \end{array}\right)$$

where h ranges over H and satisfies (i) $c_h = \bot$ and (ii) $x_h = 0$. □

Now, we consider the case of non-erasing second-order tree substitutions.

Lemma 9. *Let M be a tree automaton and let $\bar{t} = (t_c)_{c\in C}$ be a tuple of M_A-types. One can compute a tree automaton $M^{\bar{t}}$ and, for any given set B of markers and for any given $M_B^{\bar{t}}$-type t, an M_B-type t' such that, for every B-augmented tree T and for every tuple of replacing trees $\bar{F} = (F_c)_{c\in C}$ of a non-erasing substitution, if t is an $M_B^{\bar{t}}$-type of T and $\bar{t}$ is an M_A-type of $\bar{F}$, then t' is an M_B-type of $T[\![F_c]\!]_{c\in C}$.*

Proof. Let T be a B-augmented tree and T' be the B-augmented tree obtained from a non-erasing substitution $T[\![F_c/c]\!]_{c\in C}$. Without loss of generality, we assume that T, F_c for all $c \in C$, and T' are non-empty full trees. We define the *induced factorization* Π of T' according to second-order tree substitution $T[\![F_c/c]\!]_{c\in C}$. For each vertex v in T, we recursively define a set D_v of vertices of T' that correspond to v:

- given $v = \varepsilon$, we set $D_v = \{\varepsilon\}$,

- given $v \in \mathcal{D}om(T)$ and $D_v \subseteq \mathcal{D}om(T')$, for every a-successor v' of v in T', we set $D_{v'} = D_v \cdot \{w \in \mathcal{F}r(F_{T(v)}) : F_{T(v)}(w) = a\}$.

The factorization Π of T' is then defined as follows:

- $\mathcal{D}om(\Pi) = \bigcup_{v \in \mathcal{D}om(T)} D_v$,
- (u, u') is an a-labeled edge in Π iff there exist $v, v' \in \mathcal{D}om(T)$ such that $u \in D_v$, $u' \in D_{v'}$, and (v, v') is an a-labeled edge in T.

Moreover, if u is a vertex of Π, then we denote by $\widetilde{u}$ the (unique) vertex of T such that $u \in D_{\boldsymbol{\epsilon}}$. From the above definitions it is clear that for every vertex u of Π, the marked factor of T' rooted at u with respect to Π is exactly the replacing tree $F_{T(\boldsymbol{\epsilon})}$.

Now the proof of the lemma is rather trivial. Given a tree automaton M and a tuple $\bar{t} = (t_c)_{c \in C}$ of M_A-types, we define the tree automaton $M^{\bar{t}}$ as follows. First, we denote by $\overrightarrow{M}$ the retraction automaton given in Definition 5, where M is viewed as an A-augmented tree automaton. We let the states of $M^{\bar{t}}$ be exactly those of $\overrightarrow{M}$, we let the input alphabet of $M^{\bar{t}}$ be the set of colors, and we let $\big(q, c, (q_a)_{a \in A}\big)$ be a transition of $M^{\bar{t}}$ iff $\big(q, t_c, (q_a)_{a \in A}\big)$ is a transition of $\overrightarrow{M}$. Now, given the B-augmented tree T and the tuple $(F_c)_{c \in C}$ of (non-erasing) replacing trees, we let Π be the factorization of $T' = T[\![F_c]\!]_{c \in C}$ induced by the substitution $T[\![F_c]\!]_{c \in C}$ and we let R be the (A-labeled) retraction of T', with respect to Π and M, such that $R(u) = t_{T(\boldsymbol{\epsilon})}$ for every vertex u in Π. It is clear that the encoding $\overrightarrow{R}$ of the retraction R can be viewed as the memoryless recoloring of T according to the function Ω that maps a color c to the corresponding M_A-type t_c. Therefore, $\overrightarrow{P}$ is a run of $\overrightarrow{M}$ on $\overrightarrow{R}$ iff $\overrightarrow{P}$ is a run of $M^{\bar{t}}$ on T. This implies that every $\overrightarrow{M}_B$-type of $\overrightarrow{R}$ can be viewed as an $M_B^{\bar{t}}$-type of T, and vice versa. Finally, given an $\overrightarrow{M}_B$-type of $\overrightarrow{R}$ (or, equivalently, an $M_B^{\bar{t}}$-type of T), it is easy to compute an M_B-type of T' (simply follow the proof of Theorem 1) and this concludes the proof. □

Now, we focus on finite-state recolorings. We formally define Mealy tree automata with bounded lookahead (these devices compute finite-state recolorings with bounded lookahead) and tree transducers with bounded lookahead. Hereafter, given a finite prefix-closed subset D of A^*, we denote by $D{-}\mathtt{Trees}$ the set of all trees whose domain is included in D. Moreover, given a natural number d, we denote by $A^{\leq d}$ the set $\bigcup_{0 \leq i \leq d} A^i$ (note that $A^{\leq d}$ is a finite prefix-closed subset of A^*).

