Deciding Linear Height and Linear Size-To-Height Increase of Macro Tree Transducers

Paul Gallot
Universität Bremen, Germany

Sebastian Maneth
Universität Bremen, Germany

Keisuke Nakano
Tohoku University, Sendai, Japan

Charles Peyrat
ENS Paris-Saclay, France

Abstract

We present a novel normal form for (total deterministic) macro tree transducers (mtts), called "depth proper normal form". If an mtt is in this normal form, then it is guaranteed that each parameter of each state appears at arbitrary depths in the output trees of that state. Intuitively, if some parameter only appears at certain bounded depths in the output trees of a state, then this parameter can be eliminated by in-lining the corresponding output paths at each call site of that state. We use regular look-ahead in order to determine which of the paths should be in-lined. As a consequence of changing the look-ahead, a parameter that was previously appearing at unbounded depths, may be appearing at bounded depths for some new look-ahead; for this reason, our construction has to be iterated to obtain an mtt in depth-normal form. Using the normal form, we can decide whether the translation of an mtt has linear height increase or has linear size-to-height increase.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases automata, formal language theory, macro tree transducer, normal form

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.138

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

Related Version *Full Version*: https://arxiv.org/abs/2307.16500 [18]

Funding This work was partially supported by JSPS KAKENHI Grant Numbers 21K11744 and 22H00520.

Acknowledgements We thank the anonymous reviewers of a previous version of this paper for their critical comments.

1 Introduction

Tree transducers are fundamental devices in theoretical computer science. They generalize the finite state transductions from strings to (finite, ranked) trees and were invented in the 1970s in the context of compiler theory and mathematical linguistics. The most basic such transducers are the top-down tree transducer [24, 23] and the bottom-up tree transducer [25], see also [6]. These transducers traverse their input tree once, but may process subtrees in several copies. It is well known that these transducers have *linear height increase* ("LHI"), see e.g. [17].

In this paper we deal with a more powerful kind of tree transducer: the macro tree transducer [13] ("mtt"). Mtts can be seen as particularly simple functional programs on trees restricted to primitive recursion via (input) tree pattern matching. Alternatively, mtts can be seen as context-free tree grammars (introduced in [23] as "context-free dendrogrammars";

see also [15, 11, 12] and [19, Section 15]), the nonterminals of which are controlled by a top-down tree storage (in the spirit of [7]). It is an open problem, if it is decidable for a given mtt whether or not its translation can be realized by a top-down tree transducer (with "origin" semantics, this is decidable [14]). As mentioned above, it is a necessary condition for the mtt to have linear height increase ("LHI"). This raises the question, can we decide for a given mtt, whether or not its translation has LHI? Here we give an affirmative answer to this question. It is also an open problem, if it is decidable for a given mtt whether or not its translation can be realized by an attributed tree transducer [20, 21, 16] ("att"). It is well-known that atts have *linear size-to-height increase* ("LSHI"), see, e.g., [17]. This raises the question, can we decide for a given mtt, whether or not its translation is of LSHI? We give an affirmative answer. Note that it was conjectured in [10] that the methods of that paper could be adapted to give such an affirmative answer.

Let us now discuss our results in more detail. To decide both the LHI and LSHI properties, we introduce a new normal form called "depth proper". An mtt is depth proper if each parameter of every (reachable) state appears at infinitely many different depths (for different input trees). The idea of our construction is to eliminate parameters that only appear at bounded depths; we use regular look-ahead to determine which bounded paths to output at a given moment. Since in this way we may generate output "earlier" than the original transducer, new "helper states" need to be introduced which continue the translation at the correct input nodes. Both issues, the change of look-ahead and the introduction of new states may cause the newly constructed transducer *not* to be depth proper. For this reason our construction has to be iterated.

To understand the idea of the construction, let us examine a small example. We consider input and output trees over a binary symbol f and the nullary symbol a and an mtt with the following rules.

$$\begin{array}{lcl@{\qquad}lcl}
q_0(f(x_1,x_2)) & \to & f(q_1(x_2), q_2(x_2, q_{\mathrm{id}}(x_1))) & q_2(f(x_1,x_2),y_1) & \to & f(y_1, q_0(x_2)) \\
q_0(a) & \to & a & q_2(a,y_1) & \to & f(y_1,a) \\
q_1(f(x_1,x_2)) & \to & q_{\mathrm{id}}(x_1) & q_{\mathrm{id}}(f(x_1,x_2)) & \to & f(q_{\mathrm{id}}(x_1), q_{\mathrm{id}}(x_2)) \\
q_1(a) & \to & a & q_{\mathrm{id}}(a) & \to & a
\end{array}$$

The transducer realizes the following translation:

$$f(t_1, f(t_2, \underbrace{f(t_3, f(t_4, \dots))}_{t}))) \Rightarrow f(t_2, f(t_1, f(t_4, f(t_3, \dots))))$$

The following shows in detail how the rules of the mtt are applied to produce as output first the tree t_2 and then the tree t_1:

$$\begin{array}{rl}
 & q_0(f(t_1, f(t_2,t))) \\
\Rightarrow_{\text{first rule}} & f(q_1(f(t_2,t)), q_2(f(t_2,t), q_{\mathrm{id}}(t_1))) \\
\Rightarrow_{\text{second rule}} & f(q_{\mathrm{id}}(t_2), q_2(f(t_2,t), q_{\mathrm{id}}(t_1))) \\
\Rightarrow^*_{\text{last two rules}} & f(t_2, q_2(f(t_2,t), q_{\mathrm{id}}(t_1))) \\
\Rightarrow_{\text{third rule}} & f(t_2, f(q_{\mathrm{id}}(t_1), q_0(t)) \\
\Rightarrow^*_{\text{last two rules}} & f(t_2, f(t_1, q_0(t)))
\end{array}$$

An mtt uses two different types of variables. The first argument of each state in the left-hand side of every rule of the mtt is always of type input tree and performs pattern matching on the current node of the input tree. For this pattern matching, *input variables* of the form x_1 and x_2 are used to denote the first and second subtree of the current input node, respectively. The (possible) next arguments of a state in the left-hand side of a rule are the (accumulating) *parameters* $y_1, y_2, \dots$ of that state. In our example, only the state q_2

has exactly one parameter y_1. Parameters are used to built up output trees in a bottom-up fashion. In the example, consider the application of the first rule, i.e., going from line one to line two in the previous display: here the first (and only) parameter y_1 of state q_2 is instantiated by the output tree that is produced by the call $q_{\text{id}}(t_1)$.

Observe that state q_2 is *not* depth proper: each tree that it outputs is of the form $f(y_1, t)$ where t does not contain the parameter y_1. The idea of our construction is to replace each occurrence of state q_2 in the right-hand side of any rule by this tree "fragment", where at the position of t there will be the new "helper state" $[q_2, 2]$. The path "2" indicates that this state should produce the tree at the second child position of the output tree produced by q_2. We obtain the following (the rules of q_0, q_1 and input a are as before):

$$
\begin{array}{lcl}
q_0(f(x_1,x_2)) & \to & f(q_1(x_2), f(q_{\text{id}}(x_1), [q_2,2](x_2))) \\
q_1(f(x_1,x_2)) & \to & q_{\text{id}}(x_1) \\
[q_2,2](f(x_1,x_2)) & \to & q_0(x_2) \\
[q_2,2](a) & \to & a \\
q_{\text{id}}(f(x_1,x_2)) & \to & f(q_{\text{id}}(x_1), q_{\text{id}}(x_2)) \\
q_{\text{id}}(a) & \to & a
\end{array}
$$

It should be clear that the new transducer is equivalent to the original one. Moreover, the new transducer uses no parameters whatsoever, therefore it is depth proper. Given a depth proper mtt, we can decide the LSHI property as follows. We consider input trees which contain exactly one special marked input leaf (it will be marked by a state p of the look-ahead automaton, to act as a place-holder for any input tree for which the look-ahead automaton arrives in state p). For such input trees, the mtt produces output trees which only contain nested state calls to the special input leaf. The original transducer has LSHI if and only if the range of this transducer is finite (which is known to be decidable [5]). In a similar way we can decide LHI: here we consider input trees with multiple marked input leaves. To show that if such ranges are not finite, then the translation does not have LSHI (or LHI), is done via pumping arguments (which use depth properness); these pumping arguments are technically rather involved, but are (somewhat) similar to the ones used in [10] to show that it is decidable whether or not an mtt has *linear size increase* (LSI).

If we restrict the translations of mtts to LSI, then we obtain exactly the MSO definable tree translations [10]. Note that this class of translation has recently been characterized by new models of tree transducers, the *streaming tree transducer* [2] and even more recently the *register tree transducer* [3]. The LSI property is decidable for mtts (it can even be decided for compositions of mtts, and if so, then the translation is effectively MSO definable [8]). To decide LSI, the given mtt is first transformed into "proper" normal form. Properness guarantees that (1) each state (except possibly the initial state) produces infinitely many output trees and that (2) each parameter of a state is instantiated with infinitely many distinct argument trees. Note that input properness is a generalization of the proper form of [1]. Once in proper normal form, it suffices to check if the transducer is "finite copying". This means that (a) each node of each input tree is processed only a bounded number of times and that (b) each parameter of every state is copied only a bounded number of times.

2 Preliminaries

The set $\{0, 1, \dots\}$ of natural numbers is denoted by $\mathbb{N}$. For $k \in \mathbb{N}$ we denote by $[k]$ the set $\{1, \dots, k\}$; thus $[0] = \emptyset$. A ranked alphabet (set) consists of an alphabet (set) Σ together with a mapping $\text{rank}_\Sigma : \Sigma \to \mathbb{N}$ that assigns to each symbol $\sigma \in \Sigma$ a natural number called its "rank". We write $\sigma^{(k)} \in \Sigma$ to denote that $\sigma \in \Sigma$ and $\text{rank}_\Sigma(\sigma) = k$. By $\Sigma^{(k)}$ we denote the symbols of Σ that have rank k.

The set T_Σ of (finite, ranked, ordered) trees over Σ is the smallest set of strings S such that if $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \dots, s_k \in S$, then also $\sigma(s_1, \dots, s_k) \in S$. We write σ instead of $\sigma()$. For a tree $s = \sigma(s_1, \dots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \dots, s_k \in T_\Sigma$, we define the set $V(s) \subseteq \mathbb{N}^*$ of nodes of s as $\{\varepsilon\} \cup \{iu \mid i \in [k], u \in V(s_i)\}$; thus, nodes are strings over positive integers, where ε denotes the root node of s, and for a node u, ui denotes the i-th child of u. For $u \in V(s)$ we denote by $s[u]$ the label of u in s and by s/u the subtree rooted at u. Formally, let $s = \sigma(s_1, \dots, s_k)$ and define $s[\epsilon] = \sigma$, $s[iu] = s_i[u]$, $s/\varepsilon = s$, and $s/iu = s_i/u$ for $\sigma \in \Sigma^{(k)}$, $k \geq 0$, $s_1, \dots, s_k \in T_\Sigma$, $i \geq 1$ and $u \in V(s_i)$ such that $iu \in V(s)$.