Definition 8. A *Mealy tree automaton with bounded lookahead* is a tuple $N = (S, A, C, D, \delta, \Omega, s_0)$, where:

- S is a finite set of states,
- A is a finite set of labels,
- C is a finite set of colors,
- $D = A^{\leq d}$, for some $d \in \mathbb{N}$,
- $\delta : S \times D{-}\mathtt{Trees} \times A \rightarrow S$ is a transition function,
- $\Omega : S \times D{-}\mathtt{Trees} \rightarrow C$ is a recoloring function,
- s_0 is an initial state.

The *run* of N on a tree T is the (unique) S-colored tree P such that (i) $\mathcal{D}om(P) = \mathcal{D}om(T)$, (ii) $P(\varepsilon) = s_0$, and (iii), for every vertex v in P and for every a-successor v_a of v, we have $P(v_a) = \delta\big(P(v), T^{\downarrow v}|_D, a\big)$, where $T^{\downarrow v}$ denotes the subtree of T rooted at v and $T^{\downarrow v}|_D$ denotes the tree obtained from $T^{\downarrow v}$ by restricting its domain to the set D (here we assume that the domain of $T^{\downarrow v}$ is a prefix-closed subset of A^*). The *output* of N on T is the tree T' such that $\mathcal{D}om(T') = \mathcal{D}om(T)$ and, for every vertex v in T, $T'(v) = \Omega\big(P(v), T^{\downarrow v}|_D\big)$. In such a case, we say that T' is a *finite-state recoloring with bounded lookahead* of T via N.

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 (called replacing tree), which may depend on the color of the vertex and on the current state (when the transducer is equipped with a lookahead facility, the replacing tree depends also on the form of the subtree rooted at the current vertex). At each computation step, different states can be spread among different (copies of the) successors of the current vertex. In other words, this means that, unlike second-order tree substitutions, a tree transducer may exhibit different behaviors on different copies of the same subtree. In order to distinguish between such different behaviors, we mark some distinguished leaves in the replacing trees with symbols belonging to a finite set B. Then, we identify the subtree to be attached at each marked leaf by using a placeholder function $\lambda : B \rightarrow A$, which maps a B-colored leaf to the corresponding placeholder for the substitution. Moreover, like finite-state recolorings, tree transducers can be enriched with the facility of bounded/rational lookahead. Below, we give a formal definition of tree transducer with bounded lookahead (it is worth mentioning that different but equivalent definitions can be found in the literature). As usual, we assume that edges are labeled over a finite ordered set $A = \{a_1, ..., a_k\}$ and vertices are colored over a finite ordered set $C = \{c_1, ..., c_m\}$. We further introduce the set $B = \{b_1, ..., b_n\}$ of markers and we denote by $\mathtt{RegTrees}_B$ the set of all regular $B \cup C$-augmented trees.

Definition 9. A *tree transducer with bounded lookahead* is a tuple $N = (S, A, B, C, D, \lambda, \delta, \Omega, s_0)$, where

- S is a finite set of states,
- A is a finite set of edge labels,
- B is a finite set of markers, disjoint from C,
- C is a finite set of vertex colors,
- $D = A^{\leq d}$, for some $d \in \mathbb{N}$,
- $\lambda : B \rightarrow A$ is the placeholder function,
- δ is a transition function from $S \times D{-}\mathtt{Trees} \times B$ to S,
- Ω is the replacement function from $S \times D{-}\mathtt{Trees}$ to $\mathtt{RegTrees}_B$,
- s_0 is an initial state.

In order to define the output of the tree transducer N on a (possibly infinite) tree T, we shall use a limit-passing argument. For every $i \in \mathbb{N}$, we let N^i be the function that maps a state s and a (possibly infinite) tree T to a tree T' and that satisfies the following conditions:

- if T is the empty tree, then T' is the empty tree as well,
- if $i = 0$, then $T' = T$,
- if $i > 0$, then $N^i(s,T) = F\big[T^{(b)}/b\big]_{b\in B}$, where $F = \Omega\big(s, T(\varepsilon)\big)$ and $T^{(b)} = N^{i-1}\big(s_b, T^{\downarrow\lambda(b)}\big)$, with $s_b = \delta\big(s, T(\varepsilon), b\big)$.

We now consider the ω-complete partial order $\sqsubseteq$ over (possibly infinite) trees, defined by letting $T_1 \sqsubseteq T_2$ iff T_1 can be obtained by pruning some subtrees of T_2. It is easy to verify, by exploiting induction on i, that, given any tree T, the sequence $N^0(s_0,T)$, $N^1(s_0,T)$, $N^2(s_0,T)$, ... is increasing with respect to $\sqsubseteq$ (to see this, simply prune the subtrees issued from the current positions of the transducer). This allows us to define the 'limit' $N^\omega(s_0,T)$ as its least upper bound of the above sequence with respect to $\sqsubseteq$. The resulting tree is said to be the *output* of N on T.

In order to prove the closure of the class of rank n trees under finite-state recolorings with bounded lookahead, we need to relate the types of the factors of a tree T to the types of the factors of the tree T' resulting from T via a finite-state recoloring with bounded lookahead. Since the application of a finite-state recoloring with bounded lookahead requires the inspection of some descendants of the vertices in a tree, we cannot consider the types of the factors of the tree T independently from the contexts where they are embedded. As a matter of fact, consider a Mealy tree automaton with bounded lookahead N and a marked factor F of an input tree T with respect to a certain factorization Π, defined over an arbitrary set of markers B. By definition, F is a non-empty full B-augmented tree, whose root corresponds to a certain vertex u of Π and whose leaves w correspond to the successors uw of u in Π. Now note that the recoloring of the vertices of F via N may depend on the form of the subtrees of T rooted at the vertices uw, with w ranging over $\mathcal{F}r(F)$. In fact, since such a recoloring depends only on *bounded* portions of the above mentioned subtrees, the result of it can be guessed by a suitable tree automaton that runs on the marked factor F.

We now introduce some notation. Let F be a non-empty full B-augmented tree. For any given Mealy tree automaton with bounded lookahead $N = (S, A, C, D, \delta, \Omega, s_0)$ and for any given state $\widetilde{s}_0 \in S$, we define $N_{\mathfrak{s}_0}$ as the Mealy tree automaton obtained from N by replacing its initial state s_0 with $\widetilde{s}_0$. Then, a tuple of replacing trees $(H_b)_{b\in B}$, we denote by $N_{\mathfrak{s}_0}\big(F[\![H_b/b]\!]_{b\in B}\big)$ the output of $N_{\mathfrak{s}_0}$ running on the input tree $F[\![H_b/b]\!]_{b\in B}$. Finally, we define $N_{\mathfrak{s}_0}\big(F, (H_b)_{b\in B}\big)$ as the $B \times S$-augmented tree F' such that

- $\mathcal{D}om(F') = \mathcal{D}om(F)$,
- for every C-colored vertex v in F, $F'(v) = N_{\mathfrak{s}_0}\big(F[\![H_b/b]\!]_{b\in B}\big)(v)$,
- for every B-colored leaf v of F, $F'(v) = \big(F(v), P(v)\big)$, where P is the run of $N_{\mathfrak{s}_0}$ on $F[\![H_b/b]\!]_{b\in B}$.

Intuitively, the tree $N_{\mathfrak{s}_0}\big(F, (H_b)_{b\in B}\big)$ represents the recoloring via N of a marked factor F of T (with respect to a certain factorization Π) rooted at a vertex u, under the hypothesis that the state of N at u is $\widetilde{s}_0$ and that, for every b-successor u' of u in Π, H_b is the lookahead tree $T^{\downarrow u'}|_D$.

The following proposition relates the $M_{B'}$-type of the tree $N_{\mathfrak{s}_0}\big(F, (H_b)_{b\in B}\big)$, where $B' = B \times S$, to the M_B^N-type of F, where M^N is a suitable tree automaton (computable from N and M) and M_B^N is the extension of M^N with the set B of markers. In the following construction, one should keep in mind that the automaton M^N does not depend on the initial state of N, nor on the set B of markers.

Proposition 10. *Let $N = (S, A, C, D, \delta, \Omega, s_0)$ be a Mealy tree automaton with bounded lookahead and let $M = (S', A, C \cup \{\bot\}, \delta', \mathcal{I}, \mathcal{F})$ be any tree automaton. One can compute a tree automaton M^N (which does not depend on the initial state of N) and, for any given set B of markers and for any given M_B^N-type $\mathfrak{t}$, one can compute a non-empty B-augmented tree H such that, for every non-empty full B-augmented tree F, if $\mathfrak{t}$ is an M_B^N-type of F, then $H = F|_D$.*

Furthermore, from the above M_B^N-type t, from any given tuple $(H_b)_{b\in B}$ of trees with domain included in D, and from any given state $\widetilde{s}_0 \in S$, one can compute an $M_{B'}$-type t', with $B' = B \times S$, such that, for every non-empty full B-augmented tree F having t as M_B^N-type, t' is an $M_{B'}$-type of $N_{\mathfrak{s}_0}\big(F,(H_b)_{b\in B}\big)$.

Proof. We first give an intuitive idea of how the automaton M^N works. The states of M^N are quintuples of the form $z = (s, s', v, G, D')$, where $s \in S$ is the current state of N, $s' \in S'$ is the current state of M, $v \in D \cup \{\infty\}$ identifies the current position (up to a certain depth) in the input factor F, $G \in D{-}\mathtt{Trees}$ is the guessed portion of the subtree of $F[\![H_b/b]\!]_{b\in B}$ rooted at the current position of the automaton (the domain of G is always included in D), and D' is a prefix-closed subset of the domain of G defining the portion of G that is going to be checked. The transitions of M^N on F mimic both the transitions of N on $F[\![H_b/b]\!]_{b\in B}$ (note that these are deterministic, but depend on the guessed tree G) and the transitions of M on $N\big(F,(H_b)_{b\in B}\big)$. Formally, M^N is defined as follows (the acceptance condition of M^N is not relevant here):