We fix two special sets of symbols: the set $X = \{x_1, x_2, \dots\}$ of variables and the set $Y = \{y_1, y_2, \dots\}$ of parameters. For $k \geq 1$ let $X_k = \{x_1, \dots, x_k\}$ and $Y_k = \{y_1, \dots, y_k\}$. Let A be a set that is disjoint from Σ. Then the set $T_\Sigma(A)$ of trees over Σ indexed by A is defined as $T_{\Sigma'}$ where $\Sigma' = \Sigma \cup A$ and $\mathrm{rank}_{\Sigma'}(a) = 0$ for $a \in A$ and $\mathrm{rank}_{\Sigma'}(\sigma) = \mathrm{rank}_\Sigma$ for $\sigma \in \Sigma$.

For a ranked alphabet Σ and a set A the ranked set $\langle \Sigma, A \rangle$ consists of all symbols $\langle \sigma, a \rangle$ with $\sigma \in \Sigma$ and $a \in A$; the rank of $\langle \sigma, a \rangle$ is defined as $\mathrm{rank}_\Sigma(\sigma)$.

2.1 Tree Substitution

Let Σ be a ranked alphabet and let $s, t \in T_\Sigma$. For $u \in V(s)$ we define the tree $s[u \leftarrow t]$ that is obtained from s by replacing the subtree rooted at node u by the tree t. Let $\sigma_1, \dots, \sigma_n \in \Sigma^{(0)}$, $n \geq 1$ be pairwise distinct symbols and let $t_1, \dots, t_n \in T_\Sigma$. Then $t[\sigma_i \leftarrow t_i \mid i \in [n]]$ is the tree obtained from t by replacing each occurrence of σ_i by the tree t_i. We have defined trees as particular strings, and this is just ordinary string substitution (because we only replace symbols of rank zero). We refer to this as "first-order tree substitution".

In "second-order tree substitution" it is possible to replace internal nodes u (of a tree s) by new trees. These new trees use parameters to indicate where the "dangling" subtrees s/ui of the node u are to be placed. Let $\sigma_1 \in \Sigma^{(k_1)}, \dots, \sigma_n \in \Sigma^{(k_n)}$ be pairwise distinct symbols with $n \geq 1$, $k_1, \dots, k_n \in \mathbb{N}$, and $t_i \in T_\Sigma(Y_{k_i})$ for $i \in [n]$. Let $s \in T_\Sigma$. Then $s[\![\sigma_i \leftarrow t_i \mid i \in [n]]\!]$ denotes the tree that is inductively defined as (abbreviating $[\![\sigma_i \leftarrow t_i \mid i \in [n]]\!]$ by $[\![\dots]\!]$) follows: for $s = \sigma(s_1, \dots, s_k)$, if $\sigma \notin \{\sigma_1, \dots, \sigma_n\}$ then $s[\![\dots]\!] = \sigma(s_1[\![\dots]\!], \dots, s_k[\![\dots]\!])$ and if $\sigma = \sigma_j$ for some $j \in [n]$ then $s[\![\dots]\!] = t_j[y_i \leftarrow s_i[\![\dots]\!] \mid i \in [k]]$.

2.2 Macro Tree Transducers

A (deterministic bottom-up) *tree automaton* A is given by a tuple (P, Σ, h) where P is a finite set of states, Σ is a ranked alphabet, and h is a collection of mappings $h_\sigma : P^k \to P$ with $\sigma \in \Sigma^{(k)}$ and $k \geq 0$. The extension of h to a mapping $\hat{h} : T_\Sigma \to P$ is defined recursively as $\hat{h}(\sigma(s_1, \dots, s_k)) = h_\sigma(\hat{h}(s_1), \dots, \hat{h}(s_k))$ for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \dots, s_k$. For every $p \in P$ we define the subset L_p of trees in T_Σ as $\{s \in T_\Sigma \mid \hat{h}(s) = p\}$. We assume that $L_p \neq \emptyset$ for every $p \in P$.

A (total deterministic) *macro tree transducer with (regular) look-ahead* ("mttr") M is given by a tuple $(Q, P, \Sigma, \Delta, q_0, R, h)$, where

- Q is a ranked alphabet of *states*,
- Σ and Δ are ranked alphabet of *input* and *output symbols*,
- (P, Σ, h) is a tree automaton (called the *look-ahead automaton* of M),
- $q_0 \in Q^{(0)}$ is the *initial state*, and
- R is the *set of rules*, where for each $q \in Q^{(m)}$, $m \geq 0$, $\sigma \in \Sigma^{(k)}, k \geq 0$, and $p_1, \dots, p_k \in P$ there is exactly one rule of the form

  $$\langle q, \sigma(x_1 : p_1, \dots, x_k : p_k) \rangle (y_1, \dots, y_m) \to t$$

  with $t \in T_{\Delta \cup \langle Q, X_k \rangle}(Y_m)$.

  The right-hand side t of such a rule is denoted by $\mathrm{rhs}_M(q, \sigma, \langle p_1, \dots, p_k \rangle)$

We use a notation that is slightly different from the one used in the Introduction: instead of, e.g., $q_2(x_2, q_{\mathrm{id}}(x_1))$ we write $\langle q_2, x_2\rangle(\langle q_{\mathrm{id}}, x_1\rangle)$. Thus, we use angular brackets $\langle \ldots \rangle$ to indicate a state call on an input subtree, and use round brackets (after the angular brackets), to indicate the parameter arguments of the particular state call.

The semantics of an mttr M (as above) is defined as follows. We define the derivation relation $\Rightarrow_M$ as follows. For two trees $\xi_1, \xi_2 \in T_{\Delta\cup\langle Q,T_\Sigma\rangle}(Y)$, $\xi_1 \Rightarrow_M \xi_2$ if there exists a node u in ξ_1 with $\xi_1/u = \langle q, s\rangle(t_1, \ldots, t_m)$, $q \in Q^{(m)}$, $m \geq 0$, $s = \sigma(s_1, \ldots, s_k)$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, $s_1, \ldots, s_k \in T_\Sigma$, $t_1, \ldots, t_m \in T_{\Delta\cup\langle Q,T_\Sigma\rangle}(Y)$, and $\xi_2 = \xi_1[u \leftarrow \xi]$ where ξ equals

$$\zeta[\![\langle q', x_i\rangle \leftarrow \langle q', s_i\rangle \mid q' \in Q, i \in [k]]\!][y_j \leftarrow t_j \mid j \in [m]]$$

and $\zeta = \mathrm{rhs}_M(q, \sigma, \langle \hat{h}(s_1), \ldots, \hat{h}(s_k)\rangle)$. Since M is total deterministic (i.e., for every state q, input symbol $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and look-ahead states $p_1, \ldots, p_k$, M contains exactly one corresponding rule) there is for every ξ_1 a unique tree $\xi' \in T_\Delta(Y)$ such that $\xi_1 \Rightarrow_M^* \xi'$. For every $q \in Q^{(m)}$, $m \geq 0$ and $s \in T_\Sigma$ we define the *q-translation of s*, denoted by $M_q(s)$, as the unique tree t in $T_\Delta(Y_m)$ such that $\langle q, s\rangle(y_1, \ldots, y_m) \Rightarrow_M^* t$. We denote the *translation realized by M* also by M, i.e., $M = M_{q_0}$ and for every $s \in T_\Sigma$, $M(s) = M_{q_0}(s)$ is the unique tree $t \in T_\Delta$ such that $\langle q_0, s\rangle \Rightarrow_M^* t$.

Let M be an mttr as before. We define *the extension $\widehat{M}$ of M* which can also process look-ahead states at leaves of input trees. Let $\widehat{M} = (Q, P, \hat{\Sigma}, \hat{\Delta}, q_0, R \cup \hat{R}, h \cup h')$ where $\hat{\Sigma} = \Sigma \cup \{p^{(0)} \mid p \in P\}$ and $\hat{\Delta} = \Delta \cup \{\langle q, p\rangle^{(m)} \mid q \in Q^{(m)}, p \in P, m \geq 0\}$. For every $q \in Q^{(m)}$, $m \geq 0$, and $p \in P$ we let $h(p) = p$ and we let the rule $\langle q, p\rangle(y_1, \ldots, y_m) \to \langle q, p\rangle(y_1, \ldots, y_m)$ be in $\hat{R}$; note that the $\langle q, p\rangle$ on the right-hand side of this rule is an output symbol. For the original transducer M we say that the pair (q, p) is *reachable (in M)* if there is an input tree $s \in T_{\hat{\Sigma}}$ such that $\langle q, p\rangle$ occurs in $\widehat{M}(s)$. Clearly it is decidable for a given pair (q, p), whether or not it is reachable; this is because (1) inverse translations of mttrs effectively preserve regularity [13, 22], (2) the set of all trees in $T_{\hat{\Delta}}$ that contain at least one occurrence of $\langle q, p\rangle$ is (effectively) regular, and (3) emptiness of regular tree languages is decidable [4].

We say that M is *nondeleting*, if for every state $q \in Q^{(m)}$, $\sigma \in \Sigma^{(k)}, k \geq 0$, $p_1, \ldots, p_k \in P$, and $j \in [m]$, there is at least one occurrence of y_j in $\mathrm{rhs}_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$. The next proposition is proved in [9, Lemma 6.7] (for mttrs that do not copy parameters, but the proof works analogously for arbitrary mttrs).

▶ **Proposition 1.** *For every mttr, an equivalent nondeleting mttr M can be constructed. For every state q of a nondeleting mttr M of rank m and for every $j \in [m]$: $M_q(s)$ contains at least one occurrence of y_j, for every $s \in T_\Sigma$.*

It is well known that the finiteness of ranges of compositions of mttrs is decidable [5]. A (partial nondeterministic) top-down tree transducer with look-ahead ("topr" for short) is an mttr as before, where $Q = Q^{(0)}$ and R may contain none or several rules for each given q and σ.

▶ **Proposition 2.** *([5, Theorem 4.5]) For a given composition of mttrs and (partial nondeterministic) toprs it is decidable whether or not the range of the composition is finite. In the case of finiteness, the range can be constructed.*

3 Depth Proper Normal Form

The depth proper normal form requires that each parameter of each state q occurs at unbounded depth in the output trees of that state (for each given look-ahead state p such that (q, p) is reachable). Formally, let q be a state of rank $m \geq 1$, $j \in [m]$, and $p \in P$. If

(q,p) is reachable, then for every natural number n there must exist an input tree $s_n \in L_p$ such that y_j occurs at depth $> n$ in the tree $M_q(s_n)$. Conversely, we say that parameter y_j is *depth-bounded* for q and p if there exists an n for which no such input tree $s_n \in L_p$ exists; more generally, we say that $Z \subseteq Y_m$ is *depth-bounded* for q and p, if each $y \in Z$ is depth-bounded for q and p.

If Z is depth-bounded for q and p, then there are only finitely many output paths in the trees in $M_q(L_p)$ under which the parameters from Z occur. The *Z-skeleton* of an arbitrary tree t is obtained from t by replacing each top-most node u such that t/u does not contain any occurrence of a parameter from Z by some symbol. Clearly, Z is depth-bounded for q and p if and only if the Z-skeleta of all trees in $M_q(L_p)$ form a finite set.

Let Δ be an arbitrary ranked alphabet, $m \geq 1$, $t \in T_\Delta(Y_m)$, and $Z \subseteq Y_m$. Let us write $\mathsf{ps}(t) \subseteq Y_m$ for the set of parameters occurring in t. Let us now be more specific as to which symbols replace the top-most nodes u of t such that $\mathsf{ps}(t/u) \cap Z = \emptyset$. Since in our construction later we will want to obtain a transducer that is nondeleting, it will be helpful to know which parameters appear in a given deleted tree. Therefore we replace such nodes u by the set $\mathsf{ps}(t/u)$. We denote by $\lfloor t \rfloor_Z$ the Z-skeleton of t and define it inductively as follows (where $\delta \in \Delta$):

$$\lfloor t \rfloor_Z = \begin{cases} t & \text{if } t \in Z \\ \delta(\lfloor t_1 \rfloor_Z, \dots, \lfloor t_n \rfloor_Z) & \text{if } \mathsf{ps}(t) \cap Z \neq \emptyset \text{ and } t = \delta(t_1, \dots, t_n) \\ \mathsf{ps}(t) & \text{if } \mathsf{ps}(t) \cap Z = \emptyset. \end{cases}$$

The definition of $\lfloor t \rfloor_Z$ is extended to sets L of trees as $\lfloor L \rfloor_Z = \{\lfloor t \rfloor_Z \mid t \in L\}$. We call *$Y$-nodes* the nodes u in $V(\lfloor t \rfloor_Z)$ such that $\lfloor t \rfloor_Z /u = Z' \subseteq Y_m$. We denote by $\mathcal{U}(\lfloor t \rfloor_Z)$ the set of Y-nodes on $\lfloor t \rfloor_Z$. The notion of parameters in a tree naturally extends to Z-skeleta with, for a Y-node labeled Z': $\mathsf{ps}(Z') = Z'$. The proof of the next lemma is straightforward by induction on t (see full version of this paper: Lemma 3 in [18]).

▶ **Lemma 3.** *Let Δ be a ranked alphabet, $m \geq 1$, $Z \subseteq Y_m$, and $t \in T_\Delta(Y_m)$. (1) $t = \lfloor t \rfloor_Z [u \leftarrow t/u \mid u \in \mathcal{U}(\lfloor t \rfloor_Z)]$. (2) $\mathsf{ps}(\lfloor t \rfloor_Z) = \mathsf{ps}(t)$.*

Finally, we define depth properness for mtts with look-ahead.

▶ **Definition 4.** *The mttr $M = (Q, P, \Sigma, \Delta, q_0, R, h)$ is in* depth proper normal form *(or, synonymously, M is* depth proper*) if for every $q \in Q^{(m)}$, $m \geq 1$, and $p \in P$ it holds that if (q,p) is reachable, then $\lfloor M_q(L_p) \rfloor_{\{y_j\}}$ is infinite for all $j \in [m]$.*

From now on we will want to make use of the following definitions:

$$\begin{aligned} F_p &= \{q \in Q^{(m)} \mid \exists j \in [m], \lfloor M_q(L_p) \rfloor_{\{y_j\}} \text{ is finite}\} \\ Y(q,p) &= \{y_j \mid j \in [\mathrm{rank}_Q(q)] \text{ such that } \lfloor M_q(L_p) \rfloor_{\{y_j\}} \text{ is finite}\} \end{aligned}$$

It should be clear that $\lfloor M_q(L_p) \rfloor_{Y(q,p)}$ is finite for every q and p, as stated in the next lemma (the proof is in the full version: Lemma 5 in [18]).

▶ **Lemma 5.** *Let M be an mtt, q a state of M, and p a look-ahead state of M. Then $\lfloor M_q(L_p) \rfloor_{Y(q,p)}$ is finite.*

3.1 Construction of the Normal Form and Examples

Let M be an mttr as before. We assume that M is nondeleting (which is justified by Proposition 1). The idea of the construction is as follows. First, we determine all reachable pairs (q,p) such that $Y(q,p) \neq \emptyset$. Let (q,p) be such a pair and let $Z = Y(q,p)$. An occurrence of $\langle q, x_i\rangle$ in a right-hand side $\mathrm{rhs}_M(q', \sigma, \langle p_1, \dots, p_k\rangle)$ such that $p_i = p$ is called a (q,p)-*call*. Our aim is to replace each (q,p)-call by an appropriate tree from $\lfloor M_q(L_p)\rfloor_Z$. Just which tree is the appropriate one will be determined by regular look-ahead. Moreover, such trees should be modified not to contain leaf nodes labeled by subsets of Y: such nodes will be replaced by calls of new "helper states".

▶ **Definition 6.** *Let $M = (Q, P, \Sigma, \Delta, q_0, R, h)$ be a nondeleting mttr that is not depth proper. We construct the new mttr $\pi(M) = (Q \cup H, P', \Sigma, \Delta, q_0, R', h')$. For every $q \in Q^{(m)}$ and $p \in P$ such that $Y(q,p) \neq \emptyset$, H contains the following set of helper states:*

$$\{[q,p,t,u]^{(|U|)} \mid t \in \lfloor M_q(L_p)\rfloor_{Y(q,p)}, u \in V(t), t/u = U \subseteq Y_m\}$$

and P' contains (p, φ) for any function φ that assigns to each $q \in F_p$ a tree in $\lfloor M_q(L_p)\rfloor_{Y(q,p)}$. Observe that H and P' are well defined, because $\lfloor M_q(L_p)\rfloor_{Y(q,p)}$ is finite by Lemma 5. Note that $|U| \leq |Y_m \setminus Y(q,p)|$; since $Y(q,p)$ is non-empty this implies that the rank of each helper state is at most $(r-1)$, where r is the maximal rank of the states in Q.

For every $q \in Q^{(m)}$, $m \geq 0$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $(p_1, \varphi_1), \dots, (p_k, \varphi_k) \in P'$ we let the rule

$$\langle q, \sigma(x_1 : (p_1, \varphi_1), \dots, x_k : (p_i, \varphi_i)\rangle(y_1, \dots, y_m) \to rhs_M(q, \sigma, \langle p_1, \dots, p_k\rangle)[\![.]\!]$$

be in R', where the second-order tree substitution $[\![.]\!]$ is defined as follows.

$$[\![.]\!] = [\![\langle q', x_i\rangle \leftarrow \varphi_i(q')\big[u \leftarrow [q', p_i, \varphi_i(q'), u](y_{j_1}, \dots, y_{j_n}) \mid \varphi_i(q')/u = \{y_{j_1}, \dots, y_{j_n}\}, j_1 < \dots < j_n\big] \mid q' \in F_{p_i}, i \in [k]]\!].$$

We define $h'_\sigma((p_1, \varphi_1), \dots, (p_k, \varphi_k)) = (p, \varphi)$ where $p = h_\sigma(p_1, \dots, p_k)$ and, using the special second-order substitution $[\![\dots]\!]^\$$ from Definition 8, for every $q \in F_p$,

$$\varphi(q) = \Big\lfloor rhs_M(q, \sigma, \langle p_1, \dots, p_k\rangle)[\![\langle q', x_i\rangle \leftarrow \varphi_i(q') \mid q' \in F_{p_i}, i \in [k]]\!]^\$ \Big\rfloor_{Y(q,p)}.$$

The special second-order substitution $[\![\dots]\!]^\$$ is the same as the normal one except that the special first-order substitution is applied for each involved first-order substitution. The special first-order substitution is the same as the normal one except that it gives special treatment to Y-nodes which are replaced by Y-nodes containing all parameters occurring in trees to be substituted for the parameters in the original Y-nodes.

Note that, when $L_p \neq \emptyset$ for all $p \in P$, then $L_{(p,\phi)} \neq \emptyset$ for all $(p, \phi) \in P'$. For every helper state $[q,p,t,u] \in H^{(n)}$, $n \geq 0$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $(p_1, \varphi_1), \dots, (p_k, \varphi_k) \in P'$ such that $h_\sigma(p_1, \dots, p_k) = p$ we let the rule

$$\langle [q,p,t,u](\sigma(x_1 : (p_1, \varphi_1), \dots, x_k : (p_k, \varphi_k)))\rangle(y_1, \dots, y_n) \to \xi/u[y_{j_\nu} \leftarrow y_\nu \mid \nu \in [n]]$$

be in R' where $t/u = \{y_{j_1}, \dots, y_{j_n}\}$, $j_1 < \dots < j_n$, $\xi = rhs_M(q, \sigma, \langle p_1, \dots, p_k\rangle)[\![.]\!]$, and $[\![.]\!]$ is the substitution from above.