- the states of M^N are quintuples from the set $Q = S \times S' \times (D \cup \{\infty\}) \times D{-}\mathtt{Trees} \times \mathscr{P}(D)$,
- for every tuple of states $q, (q_a)_{a\in A}$ of M^N and for every color c, $\big(q, c, (q_a)_{a\in A}\big)$ is a transition of M^N iff (i) q is of the form (s, s', v, G, D'), with $G(\varepsilon) = c$, and (ii) there are some states $(s'_a)_{a\in A}$ of M, some trees $(G_a)_{a\in A}$, and some sets $(D'_a)_{a\in A}$ such that, for all $a \in A$, $q_a = (\delta(s, G, a), s'_a, v_a, G_a, D'_a)$, with v_a being either $v \cdot a$ or $v_a = \infty$, depending on whether $|v \cdot a| \in D$ or not, D'_a being the set $\{v \in A^* : av \in D'\}$, and G_a satisfying $\mathcal{D}om(G_a) = \{vw \in D : av \in \mathcal{D}om(G), |w| \leq 1\}$ and $G_a(v) = G(av)$ for all $av \in \mathcal{D}om(G)$.

Now, let B be any arbitrary set of markers, let F be a non-empty full B-augmented tree, and let P be a run of M_B^N on F (recall that M_B^N is the B-augmented tree automaton obtained from M^N). If for every leaf v of P, we have $\mathcal{P}rj_5(P(v)) = \emptyset$, then one can easily prove that the two trees F and $\mathcal{P}rj_4(P(\varepsilon))$ coincide on the set of vertices $D' = \mathcal{P}rj_5(P(\varepsilon))$. This means that, by looking at the feature $[F, P]$, one can check whether a suitable non-empty portion of the tree F was correctly guessed at the beginning of the computation of M_B^N. More formally, there is a computable function f that maps any feature of the form $[F, P]$ either to the empty tree (this happens when there is a leaf v of P such that $\mathcal{P}rj_5(P(v)) \neq \emptyset$) or to the tree $F|_{D'}$, where $D' = \mathcal{P}rj_5(P(\varepsilon))$. Therefore, given an M_B^N-type t of F, we can evaluate f on the features contained in t and extract the *largest* portion of F that was correctly guessed; we denote such a tree by G. It is easy to see that $\mathcal{D}om(G) = D \cap (\mathcal{D}om(F) \setminus \mathcal{F}r(F))$. Thus, in order to obtain the tree H satisfying $H = F|_D$, we simply have to complete the tree G with the markings at the leaves of F. Again, this can be done by inspecting the M_B^N-type t. More precisely, if we choose in t any feature $\big(q, \{X_i\}_{i\in I}, \{(b_j, q_j, Y_j)\}_{j\in J},\big)$ such that $\mathcal{P}rj_3(q) = \varepsilon$, then we can define the tree H as follows:

- $\mathcal{D}om(H) = \mathcal{D}om(G) \cup \big(\{\mathcal{P}rj_3(q_j)\}_{j\in J} \setminus \{\infty\}\big)$ $(= \mathcal{D}om(G) \cup (\mathcal{F}r(F) \cap D) = \mathcal{D}om(F) \cap D)$,
- for every $v \in \mathcal{D}om(G)$, $H(v) = G(v)$,
- for every $v \in \mathcal{D}om(H) \setminus \mathcal{D}om(G)$, there is a unique $j \in J$ such that $\mathcal{P}rj_3(r_i) = v$, and thus we set $H(v) = b_j$.

It is easy to verify that $H = F|_D$.

As for the second claim, let $(H_b)_{b\in B}$ be a tuple of trees whose domains are included in D and let $\widetilde{s}_0$ be a state of N. Given a run P of M_B^N on F, one can prove that, if the following two conditions hold

C1) $\mathcal{P}rj_1(P(\varepsilon)) = \widetilde{s}_0$,

C2) for every b-colored leaf v of F, $\mathcal{P}rj_4(P(v))$ coincides with H_b,

then the lookahead trees appearing in $\mathcal{P}rj_4(P)$ were correctly guessed, namely, for every vertex v of F, $\mathcal{P}rj_4(P(v)) = \big(F[\![H_b]\!]_{b\in B}\big)^{\downarrow v}|_D)$. Moreover, in such a case, we know that

P1) $\mathcal{P}rj_1(P)$ is the restriction to $\mathcal{D}om(F)$ of the run of $N_{\mathfrak{s}_0}$ on $F[\![H_b/b]\!]_{b\in B}$,

P2) $\mathcal{P}rj_2(P)$ is a run of M on $N_{\mathfrak{s}_0}\big(F,(H_b)_{b\in B}\big)$.