We now show how the depth proper normal form is achieved using an example. An additional example (which makes more interesting use of helper states) can be found in the full version of this paper: at page 21 in [18]. Let $M = (Q, \{p\}, \Sigma, \Delta, q_0, R, h_0)$ with $Q = \{q_0^{(0)}, q_1^{(1)}, q_2^{(2)}\}$, $\Sigma = \{a^{(1)}, b^{(1)}, e^{(0)}\}$, and $\Delta = \{f^{(2)}, g^{(1)}, e^{(0)}\}$ be an mttr where $(\Sigma, \{p\}, h_0)$ with $L_p = T_\Sigma$ and R consists of these rules:

$$\begin{aligned} \langle q_1, a(x)\rangle(y_1) &\to \langle q_2, x\rangle(y_1, \langle q_1, x\rangle(y_1)) & \langle q_2, a(x)\rangle(y_1, y_2) &\to f(y_1, \langle q_1, x\rangle(g(y_2))) \\ \langle q_1, b(x)\rangle(y_1) &\to y_1 & \langle q_2, b(x)\rangle(y_1, y_2) &\to f(y_2, y_1) \\ \langle q_1, e\rangle(y_1) &\to g(y_1) & \langle q_2, e\rangle(y_1, y_2) &\to f(y_2, y_1) \end{aligned}$$

We suppose that the q_0-rules are defined so that all states are reachable. Now we have $F_p = \{q_2\}$, $Y(q_2, p) = \{y_1\}$, and $\lfloor M_{q_2}(L_p) \rfloor_{\{y_1\}} = \{t_1, t_2\}$ with $t_1 = f(y_1, \{y_2\})$ and $t_2 = f(\{y_2\}, y_1)$. As before, we can rewrite q_2-calls with the skeleta, but since there are two possibilities t_1 and t_2, we need to separate the rules according to the input using the look-ahead. In general, F_p contains several states each of which may have multiple skeleta, so each look-ahead contains a finite map from F_p to skeleta. Let $\varphi_1 = \{q_2 \mapsto t_1\}$ and $\varphi_2 = \{q_2 \mapsto t_2\}$ such that $L_{p,\varphi_1} = \{a(s) \mid s \in \mathcal{T}_\Sigma\}$ and $L_{p,\varphi_2} = \{b(s) \mid s \in \mathcal{T}_\Sigma\} \cup \{e\}$. The (q_1, a)-rule containing a q_2-call is separated as

$$\begin{aligned} &\langle q_1, a(x : (p, \varphi_1))\rangle(y_1) \to f(y_1, \langle [q_2, p, t_1, 2], x\rangle(\langle q_1, x\rangle(y_1))) \\ &\langle q_1, a(x : (p, \varphi_2))\rangle(y_1) \to f(\langle [q_2, p, t_2, 1], x\rangle(\langle q_1, x\rangle(y_1)), y_1) \end{aligned}$$

where $[q_2, p, t_1, 2]$ and $[q_2, p, t_2, 1]$ are helper states. Each helper state has rank 1 because the corresponding node in the skeleton is a Y-node of length 1. The arguments of the call are inherited from the arguments of the original q_2-call that occur in the sequence. For example, $\langle [q_2, p, t_1, 2], x\rangle$ is called with $\langle q_1, x\rangle(y_1)$ since t_1 has a Y-node $\{y_2\}$ and the original q_2-call has $\langle q_1, x\rangle(y_1)$ as the second argument. The rules of these helper states are constructed from the original q_2-rule with substitution (which causes nothing since no states in F_p are called) and extracting a subtree at the Y-node, that is,

$$\begin{aligned} \langle [q_2, p, t_1, 2], a(x : (p, \varphi))\rangle(y_1) &\to \langle q_1, x\rangle(g(y_1)) \\ \langle [q_2, p, t_2, 1], b(x : (p, \varphi))\rangle(y_1) &\to y_1 \\ \langle [q_2, p, t_2, 1], e\rangle(y_1) &\to y_1 \end{aligned}$$

where $\varphi \in \{\varphi_1, \varphi_2\}$ and we had to rename the parameter y_2 into y_1 (because the helper states only refer to y_2). Note that rules for $([q_2, p, t_1, 2], b)$, $([q_2, p, t_1, 2], e)$ and $([q_2, p, t_2, 1], a)$ do not have to be considered. These rules are not referred because the states are never called with the input symbols due to their look-ahead. For example, the $[q_2, p, t_1, 2]$-call occurs only in the (q_1, a)-rule with $x \in L_{p,\varphi_1}$ in which the root symbol cannot be b.

Thereby we have been able to remove every call of states in F_p. However, new improper states may be generated by the separation of rules because of the look-ahead introduction. In fact, we have $F_{p,\varphi_2} = \{q_1, q_2, [q_2, p, t_2, 1]\}$ in the example above. Since every q_2-call has already been removed in the previous step, we have to apply the same technique again for the calls of q_1 and $[q_2, p, t_2, 1]$. We have $Y(q_1, (p, \varphi_2)) = \{y_1\}$ and $Y([q_2, p, t_2, 1], (p, \varphi_2)) = \{y_1\}$. Moreover $\lfloor M'_{q_1}(L_{p,\varphi_2}) \rfloor_{\{y_1\}} = \{y_1, g(y_1)\}$ and $\left\lfloor M'_{[q_2,p,t_2,1]}(L_{p,\varphi_2}) \right\rfloor_{\{y_1\}} = \{y_1\}$.

Look-ahead has to be introduced to determine which skeleton to output. Two maps over F_{p,φ_2}, except for q_2 whose call has already been removed, are defined: $\varphi_3 = \{q_1 \mapsto y_1, [q_2, p, t_2, 1] \mapsto y_1\}$ and $\varphi_4 = \{q_1 \mapsto g(y_1), [q_2, p, t_2, 1] \mapsto y_1\}$ such that $L_{p,\varphi_2,\varphi_3} = \{b(s) \in L_{p,\varphi_2} \mid s \in \mathcal{T}_\Sigma\}$ and $L_{p,\varphi_2,\varphi_3} = \{e\}$. The (q_1, a)- and $([q_2, p, t_1, 2], a)$-rules with look-ahead φ_2 which contains a q_2-call are separated as follows:

$$\begin{array}{lcl}
\langle q_1, a(x:(p,\varphi_2,\varphi_3))\rangle(y_1) & \to & f(y_1,y_1) \\
\langle [q_2,p,t_1,2], a(x:(p,\varphi_2,\varphi_3))\rangle(y_1) & \to & g(y_1) \\
\langle q_1, a(x:(p,\varphi_2,\varphi_4))\rangle(y_1) & \to & f(g(y_1),y_1) \\
\langle [q_2,p,t_1,2], a(x:(p,\varphi_2,\varphi_4))\rangle(y_1) & \to & g(g(y_1))
\end{array}$$

The resulting mttr is depth proper.

3.2 Correctness Proof and Termination of Iteration

Here we prove the correctness of transducer $\pi(M)$ that was defined in Definition 4. Lemma 7 establishes the correctness of the look-ahead, relates the states of $\pi(M)$ to those of M, and shows that the transducer $\pi(M)$ is nondeleting. The latter is needed, so that the construction of π can be carried out iteratively (recall from Definition 4 that M is required to be nondeleting in order to construct $\pi(M)$). To prove Point (2) we use a "special" kind of second-order tree substitution which replaces Y-nodes by new Y-nodes consisting of parameters in the output trees that would have been substituted for the parameters in the original Y-node (Definition 8).

▶ **Lemma 7.** *Let M be a nondeleting mttr and $N = \pi(M)$ be the mttr of Definition 6, both with the tuples as in that definition. Let $s \in T_\Sigma$ with $\hat{h}'(s) = (p, \varphi)$.*
(1) *$p = \hat{h}(s)$,*
(2) *$\forall q \in F_p$: $\varphi(q) = \lfloor M_q(s) \rfloor_{Y(q,p)}$,*
(3) *$\forall q \in Q$: $N_q(s) = M_q(s)$,*
(4) *$\forall q \in F_p$ and $u \in V(t)$ with $t = \varphi(q)$ and $t/u = \{y_{j_1}, \dots, y_{j_n}\}$ with $j_1 < \dots < j_n$: $N_{[q,p,t,u]}(s) = M_q(s)/u[y_{j_\nu} \leftarrow y_\nu \mid \nu \in [n]]$, and*
(5) *the mttr N is nondeleting.*

We first need a small lemma showing that the skeleton of the output of an mttr M can be directly computed from given a input tree by modifying the rules of M. For this lemma we first need to define how to compute second-order substitutions of skeleta, which will be used for the modification of the right-hand sides of rules. We do so on a *nondeleting* mttr M, i.e. such that states always use all their parameters.

▶ **Definition 8.** *Let Γ be a ranked alphabet and let $t_1, \dots, t_n \in T_\Gamma(Y)$. Let $s \in T_\Gamma(Y_n \cup \mathcal{P}(Y_n))$. The* special first-order substitution *$[y_i \leftarrow t_i \mid i \in [n]]^\$$ (for short $[.]^\$$) applied to s is inductively defined as:*

$$s[.]^\$ = \begin{cases} t_i & \text{if } s = y_i \text{ for } i \in [n] \\ \gamma(s_1[.]^\$, \dots, s_k[.]^\$) & \text{if } s = \gamma(s_1, \dots, s_k) \\ \bigcup_{i \in U} \mathsf{ps}(t_i) & \text{if } s = \{y_i \mid i \in U\} \subseteq Y_n \text{ for some } U \subseteq [n]. \end{cases}$$

Let $\gamma_1^{(k_1)}, \dots, \gamma_n^{(k_n)} \in \Gamma$, $n \geq 1$ be pairwise different symbols and assume now that $t_i \in T_\Gamma(Y_{k_i} \cup \mathcal{P}(Y_{k_i}))$ for $i \in [n]$ and that $s \in T_\Gamma(Y_n)$. The special second-order substitution *$[\![\gamma_i \leftarrow t_i \mid i \in [n]]\!]$ (for short $[\![.]\!]^\$$) applied to s is inductively defined as:*

$$s[\![.]\!]^\$ = \begin{cases} t_i[y_j \leftarrow s_j[\![.]\!]^\$ \mid j \in [k_i]]^\$ & \text{if } s = \gamma_i(s_1, \dots, s_{k_i}) \text{ for } i \in [n] \\ \gamma(s_1[\![.]\!]^\$, \dots, s_k[\![.]\!]^\$) & \text{if } s = \gamma(s_1, \dots, s_k) \text{ with } \gamma \notin \{\gamma_1, \dots, \gamma_n\} \\ s & \text{if } s = y_j \text{ for } j \in [n]. \end{cases}$$

For all sets $Z \subseteq Y_m$ *such that no* Y*-node in* $t\llbracket.\rrbracket^{\$}$ *intersects* Z*, we define the* Z*-skeleton* $\lfloor t\llbracket.\rrbracket^{\$} \rfloor_Z$ *of* $t\llbracket.\rrbracket^{\$}$ *inductively as before, with a special case for* Y*-nodes: for all* Y*-nodes* S *we have* $\lfloor S \rfloor_Z = S \subseteq Y_m \setminus Z$*.*

▶ **Lemma 9.** *Let* M *be a nondeleting mttr as before. Let* $q \in Q$*,* $\sigma \in \Sigma^{(k)}$*, and* $p_1, \ldots, p_k \in P$*. Let* $p = h(\sigma(p_1, \ldots, p_k))$ *and* $t = rhs_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$*. Let* $s_1 \in L_{p_1}, \ldots, s_k \in L_{p_k}$*. By* $\llbracket.\rrbracket^{\$}$ *we denote the substitution* $\llbracket\langle q', x_i\rangle \leftarrow \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} \mid q' \in Q, i \in [k]\rrbracket^{\$}$ *and by* $\llbracket M\rrbracket$ *we denote* $\llbracket\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in Q, i \in [k]\rrbracket$*.*

(1) If $y \in Y(q,p)$ *and* y *occurs in* t *in the* j*-th argument of a node* $\langle q', x_i\rangle$ *for* $q' \in Q$ *and* $i \in [k]$*, then* $y_j \in Y(q', p_i)$*.*
(2) No Y*-node in* $t\llbracket.\rrbracket^{\$}$ *intersects* $Y(q,p)$*.*
(3) $\lfloor t\llbracket.\rrbracket^{\$}\rfloor_{Y(q,p)} = \lfloor t\llbracket M\rrbracket\rfloor_{Y(q,p)}$

Proof. If some $y_j \notin Y(q', p_i)$ then $\lfloor M_{q'}(L_{p_i})\rfloor_{y_j}$ is infinite and, if y occurs in t_j ($j \in [m]$), then $\lfloor M_q(L_p)\rfloor_y$ is also infinite and $y \notin Y(q,p)$. So (1) holds.

If $y \in Y(q,p)$ occurs in a Y-node of $t\llbracket.\rrbracket^{\$}$, then it occurs in t in the j-th argument of a node $\langle q', x_i\rangle$ with $y_j \notin Y(q', p_i)$, which contradicts (1). So (2) holds.

The statement (3) is proved by induction on t. The cases of $t = y_j$ and $t = \gamma(t_1, \ldots, t_n)$ are easy. In the case of $t = \langle q', x_i\rangle(t_1, \ldots, t_m)$, we have

$$\begin{aligned}
\left\lfloor t\llbracket.\rrbracket^{\$}\right\rfloor_{Y(q,p)} &= \left\lfloor \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} [y_j \leftarrow t_j\llbracket.\rrbracket^{\$} \mid j \in [m]]^{\$} \right\rfloor_{Y(q,p)} \\
&= \left\lfloor \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} [y_j \leftarrow t_j\llbracket M\rrbracket \mid j \in [m]]^{\$} \right\rfloor_{Y(q,p)} \\
&= \lfloor M_{q'}(s_i)[y_j \leftarrow t_j\llbracket M\rrbracket \mid j \in [m]]\rfloor_{Y(q,p)} \\
&= \lfloor t\llbracket M\rrbracket\rfloor_{Y(q,p)} \,.
\end{aligned}$$

◀

We can now prove Lemma 7:

Proof. All the statements are proven by induction on the structure of s. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. For $i \in [k]$ let $\hat{h}'(s_i) = (p_i, \varphi_i)$. By the definition of h', $p = h_\sigma(p_1, \ldots, p_k)$, which is equal to $\hat{h}(s)$. Thus, Statement (1) holds. For Statement (2) let $q \in F_p$: Then $\varphi(q)$ is defined as $\lfloor \zeta\llbracket\varphi_i\rrbracket^{\$}\rfloor_{Y(q,p)}$ where $\zeta = \mathrm{rhs}_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$ and $\llbracket\varphi_i\rrbracket^{\$}$ denotes the special substitution $\llbracket\langle q', x_i\rangle \leftarrow \varphi_i(q') \mid q' \in F_{p_i}, i \in [k]\rrbracket^{\$}$. By induction, $\lfloor \zeta\llbracket\varphi_i\rrbracket^{\$}\rfloor_{Y(q,p)}$ equals $\left\lfloor \zeta\llbracket\langle q', x_i\rangle \leftarrow \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} \mid q' \in F_{p_i}, i \in [k]\rrbracket^{\$}\right\rfloor_{Y(q,p)}$. By Lemma 9(3) the latter equals $\lfloor \zeta\llbracket\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in F_{p_i}, i \in [k]\rrbracket\rfloor_{Y(q,p)} = \lfloor M(s)\rfloor_{Y(q,p)}$.

We now prove Statement (3). Let $q \in Q$. Then $N_q(s) = \zeta\llbracket.\rrbracket\llbracket N\rrbracket$, where $\zeta = \mathrm{rhs}_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$, $\llbracket.\rrbracket$ is the substitution as in the construction, and $\llbracket N\rrbracket = \llbracket\langle r, x_i\rangle \leftarrow N_r(s_i) \mid r \in Q', i \in [k]\rrbracket$. By the induction hypothesis of Statement (2), we can replace $\varphi_i(q')$ by $\lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)}$ in the substitution $\llbracket.\rrbracket$. This gives

$$\begin{aligned}
&\zeta\llbracket\langle q', x_i\rangle \leftarrow \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} [u' \leftarrow [q', p_i, \varphi_i(q'), u'](y_{j_1}, \ldots, y_{j_n}) \mid \\
&\qquad\qquad \varphi_i(q')/u' = \{y_{j_1}, \ldots, y_{j_n}\}, j_1 < \cdots < j_n] \mid q' \in F_{p_i}, i \in [k]\rrbracket\llbracket N\rrbracket.
\end{aligned}$$

This can be written as $\zeta\llbracket.\rrbracket\llbracket H\rrbracket\llbracket Q\rrbracket$, where $\llbracket H\rrbracket = \llbracket\langle q', x_i\rangle \leftarrow N_{q'}(s_i) \mid q' \in H, i \in [k]\rrbracket$ and $\llbracket Q\rrbracket = \llbracket\langle q', x_i\rangle \leftarrow N_{q'}(s_i) \mid q' \in (Q \setminus F_{p_i}), i \in [k]\rrbracket$. By induction of Statement (4) the substitution $\llbracket H\rrbracket$ replaces the subtree $[q', p_i, \varphi_i(q'), u'](y_{j_1}, \ldots, y_{j_n})$ by the tree $M_{q'}(s_i)/u[y_{j_\nu} \leftarrow y_\nu \mid \nu \in [n]][y_\nu \leftarrow y_{j_\nu} \mid \nu \in [n]] = M_{q'}(s_i)/u$. Thus we obtain:

$$\begin{aligned}
&\zeta\llbracket\langle q', x_i\rangle \leftarrow \lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)} [u' \leftarrow M_{q'}(s_i)/u' \mid u' \in \mathcal{U}(\lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)})] \\
&\qquad\qquad\qquad\qquad \mid q' \in F_{p_i}, i \in [k]\rrbracket\llbracket Q\rrbracket
\end{aligned}$$

By Lemma 3 (for $Z = Y(q', p_i)$ and $t = M_{q'}(s_i)$) the tree on the right of the arrow in the leftmost second-order substitution equals $M_{q'}(s_i)$. We have:

$$\zeta[\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in F_{p_i}, i \in [k]]\!][\![\langle q', x_i\rangle \leftarrow N_{q'}(s_i) \mid q' \in Q \setminus F_{p_i}, i \in [k]]\!].$$

By induction of Statement (3), $N_{q'}(s_i) = M_{q'}(s_i)$ for $q \in Q \setminus F_{p_i}$. This gives us exactly $M_q(s)$, by the definition of the semantics of mttrs. Thus,

$$N_q(s) = \zeta[\![.]\!][\![N]\!] = M_q(s). \tag{1}$$

This concludes the proof of Statement (3).

We now prove Statement (4). Let $q \in F_p$ and $u \in V(t)$ with $t = \varphi(q)$ and $t/u \subseteq Y$. By the definition of the rules for the helper states, $N_{[q,p,t,u]}(s) = (\zeta[\![.]\!])/u[\![N]\!][y]$ where $t/u = \{y_{j_1}, \ldots, y_{j_n}\}$, $j_1 < \cdots < j_n$, and $[y] = [y_{j_\nu} \leftarrow y_\nu \mid \nu \in [n]]$. It follows from Lemma 9(1) that if $\langle q', x_i\rangle$ occurs in $\zeta = \mathrm{rhs}_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$ and $q \in F_p$, then $q' \notin Q \setminus F_{p_i}$. Hence, every proper ancestor v of u is labeled by a symbol in Δ, i.e., $(\zeta[\![.]\!][y])[v] \in \Delta$. This implies that we can move the "$/u$" operation of taking the subtree at node u to the right (after the application of the substitution $[\![N]\!]$) in the above displayed formula. We obtain $\zeta[\![.]\!][\![N]\!]/u[y]$. By the right equation in Formula 1, this equals $M_q(s)/u[y]$.

To prove Statement (5), let $q \in Q^{(m)}$, $m \geq 0$. Then

$$\zeta' = \mathrm{rhs}_N(q, \sigma, \langle (p_1, \varphi_1), \ldots, (p_k, \varphi_k)\rangle) = \zeta[\![.]\!],$$

where $\zeta = \mathrm{rhs}_M(q, \sigma, \langle p_1, \ldots, p_k\rangle)$ and $[\![.]\!]$ is as before. By Statement (2), $[\![.]\!]$ substitutes occurrences of $\langle q', x_i\rangle$ with $i \in [k]$ and $q' \in F_{p_i}$ by the tree $\lfloor M_{q'}(s_i)\rfloor_{Y(q',p_i)}$ in which leaves labeled by $Z \subseteq Y_m$ are replaced by $\langle q_H, x_i\rangle(y_{j_1}, \ldots, y_{j_n})$ with $Z = \{y_{j_1}, \ldots, y_{j_n}\}$. By Lemma 3(2) this implies that y_j occurs in ζ' for each $j \in [m]$. ◀

We show that the iteration of the construction $\pi(M)$ will terminate with a transducer that is depth proper. First, let us discuss what property a single iteration of π ensures. Let $p \in P$. Note that the set F_p is defined independently of reachability, i.e., F_p may contain states q such that (q, p) is *not* reachable. Let φ such that $(p, \varphi) \in P'$. Then $F_p \subseteq F_{(p,\varphi)}$. This inclusion follows from Lemma 7 as follows: let $s \in L_{(p,\varphi)}$ and let $q \in F_p$ be of rank m. The latter means that there exists a $j \in [m]$ and a number n such that every occurrence of y_j in $M_q(s')$ is at depth $\leq n$ for every $s' \in L_p$. By Lemma 7 (1), $s \in L_p$ and by Lemma 7 (3), $N_q(s) = M_q(s)$. Thus, every occurrence of y_j in $N_q(s)$ also occurs at depth $\leq n$. So $q \in F_{(p,\varphi)}$.