Conversely, every run of M on $N_{\mathfrak{s}_0}\big(F,(H_b)_{b\in B}\big)$ can be obtained as the projection into the second component of a suitable run of M_B^N on F that satisfies Conditions C1) and C2). Finally, if the M_B^N-type t of F is a set of the form

$$\left\{ \begin{pmatrix} q_h, \\ \{X_{h,i} : i \in I\}, \\ \{(b_j, r_{h,j}, Y_{h,j}) : j \in J\} \end{pmatrix} : h \in H \right\}$$

then we can compute an $M_{B'}$-type of $N_{\mathfrak{e}_0}\big(F,(H_b)_{b\in B}\big)$, where $B'=B\times S$, as

$$\left\{\begin{pmatrix}\mathcal{P}rj_2(q_h),\\ \big\{\mathcal{P}rj_2(X_{h,i})\ :\ i\in I\big\},\\ \big\{\big((b_j,\mathcal{P}rj_2(r_{h,j})),\mathcal{P}rj_2(r_{h,j}),\mathcal{P}rj_2(Y_{h,j})\big)\ :\ j\in J\big\}\end{pmatrix}\ :\ h\in H'\right\}$$

where H' is the set of all indices $h\in H$ such that $\mathcal{P}rj_1(q_h)=\widetilde{s}_0$ and, for every $j\in J$, $\mathcal{P}rj_4(r_{h,j})=H_{b_j}$. □

Now, we can prove Theorem 3.

Proof. Assume that the class of rank 0 (i.e., regular) trees is closed under regular tree morphisms and finite-state recolorings with bounded lookahead (this is easy to prove). It suffices to prove analogous closure properties for rank n trees, where $n>0$, under the assumption that analogous properties hold for rank $n-1$ trees.

As for the closure of rank n trees under regular tree morphisms, let σ be the regular tree morphism specified by a tuple $\bar{F}=(F_c)_{c\in C}$ of regular replacing trees, let M be a tree automaton, let T be a rank n tree with footprint ξ, and let $T'=\sigma(T)$. We have to show that T' is a rank n tree and we have to build a footprint ξ' of T' on the grounds of σ and ξ. We shall exploit the inductive hypothesis that rank $n-1$ trees are closed under regular tree morphisms. We denote by $\bar{t}=(t_c)_{c\in C}$ the basic M_A-type of $\bar{F}$ (recall that M_A is the A-augmented tree automaton obtained from M). We define M^σ as the (computable) tere automaton $M^{\bar{t}}$ specified by Proposition 7. Given an arbitrary finite set B, we identify $2+|B|$ distinct classes of B-augmented trees:

1. the class of trees whose image under σ is the empty tree,
2. the class trees whose image under σ is the singleton tree b, for each choice of b in B,
3. the class of trees whose image under σ is a non-empty non-singleton tree.

Without loss of generality, we can assume that the tree automaton M^σ is able to distinguish between the above categories of trees (is this is not the case, simply refine the state space of M^σ). More formally, we assume that, if F and F' are two B-augmented trees having a common M^σ_B-type t, then either $F=F'$ holds (this accounts for case 1. and case 2. above) or both F and F' are non-empty non-singleton trees (case 3. above). Now, recall that, given an M^σ_B-type t of a B-augmented tree F, one can compute an M_B-type t' of the tree $\sigma(F)$ (this follows from Proposition 7 and from the definition of M^σ). This is to say that there exist two computable functions f and g such that, whenever t is an M^σ_B-type of a B-augmented tree F, then

i) $f(t)$ is an M_B-type of $\sigma(F)$,
ii) $g(t)=0$ iff $\sigma(F)=\emptyset$,
iii) $g(t)=b$ iff $\sigma(F)=b$,
iv) $g(t)=1$ iff $\sigma(F)$ is a non-empty non-singleton tree.

We now fix $B=\mathcal{P}rj_1(\xi(M^\sigma))$ and $\overrightarrow{\xi}=\mathcal{P}rj_2(\xi(M^\sigma))$. By definition of rank n tree, we know that there exist a factorization Π of T, with respect to the set B, and a rank $n-1$ tree $\overrightarrow{R}$ with footprint $\overrightarrow{\xi}$ which encodes a retraction of T with respect to Π and M^σ. We have to define a suitable factorization Π' of T' in such a way that each marked factor of T' can be viewed as the image of a corresponding marked factor of T under the regular tree morphism σ. The factorization Π' is a straightforward generalization of the notion of *induced* factorization. Given a vertex v in T, we define the set D_v consisting of all vertices of T' that correspond to v under σ:

- given $v=\varepsilon$, we set $D_v=\{\varepsilon\}$,
- given $v\in\mathcal{D}om(T)$ and the set D_v, we define, for every a-successor v_a of v, $D_{v_a}=D_v\cdot\{w\ :\ w\in\mathcal{F}r(F_{T(v)}),F_{T(v)}(w)=a\}$.

We thus define the factorization Π' of T', with respect to B, as follows:

- $\mathcal{D}om(\Pi')=\bigcup_{v\in\mathcal{D}om(\Pi)}D_v$,
- for every $u,u'\in\mathcal{D}om(\Pi')$, (u,u') is a b-labeled edge in Π' iff u' is a descendant of u in T' and there is a b-labeled edge (v,v') in Π such that $u\in D_v$ and $u'\in D_{v'}$.