We now consider reachability. We say that a state q is *depth proper*, if for all $p \in P$ such that (q, p) is reachable, $q \notin F_p$. If $q \in F_p$, then for all φ such that $(p, \varphi) \in P'$ it holds that $(q, (p, \varphi))$ is not reachable. This property follows immediately from the definition of look-ahead and the rules of $\pi(M)$: the substitution $[\![.]\!]$ replaces each state call $\langle q', x_i\rangle$ with $q \in F_{p_i}$ by a tree that does not contain states of Q. So, if $(q, (p, \varphi))$ is reachable, then $q \notin F_p$; however, it may be that $q \in F_{(p,\varphi)}$, which means that q is not depth proper. It means that if $F_{(p,\varphi)} = F_p$ for all $(p, \varphi) \in P'$, then all states $q \in Q$ are depth proper. Let $Q_0 = Q$ and consider now the iterated application of π. Clearly, after some iterations of π, it will hold that $F_{(p,\varphi)} = F_p$ for all $(p, \varphi) \in P'$. To see this, consider the chain of inclusions

$$F_p \cap Q_0 \subseteq F_{(p,\varphi_1)} \cap Q_0 \subseteq \cdots \subseteq F_{(p,\varphi_1,\ldots,\varphi_k)} \cap Q_0 \subseteq \ldots$$

for any maps φ_i introduced in the look-ahead of $\pi^i(M)$. Since Q_0 is finite, the chain contains only finitely many strict inclusions. Hence there is a minimal n such that $F_{(p,\varphi_1,\ldots,\varphi_n)} \cap Q_0 = F_{(p,\varphi_1,\ldots,\varphi_{n'})} \cap Q_0$ for all $n' > n$.

Consider a tree with an artificial root node which contains all such chains, i.e., for each $p \in P$ there is exactly one child of the root node labeled F_p, and a node labeled F_p has children labeled $F_{(p,\varphi)}$ for each $(p, \varphi) \in P'$, etc. Moreover, a node labeled $F_{(p,\varphi_1,\dots,\varphi_n)}$ as in the chain above is a leaf of this tree. Since each node of this tree is finitely branching (because P' is finite) and each path has finite length, we know by König's lemma that the tree is finite. Thus, if d is the depth of this tree, then for the mttr $M' = \pi^d(M)$, all states in Q_0 are depth proper.

Let m be the maximal rank of the states in Q_0. Since all helper states are of rank $< m$, we know that M' contains no improper states of rank $\geq m$. We now proceed in the same fashion and construct a transducer $M'' = \pi^{n'}(M')$ which contains no improper states of rank $\geq (m - 1)$. In a similar way we eventually obtain an mttr for which *all states* are depth proper (and which is equivalent to M). Thus, even though we do not constructively derive a precise bound, we know that after *some* number of applications of π we are sure to obtain a depth proper mttr.

Before we state the main theorem of this section, we need the following lemma (the proof is a straightforward reduction to Proposition 2 and can be found in the full version of this paper: Lemma 8 in [18]).

▶ **Lemma 10.** *Let $M = (Q, P, \Sigma, \Delta, q_0, R, h)$ be an mttr and let $q \in Q^{(m)}$, $m \in \mathbb{N}$, $j \in [m]$, and $p \in P$. It is decidable whether or not $\lfloor M_q(L_p) \rfloor_{\{y_j\}}$ is finite. In case of finiteness, $\lfloor M_q(L_p) \rfloor_{\{y_j\}}$ can be constructed.*

Since for a pair (q, p) it is decidable whether or not it is reachable (see Section 2.2), Lemma 10 implies that it is decidable whether or not a given mttr is depth proper.

▶ **Theorem 11.** *For every mttr M, we can construct an equivalent mttr M' such that M' is depth proper.*

Proof. There is a nondeleting mttr M_0 equivalent to M ([9] or Proposition 1). We repeatedly construct equivalent transducers $\pi(M)$, $\pi(\pi(M))$, etc. until a proper mttr is obtained (which is decidable by Lemma 10). The repetition terminates (first eliminating all reachable calls of improper states of the highest rank m, then those or rank $m - 1$, etc.) as explained above. ◀

4 Linear Height and Linear Size-to-Height Increase

In this section we define the Linear Height and Linear Size-to-Height Increase properties. We then characterize and give decision algorithms for those properties by using the depth proper form.

Let Γ be a ranked alphabet and t be a tree over Γ. We define the size $|t|$ of a tree t as its number of nodes $|V(t)|$. The height $\mathsf{ht}(t)$ of t is defined as $\mathsf{ht}(t) = 0$ if $t \in \Gamma^{(0)}$ and $\mathsf{ht}(t) = 1 + \max\{\mathsf{ht}(t_i) \mid i \in [k]\}$ if $t = \gamma(t_1, \dots, t_k)$ for $\gamma \in \Gamma^{(k)}$, $k \geq 1$, and $t_1, \dots, t_k \in T_\Gamma$.

Let M be an mttr (with input ranked alphabet Σ). Then M has *linear size-to-height increase* (for short LSHI) if there exists a number c such that for every input tree $s \in T_\Sigma$: $\mathsf{ht}(M(s)) \leq c \cdot |s|$. The mttr M has *linear height increase* (for short LHI) if there exists a number c such that for every input tree $s \in T_\Sigma$: $\mathsf{ht}(M(s)) \leq c \cdot \mathsf{ht}(s)$.

We now introduce two additional properties for mttrs which will allow us to decide whether a given mttr has LSHI or LHI. Recall that $\widehat{M}$ denotes the extension of M: $\widehat{M}$ can translate input trees which may contain leaves that are labeled by elements from P (the set of look-ahead states of M). Whenever the state q of M, of rank m, encounters an input node u labeled by an element p of P, the transducer $\widehat{M}$ outputs $\langle q, p\rangle(y_1, \dots, y_m)$. We call a tree in $s \in T_\Sigma(P)$ a Σ-*context* if it contains exactly one occurrence u of an element of P.

We say that the mttr M is *finite nesting* (for short fnest), if there exists a number c such that for every Σ-context s there are at most c-many occurrences of symbols $\langle q,p\rangle$ with $q \in Q$ on any path of the tree $\widehat{M}(s)$; in this case, we say that c is a *nesting bound* of M. We say that M is *finite yield nesting* (for short fynest), if there exists a number c such that for every input tree $s \in T_\Sigma(P)$ there are at most c-many occurrences of symbols from $\langle q,p\rangle$ with $q \in Q$ on any path of the tree $\widehat{M}(s)$; in this case, we say that c is a *yield nesting bound* of M. The proof of the next lemma is straightforward (by reduction to Proposition 2).

▶ **Lemma 12.** *Let M be an mttr. Then (1) it is decidable whether or not M is finite nesting and (2) it is decidable whether or not M is finite yield nesting.*

Proof. Let $M = (Q, P, \Sigma, \Delta, q_0, R, h)$. We use the extension $\widehat{M} = (\hat{Q}, P, \hat{\Sigma}, \hat{\Delta}, q_0, \hat{R}, h)$ of M with input trees in $s \in T_\Sigma(P)$ which contain (1) exactly one or (2) arbitrarily many occurrences of elements of P. We then use a nondeterministic top-down tree transducer N which chooses any path in the tree $\widehat{M}(s)$ and outputs only the elements from $\langle Q,P\rangle$ on that path, now seen as unary symbols. The resulting output language $N(\widehat{M}(T_\Sigma))$ is finite if and only if M is (1) fnest or (2) fynest.

Formally, $N = (\{q_1^{(0)}\}, \hat{\Delta}, \Gamma, q_1, R')$ where $\Gamma = \langle Q,P\rangle \cup \{e^{(0)}\}$. For every $\delta \in \Delta^{(k)}$, $k \geq 1$, and $i \in k$ we let the rule $\langle q_1, \delta(x_1,\dots,x_k)\rangle \to \langle q_1, x_i\rangle$ be in R'. For every $\delta \in \Delta^{(0)}$ we let the rule $\langle q_1, \delta\rangle \to e$ be in R'. For every $\langle q,p\rangle \in \langle Q,P\rangle^{(m)}$, $m \geq 1$, and $i \in [m]$ we let the rule $\langle q_1, \langle q,p\rangle(x_1,\dots,x_m)\rangle \to \langle q,p\rangle(\langle q_1, x_i\rangle)$ be in R'. For every $\langle q,p\rangle \in \langle Q,P\rangle^{(0)}$ we let the rule $\langle q_1, \langle q,p\rangle\rangle \to \langle q,p\rangle$ be in R'. It is straightforward to show (by induction on the structure of s), that $N(\widehat{M}(T_\Sigma(P)))$ is finite if and only if M is fynest. Let L be the set of trees in $T_\Sigma(P)$ which contain exactly one occurrence of an element of P. It is straightforward to show (by induction on the structure of s), that $N(\widehat{M}(L))$ is finite if and only if M is fnest. ◀

Informally the next lemma is easy to understand, e.g., for Statement (1), if M is finite nesting with bound c, then a single node of an input tree can only "contribute" at most $c \cdot \mathrm{mhr}$ to the height of the output tree, where mhr denotes the maximum height of the right-hand side of any rule of the mttr.

▶ **Lemma 13.** *Let M be an mttr. (1) If M is finite nesting, then it is of linear size-to-height increase. (2) If M is finite yield nesting, then it is of linear height increase.*

Proof. Informally, we can understand this lemma by looking at a given path O in an output tree and, using origin semantics, at how many nodes along this path have their origin in different parts of the input tree.

For (1), the finite nesting property gives a bound c on the number of state calls to a single input node, nested along path O. Intuitively, noting mhr the maximum height of the right-hand side of a rule, $c \cdot \mathrm{mhr}$ is a bound on the number of output nodes along path O with their origin in a single input node. This bound clearly implies that the height of the output (maximum number of nodes on a path) is linearly bounded by the size of the input.

For (2), instead of looking at a single input node, we look at all the input nodes at a given depth d in the input. The finite yield nesting property implies a bound c on the nesting (along a path O) of state calls to input nodes of depth d. Each such call may produce at most mhr nodes along path O with their origin in a node of depth d. So c.mhr is a bound for the number of nodes along path O with their origin in a node of depth d.

Formally, we apply $\widehat{M}$ to a tree $t \in T_\Sigma(P)$. We modify t by substituting nodes in P, and we bound the growth of the height of $\widehat{M}(t)$ for each substitution. We will conclude by stating that any input tree $s \in T_\Sigma$ can be built by successive substitutions, and so the height of the output is linearly bounded by the number of substitutions (which will be the size of s for (1), and the height of s for (2)). Let $M = (Q, P, \Sigma, \Delta, q_0, R, h)$ and let mhr be the maximum height of the right-hand side of any rule in R. Let s be a fixed tree in T_Σ.