Note that, for every vertex u in Π', there is a vertex v in Π such that $u\in D_v$. Moreover, for every pair of vertices v,v' in Π, either $D_v\cap D_{v'}=\emptyset$ or $D_v=D_{v'}$ (in the latter case v is an ancestor or a descendant of v' and at least one of $F_{T(v)}$ and $F_{T(v')}$ is a singleton tree). Thus, for every vertex u of Π', we can define a corresponding vertex $\widetilde{u}$ in Π as the *least* vertex (according to the ordering given by the successor relation of Π) among all vertices v of Π such that $u\in D_v$ (note that all such vertices lie on the same path). We let the reader check that for every vertex u of Π', the marked factor F'_u of T' rooted at u with respect to Π' is isomorphic to the tree $\sigma(T_{\mathfrak{e}})$, where $T_{\mathfrak{e}}$ is the marked factor of T rooted at $\widetilde{u}$ with respect to Π. Now, let assume $B=\{b_1,...,b_n\}$. For any given M^σ_B-type t, we define the B-augmented tree F'_t as follows:

- if $g(t) = 0$, then F'_t is the empty tree,
- if $g(t) = b_j$, for some $b_j \in B$, then F'_t is the singleton tree b_j,
- if $g(t) = 1$, then F'_t is the tree $t'\langle b_1, ..., b_n\rangle$, where $t' = f(t)$.

Clearly, the tree F'_t can be effectively built on the grounds of the M^σ_B-type t. We then denote by σ' the regular tree morphism that replaces each M^σ_B-type t with the B-augmented tree F'_t and we let $\overrightarrow{R}' = \sigma'(\overrightarrow{R})$. It is easy to see that $\overrightarrow{R}'$ encodes a retraction of T' with respect to Π and M. Since, by inductive hypothesis, rank $n-1$ trees are closed under regular tree morphisms, one can compute a footprint $\overrightarrow{\xi}'$ of $\overrightarrow{R}'$ on the grounds of the footprint $\overrightarrow{\xi}$ of $\overrightarrow{R}$. Hence, the function ξ' that maps a tree automaton M to the pair $(B, \overrightarrow{\xi}')$ is a footprint of T' and this concludes the first part of the proof.

As for the closure under finite-state recolorings with bounded lookahead, let $N = (S, A, C, D, \delta, \Omega, s_0)$ be a Mealy tree automaton with bounded lookahead, where $D = A^{\leq d}$. Further let T be a rank n tree with footprint ξ, P be the run of N on T, and T' be the output of N on T. We denote by M^N the tree automaton specified in Proposition 10 and we let $B = \mathcal{P}rj_1(\xi(M^N))$ and $\overrightarrow{\xi} = \mathcal{P}rj_2(\xi(M^N))$. By definition of rank n tree, we know that there exists a factorization Π of T with respect to B and a rank $n-1$ tree $\overrightarrow{R}$ with footprint $\overrightarrow{\xi}$ encoding a retraction of T with respect to Π and M^N. We have to build a factorization Π' and the encoding $\overrightarrow{R}'$ of a retraction of T' on the grounds of Π and $\overrightarrow{R}$. To do that, we first build the lookahead trees $T^{\downarrow u}|_D$, for every vertex u in Π. For the sake of brevity, for every $u \in \mathcal{D}om(\Pi)$, we denote by $\overrightarrow{u}$ the (unique) vertex of $\overrightarrow{R}$ that corresponds to u (under the bisimulation relation between $\overrightarrow{R}$ and the retraction of T), by F_u the marked factor of T rooted at u with respect to Π, and by $t_{\overrightarrow{u}}$ the M^N_B-type $\overrightarrow{R}(\overrightarrow{u})$ of the marked factor F_u. From Proposition 10, we know that there is a computable function f such that, for every $u \in \mathcal{D}om(\Pi)$, $f(t_{\overrightarrow{u}})$ coincides with the marked factor F_u restricted to the domain D. Since every leaf of F_u corresponds to a successor u' of u in Π, $f(t_{\overrightarrow{u}})$ is a *non-empty* tree and thus one can build arbitrary large portions of the subtree $T^{\downarrow u}$ by evaluating f on the colors (i.e., the M^N_B-types) of the successors of $\overrightarrow{u}$ in $\overrightarrow{R}$. Precisely, there is a computable function g such that, for every $u \in \mathcal{D}om(\Pi)$,

$$g\big(\overrightarrow{R}^{\downarrow\overrightarrow{u}}|_E\big) = T^{\downarrow u}|_D,$$

where $E = B^{\leq d}$. On the grounds of such function g, we can define another (computable) function h that maps any lookahead subtree $\overrightarrow{R}^{\downarrow\overrightarrow{u}}|_{E'}$, where $E' = B^{\leq d+1}$, to the tuple $(H_b)_{b\in B}$, where $H_b = T^{\downarrow u_b}|_D$, with u_b being any b-successor of u in Π (note that, by previous arguments, H_b does not depend on the choice of the successor).