To prove (1), consider an arbitrary set U of pairwise independent (i.e. not being descendants of each other) nodes of a fixed input tree $s \in T_\Sigma$. Let $s' = s[u \leftarrow h(s/u) \mid u \in U]$, let $u \in U$, and $\sigma = s[u] \in \Sigma^{(k)}$ with $k \geq 0$. Let c be a nesting bound for M, then, along any output path in $\widehat{M}(s')$, there are at most c state calls $\langle q, s'/u\rangle$ with origin u in s'. Then $\widehat{M}(s'[u \leftarrow \sigma(h(s/u1), \dots, h(s/uk))])$ is obtained by replacing such state calls with the corresponding right-hand side of rules, which implies that: $\mathsf{ht}(\widehat{M}(s'[u \leftarrow \sigma(h(s/u1), \dots, h(s/uk))])) \leq c \cdot \mathrm{mhr} + \mathsf{ht}(\widehat{M}(s'))$. The tree $s \in T_\Sigma$ can be obtained from the tree $h(s) \in T_\Sigma(P)$ by $|s|$ such substitutions. The height of $\widehat{M}(h(s)) = \langle q_0, h(s)\rangle$ is 0. So $\mathsf{ht}(M(s)) \leq c \cdot \mathrm{mhr} \cdot |s|$, so M of linear size-to-height increase.

To prove (2), for $i \in [\mathsf{ht}(s)]$, let U_i be the set of nodes at depth i in s, and let s_i be the tree obtained from s by replacing all nodes $u \in U_i$ by $h(s/u)$. Let c be a yield nesting bound for M, then, along any output path O in $\widehat{M}(s_i)$, there are at most c state calls $\langle q, s_i/u\rangle$ with $u \in U_i$. Then $\widehat{M}(s_{i+1}) = \widehat{M}(s_i[u \leftarrow \sigma_u(h(s/u1), \dots, h(s/uk)) \mid u \in U_i])$ is obtained by replacing these state calls in $\widehat{M}(s_i)$ with the corresponding right-hand side of rules, so $\mathsf{ht}(\widehat{M}(s_{i+1})) \leq c \cdot \mathrm{mhr} + \mathsf{ht}(\widehat{M}(s_i))$. By applying this $\mathsf{ht}(s) + 1$ times, starting from s_0, we obtain that the height of $M(s)$ is $\leq c \cdot \mathrm{mhr} \cdot (\mathsf{ht}(s) + 1)$. So M is of linear-height-increase. ◀

The next lemma is a central piece of the paper. This is where we use the *depth proper property.*

▶ **Lemma 14.** *Let M be an mttr that is depth proper. (1) If M is not finite nesting, then M does not have linear size-to-height increase. (2) If M is not finite yield nesting, then M does not have linear height increase.*

Proof. Let M be given by a tuple as usual. To prove (1), assume that M is not fnest. We will use this and the *depth proper property* to show that M does not have LSHI. Since M is not fnest (and has only finitely many states) there must be some state $q \in Q^{(m)}$ with $m \geq 1$ that occurs arbitrarily often on paths of output trees of $\widehat{M}$. More precisely, there are infinite sequences of contexts $c_0, c_1, \dots$ and numbers $n_0 < n_1 < \cdots$ such that q occurs $\geq n_0$ times on a path in $\widehat{M}(c_0)$ and q occurs $\geq n_1$ times on a path in $\widehat{M}(c_0[u_0 \leftarrow c_1])$ where u_0 is the path in c_0 to a node $p \in P$, etc. From this we can deduce (by considering sufficiently many numbers n_i), similarly to the proof of Lemma 6.5 of [10], that M is "(nested) input pumpable", i.e., there exist $q_1, q_2, j, s_0, s_1, u_0, u_1, p$ such that

1. $\langle q_1, p\rangle$ occurs in $\widehat{M}(s_0[u_0 \leftarrow p])$,
2. $\widehat{M}_{q_1}(s_1[u_1 \leftarrow p])$ has either: a subtree $\langle q_1, p\rangle(t_1, \dots, t_m)$ such that some $t_{j'}$ contains a subtree $\langle q_2, p\rangle(\xi_1, \dots, \xi_l)$ where ξ_j contains $y_{j'}$ for some $j' \in [m]$, or a subtree $\langle q_2, p\rangle(t_1, \dots, t_l)$ such that t_j contains a subtree $\langle q_1, p\rangle(\xi_1, \dots, \xi_m)$,
3. $\widehat{M}_{q_2}(s_1[u_1 \leftarrow p])$ has a subtree $\langle q_2, p\rangle(t_1, \dots, t_l)$ such that t_j contains y_j, and
4. $p = h(s_1/u_1) = h(s_1[u_1 \leftarrow p])$.

By "pumping", i.e., considering $s_n = s_0[u_0 \leftarrow s_1[u_1 \leftarrow s_1[u_1 \leftarrow \dots]]]$ with n replacements of the node u_1, we obtain that $\widehat{M}_{q_1}(s_n)$ contains a path with at least n nested occurrences of $\langle q_2, p\rangle$. Note that this proof is simpler than that of Lemma 6.5 of [10] because we only look here at the height of outputs instead of the size of outputs. This is simpler because, in a mttr, a state call can copy a parameter containing large outputs of other state calls, causing a size growth of the output that is difficult to track, but these copies cannot be copied vertically on top of each other, so the output height is easier to track.

This is where we use the *depth proper property*: we first assume by contradiction that M has LSHI, i.e., there exists a c such that for every input tree $s \in T_\Sigma$: $\mathsf{ht}(M(s)) \leq c \cdot |s|$. Since M is *depth proper*, we may choose $s \in L_p$ such that $M_{q_2}(s)$ contains an occurrence of y_j at depth $\geq c \cdot c_1 + 1$, where $c_1 = |s_1[u_1 \leftarrow p]| - 1$. We know that $\widehat{M}_{q_1}(s_n)$ contains at least n nested occurrences of q_2 (where the j-th subtree of q_2 always contains further nested occurrences of q_2). Now let $t_n = s_0[u_0 \leftarrow s_n[u_1^n \leftarrow s]]$ and take $n > c(c_0 + c_2)$, where $c_0 = |s_0[u_0 \leftarrow p]| - 1$ and $c_2 = |s|$. Since $|t_n| = c_0 + nc_1 + c_2$, we obtain that $\mathsf{ht}(M(t_n)) > c \cdot |t_n|$ because $\mathsf{ht}(M(t_n)) \geq n(cc_1 + 1) > ncc_1 + c(c_0 + c_2) = c \cdot |t_n|$ by the choice of n. So *nested input pumpability* implies that M is not of LSHI.

We now prove that if M is not finite nesting, then it must be *nested input pumpable*. In order to do so, we first introduce a few notations and characterize *nested input pumpability* and the *finite nesting* property using these notations.

Let c be a Σ-context and $q \in Q$ be a state of M. To talk about the nesting of states in $\widehat{M}_q(c)$, we first give a notation for paths:

1. For any node u at depth n in $\widehat{M}_q(c)$, we note the path to node u as the sequence of pairs:

   $$(\ell_1, i_1)\,(\ell_2, i_2)\ldots(\ell_n, i_n)\,(\ell_{n+1}, \perp)$$

   where $i_1, \ldots, i_n$ are indexes such that $u = i_1\, i_2 \ldots i_n$ and, for all $j \leq n+1$, ℓ_j is the label of node $i_1 \ldots i_{j-1}$ or, if node $i_1 \ldots i_{j-1}$ is labeled by a state call $\langle q', p\rangle$, then $\ell_j = q'$,
2. Since we are only interested in the nesting of states, we remove from such paths all pairs (ℓ_j, i_j) where $\ell \in \Delta$. We obtain nesting sequences of the form:

   $$(q_1, k_1)\,(q_2, k_2)\ldots(q_n, k_n)\,(\ell_{n+1}, k_{n+1})$$

   where ℓ_{n+1} is either a state in Q or a parameter, $k_{n+1} \in \{\perp\} \cup \mathbb{N}$, and for all $j \leq n$, $k_j \in [m_j]$ where m_j is the arity of state q_j.
3. For each such sequence, if $\ell_{n+1} = q_{n+1} \in Q$ then we write:

   $$(q, \perp) \to_c (q_1, k_1)\,(q_2, k_2)\ldots(q_n, k_n)\,(\ell_{n+1}, k_{n+1})$$

   Otherwise $\ell_{n+1} = y_k$ is a parameter of q, $k_{n+1} = \perp$ and we write:

   $$(q, k) \to_c (q_1, k_1)\,(q_2, k_2)\ldots(q_n, k_n)$$

This defines a relation $\to_c \subseteq \Theta \times \Theta^*$ where $\Theta = \{(q, k) \mid q \in Q^{(m)}, k \in [m] \cup \{\perp\}\}$ and Θ^* denotes the set of (possibly empty) sequences of elements of Θ. Note that if $(q, \perp) \to_c w$, then in the nesting sequence $w \in \Theta^*$ only the last pair may contain $\perp$. A *nesting loop* is given by a Σ-context c with a leaf labeled p such that $h(c) = p$, and two pairs $(q_1, k_1), (q_2, k_2) \in \Theta$ such that:

- $(q_1, k_1) \to_c w_1\,(q_1, k_1)\,w_2\,(q_2, k_2)\,w_3$ or $(q_1, k_1) \to_c w_1\,(q_2, k_2)\,w_2\,(q_1, k_1)\,w_3$,
- $(q_2, k_2) \to_c w_4\,(q_2, k_2)\,w_5$,
- (q_1, p) is reachable, i.e., there exists Σ-context c_0 with a leaf labeled p such that $\langle q_1, p\rangle$ appears in $\widehat{M}(c_0)$,

for some nesting sequences $w_1, w_2, w_3, w_4, w_5 \in \Theta^*$. This allows us to rephrase the *nested input pumpability* property as the existence of a *nesting loop*. We want to prove that if M is not finite nesting then it has a nesting loop.

We extend the relation $\to_c$ to sequences of pairs on the left so that, for pairs (q_1, k_1), $(q_2, k_2) \in \Theta$ and sequences $w_1, w_2 \in \Theta^*$, if $(q_1, k_1) \to_c w_1$ and $(q_2, k_2) \to_c w_2$ then $(q_1, k_1)\,(q_2, k_2) \to_c w_1 w_2$. More generally, for all sequences $w_1, w_1', w_2, w_2' \in \Theta^*$, if $w_1 \to_c w_1'$ and $w_2 \to_c w_2'$ then $w_1\, w_2 \to_c w_1'\, w_2'$. We can now show the following claim:

▷ Claim 15. For all Σ-contexts c and c' with leafs labeled resp. p and p' such that $p = h(c')$, we can define the Σ-context $c \cdot c' = c[p \leftarrow c']$ and, for all sequences $w, w'' \in \Theta^*$, if $w \to_{c \cdot c'} w''$ then there exists a sequence $w' \in \Theta^*$ such that $w \to_c w' \to_{c'} w''$.