We now provide a factorization Π' and the encoding $\overrightarrow{R}'$ of a retraction of T'. Let $B' = B \times S$ and let Π' be the factorization of T' with respect to B' defined as follows:

- $\mathcal{D}om(\Pi') = \mathcal{D}om(\Pi)$,
- (u, u') is a $(b, P(u'))$-labeled edge of Π' iff (u, u') is a b-labeled edge of Π.

It is routine to check that the marked factor F'_u of T' in u coincides with the tree $N_{P(u)}\big(F_u, h(U)\big)$, where $U = \overrightarrow{R}^{\downarrow\overrightarrow{u}}|_{E'}$. By exploiting Proposition 10, one can compute an $M_{B'}$-type t' of F'_u on the grounds of the state $P(u)$ and the lookahead tree U. As regards the state $P(u)$, this can be retrieved from the label of the edge of Π that reaches the vertex u (if u is the root, then $P(u)$ is the initial state of N). This allows us to denote by $\overrightarrow{R}'(\overrightarrow{u})$ the $M_{B'}$-type of the marked factor F'_u of T', for any $u \in \mathcal{D}om(\Pi')$, and hence give $\overrightarrow{R}'$ the status of encoding of a retraction of T' with respect to Π and M.

As a matter of fact, the encoding $\overrightarrow{R}'$ can be obtained from $\overrightarrow{R}$ via the tree transducer with bounded lookahead $N' = (S', A', B', C', E', \lambda, \delta', \Omega', s'_0)$, where

- $S' = S$ (namely, the states of N' are exactly those of N);
- $A' = A_{in} \cup A_{out}$, where $A_{in} = B$ and $A_{out} = B'$;
- $C' = C_{in} \cup C_{out}$, where C_{in} is the set of all M^N_B-types and C_{out} is the set of all $M_{B'}$-types;
- λ maps an element $(b, s) \in B'$ to $b \in A'$;
- δ' maps a triple (s, U, b), where $s \in S'$, U is an A_{in}-labeled C_{in}-colored tree whose domain is a subset of E', and $b \in B'$, to the state $\mathcal{P}rj_2(b)$;
- Ω' maps a pair (s, U), where $s \in S'$ and U is an A_{in}-labeled C_{in}-colored tree whose domain is a subset of E', to the A_{out}-labeled B'-augmented C_{out}-colored regular tree U' such that
  i) $\mathcal{D}om(U') = \{\varepsilon\} \cup B'$,
  ii) $U'(\varepsilon)$ is the $M_{B'}$-type t' calculated as in Proposition 10 when we let $t = U(\varepsilon)$ and $(H_b)_{b\in B} = h(U)$,
  iii) for every $b \in B'$, $U'(b') = b'$;
- $s'_0 = s_0$ (namely, the initial state of N' is exactly that of N).

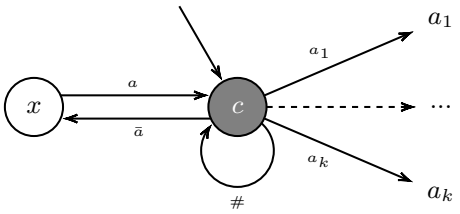

Figure 6. The 'slice' $G_{c,a}$ of a tree generator.

Since $\vec{R}$ belongs to the class of rank $n-1$ trees, which was supposed to be closed under applications of tree transducers with bounded lookahead, we immediately have that $\vec{R}'$ is a rank $n-1$ tree. Therefore, one can compute a footprint $\vec{\xi}'$ of $\vec{R}'$ on the grounds of the footprint $\vec{\xi}$ of $\vec{R}$. Finally, the function ξ' that maps a tree automaton M to the pair $(B', \vec{\xi}')$ is a footprint of T'. □

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 $\gamma(\bar{F})$.*

Proof. This result follows trivially from Proposition 7 since any tree insertion is specified by a tree T and by a tuple of colors $c_1, ..., c_n$ and since the basic $M^{\bar{t}}$-type of T is computable for the regular tree T. □

Theorem 5. *Given a 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 $\mathcal{B}ack\mathcal{U}nf$.*

Proof. In Section 4.2, we proved a similar result for a very specific case, namely, the semi-infinite line, which is a rank 0 tree. Precisely, we showed that a suitable retraction of the unfolding of the semi-infinite line with backward edges and loops can be obtained via a finite-state recoloring (without lookahead) of the semi-infinite line. The same proof technique can be applied to any rank n tree, thus proving that reducible trees are closed under $\mathcal{B}ack\mathcal{U}nf$.
Let T be an A-labeled C-colored rank n tree. We set $A' = A \cup \bar{A} \cup \{\#\}$ and we denote by G the extension of T with $\bar{A}$-labeled backward edges and $\#$-labeled loops, namely, $G = h^{-1}(T)$, where $h(a) = \{a\}$, $h(\bar{a}) = \bar{a}$, and $h(\#) = \{\varepsilon\}$. We further denote by T' the A'-labeled C-colored tree resulting from the unfolding of G, namely, $T' = \mathcal{U}nf(G) = \mathcal{B}ack\mathcal{U}nf(T)$. Note that for every vertex v in T', there is a (unique) corresponding path π_v in G starting from the source vertex (i.e., the root of T). Hence, we can denote by $\widetilde{v}$ the last vertex of G along the path π_v. Now, we define a factorization Π' of T', using only edge labels from A, as follows:

- $\mathcal{D}om(\Pi)$ is the set of all vertices v of T' for which there is no ancestor $v' \neq v$ of v in T' such that $\widetilde{v} = \widetilde{v}'$,
- (v, v') is an a-labeled edge in Π iff $(\widetilde{v}, \widetilde{v}')$ is an a-labeled edge in T.

Note that Π can have unbounded degree. However, for every pair of vertices u, u' in Π, if the sequences of labels along the access paths of u and u' in Π are the same, then (since the corresponding paths in G lead to the same vertex) we have that the two subtrees $(T')^{\downarrow u}$ and $(T')^{\downarrow u'}$ are isomorphic (and hence, the two marked factors $(T')^{+}_{\Pi}[u]$ and $(T')^{+}_{\Pi}[u']$ are isomorphic as well). Therefore, by identifying access paths having the same sequence of labels, we can define the encoding of a retraction of T' with respect to Π and any given tree automaton M. We shall prove that such an encoding is a rank n tree, from which it follows that T' is a rank $n+1$ tree.
Let $C' = A \cup C \cup \{x\}$, where x is a fresh color not belonging to $A \cup C$ (such a color x is used as an insertion variable for second-order tree substitutions). For every color $c \in C$ and for every label $a \in A$, let $\gamma_{c,a}$ be the regular tree insertion specified by the regular tree resulting from the unfolding of the rooted finite graph depicted in Figure 6, namely, $G_{c,a} = (V, (E_{a'})_{a' \in A}, C)$ such that

- $V = \{c, x\} \cup A$,
- $E_{\#} = \{(c, c)\}$, $E_a = \{(c, a), (x, c)\}$, $E_{\bar{a}} = \{(c, x)\}$, and for every $a' \neq a$, $E_{a'} = \{(c, a)\}$ and $E_{\bar{a}'} = \emptyset$,
- the coloring of $G_{c,a}$ is given by the identity function, namely, the vertex x is colored by x, the vertex c is colored by c, and each vertex $a \in A$ is colored by a,
- the root of $G_{c,a}$ is its unique c-colored vertex.

Now, notice that, if $T(\varepsilon) = c$ then, for any choice of $a \in A$, the marked factor of T' issued from the root of Π is isomorphic to the tree $\gamma_{c,a}(\emptyset)$, where $\emptyset$ denotes the empty tree. Moreover, it is easy to verify that, if u is a vertex of the factorization Π, u' is an a-successor of u in Π, and $T(\widetilde{u}') = c$, then the marked factor of T' issued from u' is isomorphic to the tree $\gamma_{c,a}(T'_u)$, where T'_u is the marked factor of T' in u. Thus, for every vertex u in Π, the marked factor of T' rooted at u depends only on the labels along the access path of u in Π. Now, by Theorem 4, given a regular tree insertion $\gamma_{c,a}$ and a tree automaton M, we can denote by $\gamma^M_{c,a}$ a computable function over the set $\mathscr{T}_M$ such that, for every A-augmented tree F and for every M-type t of F, $\gamma^M_{c,a}(t)$ is an M-type of $\gamma_{c,a}(F)$. This implies that

i) $\gamma^M_{c,a}(t_\emptyset)$ is an M-type of the marked factor of T' issued from the root of Π, provided that $T(\varepsilon) = c$ and $t_\emptyset$ is the basic M-type of the empty tree,

ii) for every a-labeled edge (u, u') in Π, $\gamma_{c,a}(t_u)$ is an M-type of the marked factor of T' rooted at u', provided that $T(\widetilde{u}') = c$ and t_u is an M-type of the marked factor of T' in u.

The above properties allow us to define the encoding $\overrightarrow{R}'$ of a retraction of T' with respect to Π and M as follows:

- $\mathcal{D}om(\overrightarrow{R}') = \mathcal{D}om(T)$,
- $\overrightarrow{R}'(\varepsilon) = \gamma^M_{T(\varepsilon),a}(t_\emptyset)$, for any arbitrary choice of $a \in A$,
- $\overrightarrow{R}'(ua) = \gamma^M_{T(ua),a}(\overrightarrow{R}'(u))$ for every $u \in \mathcal{D}om(T)$ and for every $a \in A$.

Clearly, such a retraction $\overrightarrow{R}'$ can be obtained by applying a finite-state recoloring to T. By Theorem 3, $\overrightarrow{R}'$ is a rank n tree whose footprint $\overrightarrow{\xi}'$ can be computed on the grounds of the footprint of T. This implies that the function ξ' that maps an A-augmented tree automaton M to the pair $(A, \overrightarrow{\xi}')$ is computable and it is a footprint of T'. Therefore, T' is a rank $n+1$ tree. □