Proof. We only need to show this for $w = (q_0, k_0) \in \Theta$ because of the definition of $\to_c$ on sequences of pairs. Because $(q_0, k_0) \to_{c \cdot c'} w''$, there must be a path Π in $\widehat{M}_{q_0}(c \cdot c')$ reducing to w'' (by removing pairs in $\Delta \times \mathbb{N}$ and removing $(y_{k_0}, \bot)$ if $k_0 \neq \bot$). Because $\widehat{M}_{q_0}(c \cdot c') = \widehat{M}_{q_0}(c)[\langle q, p\rangle \leftarrow \widehat{M}_q(c')]$, path Π can be similarly obtained from a path Π' in $\widehat{M}_{q_0}(c)$ by substituting each (q, k) with a path in $\widehat{M}_q(c')$. More specifically, noting $(q_1, k_1), \dots, (q_n, k_n)$ the pairs in path Π' that are in Θ (in order of apparition in Π'), we substitute in Π':

- each occurrence of a pair $(q_i, k_i) \in \Theta$ by a path Π'_i such that $\Pi'(y_{k_0,\bot})$ is a path in $\widehat{M}_{q_i}(c')$ (for $i \leq n$),
- each occurrence of a pair $(q_n, \bot)$ by a path Π'_n in $\widehat{M}_{q_n}(c')$.

We get: $\Pi = \Pi'[(q_i, k_i) \leftarrow \Pi'_i]$ and, by removing pairs in $(\Delta \times \mathbb{N}) \cup (Y^m \times \{\bot\})$:

$$w'' = w'_1\, w'_2\, \dots\, w'_n$$

where for all $i \leq n$, w'_i is obtained from Π'_i by removing pairs in $(\Delta \times \mathbb{N}) \cup (Y^m \times \{\bot\})$. Then, for all $i \leq n$ and by definition of Π'_i, we have $(q_i, k_i) \to_{c'} w'_i$. So $(q_1, k_1) \dots (q_n, k_n) \to_{c'} w'_1\, w'_2\, \dots\, w'_n = w''$.

We note w' the sequence obtained from Π' by removing pairs in $(\Delta \times \mathbb{N}) \cup (Y^m \times \{\bot\})$. Then $w' = (q_1, k_1) \dots (q_n, k_n)$ and so $(q_0, k_0) \to_c w' \to_{c'} w''$. ◁

We could also prove that $\to_{c \cdot c'} = \to_{c'} \circ \to_c$, but it is not necessary for this proof.

To prove that M has a nesting loop (i.e. M is nested input pumpable), we assume that M is not finite nesting. Then, for all $n \in \mathbb{N}$, there exists a Σ-context c_n such that: $(q_0, \bot) \to_{c_n} w$ for some $w \in \Theta^*$ with $|w| \geq n$. We can decompose any such c_n into a concatenation $c_{n,1} \cdot c_{n,2} \cdot \dots c_{n,r}$ and use the claim to obtain:

$$(q_0, \bot) \to_{c_{n,1}} w_1 \to_{c_{n,2}} w_2 \cdots \to_{c_{n,r}} w_r$$

where $w_1, w_2, \dots w_r \in \Theta^*$ and $|w_r| = |w| \geq n$. By choosing a large enough n, we will show how to find a *nesting loop*. To do that, we decompose c_n into several contexts and use the claim.

A Σ-context c is *atomic* if its leaf labeled in P is a child node of its root. Let $c = \sigma(t_1, \dots, t_{i-1}, p_i, t_{i+1}, \dots, t_k)$ be an atomic Σ-context and $q \in Q$ a state of M. Noting $p_j = h(t_j)$ for $j \neq i$, there is in M a rule $\langle q, \sigma(x_1 : p_1, \dots, x_k : p_k)\rangle(y_1, \dots, y_m) \to t$. Then $\widehat{M}_q(c) = t[\langle q', x_j\rangle \leftarrow M_{q'}(c/j) \mid j \neq i]$. Because $M_{q'}(c/j) \in T_\Delta$ for all $q' \in Q$ and $j \neq i$, the nesting of state calls in $\widehat{M}_q(c)$ is the nesting of state calls of the form $\langle q', x_i\rangle$ in t. So, for $(q, k) \in \Theta$, the length of nesting sequences w such that $(q, k) \to_c w$ is bounded by the height of t. There is a finite number of rules for M, so there is a finite number of such t and the length of sequences $w \in \Theta$ such that $(q, k) \to_c w$ has an upper bound B that does not depend on q, k or c. In other words, for all $(q, k) \in \Theta$, $w \in \Theta^*$ and atomic Σ-context c we have:

$$(q, k) \to_c w \quad\Rightarrow\quad |w| \leq B$$

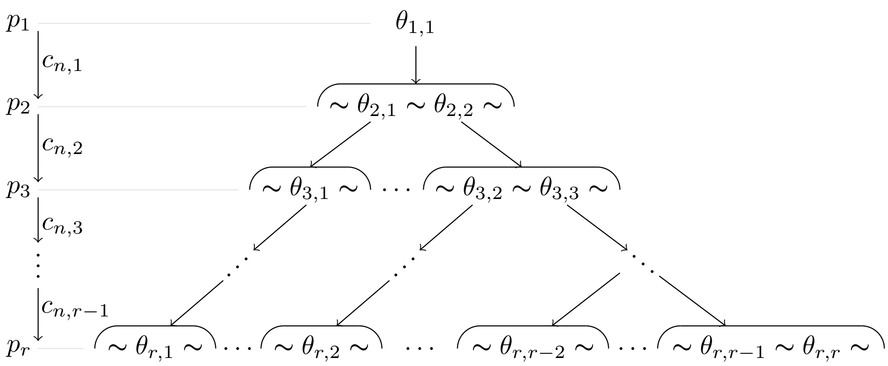
Moreover, for all $w_1, w_2 \in \Theta^*$: $w_1 \to_c w_2 \quad\Rightarrow\quad |w_2| \leq B\,|w_1|$.

We decompose the Σ-context c_n into atomic Σ-contexts $c_{n,1}, c_{n,2}, \dots, c_{n,r}$. Since $(q_0, \bot) \to_{c_{n,1}} w_1 \to_{c_{n,2}} w_2 \cdots \to_{c_{n,r}} w_r$, we have $|w_r| \leq B^r$ and, since $|w_r| \geq n$: $n \leq B^r$. So, by choosing n large enough, we can also make r as big as we want. In order to find a nesting loop, we require more structure on the nesting sequences $c_{n,1}, \dots, c_{n,r}$. The precise structure we need is described in the following claim:

▷ Claim 16. For all for all $r \in \mathbb{N}$, if there exists a Σ-context c, a pair $\theta \in \Theta$ and a sequence $w \in \Theta^*$ with $\theta \to_c w$ and $|w| \geq B^r$, then there exists Σ-contexts $c_1, \ldots, c_{r-1}$, look-ahead states $p_1, \ldots, p_r$ and pairs $\theta_{i,j} \in \Theta$ for all $i, j \in [r]$ with $j \leq i$ such that:

- for all $i \in [r-1]$, $h(c_i) = p_i$ and c_i has a leaf labeled p_{i+1},
- $\theta_{1,1} = \theta$,
- for all $i, j \in [r-1]$ with $j < i$, there exists $w_{i,j}, w'_{i,j} \in \Theta^*$ such that: $\theta_{i,j} \to_{c_i} w_{i,j}\, \theta_{i+1,j}\, w'_{i,j}$,
- for all $i \in [r-1]$, there exists $w_i, w'_i, w''_i \in \Theta^*$ such that either $\theta_{i,i} \to_{c_i} w_i\, \theta_{i+1,i}\, w'_i\, \theta_{i+1,i+1}\, w''_i$ or $\theta_{i,i} \to_{c_i} w_i\, \theta_{i+1,i+1}\, w'_i\, \theta_{i+1,i}\, w''_i$.

These conditions can be summed up graphically. To simplify the picture, we replace all sequences $w_{i,j}, w'_{i,j}, w_i, w'_i, w''_i$ for $i, j \in [r]$ with the symbol $\sim$.



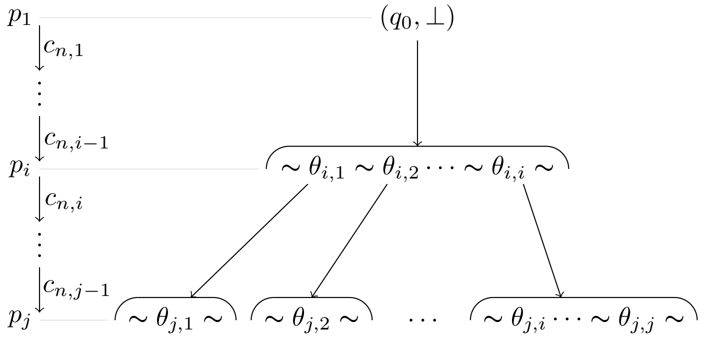
Note that, in this representation, we chose to represent $\theta_{i,i} \to_{c_i} w_i\, \theta_{i+1,i}\, w'_i\, \theta_{i+1,i+1}\, w''_i$ instead of $\theta_{i,i} \to_{c_i} w_i\, \theta_{i+1,i+1}\, w'_i\, \theta_{i+1,i}\, w''_i$ for all $i \in [r-1]$. But this distinction does not matter to the proof of the claim. From now on we use $\sim$ to denote arbitrary sequences in Θ^* which we will not use to find a nesting loop.

Proof. We prove this by induction on r. Let c be a Σ-context, θ a pair in Θ and w a sequence in Θ^* with $\theta \to_c w$ and $|w| \geq B^{r+1}$. We split c into atomic Σ-contexts $c'_1, \ldots, c'_n$, then we have sequences $w_1, \ldots, w_{n-1} \in \Theta^*$ such that $\theta \to_{c'_1} w_1 \cdots \to_{c'_{n-1}} w_{n-1} \to_{c'_n} w$. Let i be the largest i such that there is a pair θ' in the sequence w_i with $\theta' \to_{c'_{i+1} \cdots c'_n} w'$ and $|w'| \geq B^r$. If we had $|w'| \geq B^{r+1}$ then, because c'_{i+1} is atomic, we would have a θ'' in the sequence c'_{i+1} with $\theta'' \to_{c'_{i+2} \cdots c'_n} w''$ and $|w''| \geq B^r$. So $B^r \leq |w'| < B^{r+1} \leq |w|$. Therefore there is another pair $\theta_{2,1}$ in w_i (other than θ') with $\theta_{2,1} \to_{c'_{i+1} \cdots c'_n} w''$ and $|w''| \geq 1$.

Since $\theta' \to_{c'_{i+1} \cdots c'_n} w'$ and $|w'| \geq B^r$, we use the induction hypothesis on θ' and $c'_{i+1} \cdots c'_n$. In order to prove the induction for $r+1$, we rename the Σ-contexts $c_1, \ldots c_{r-1}$, look-ahead states $p_1, \ldots, p_r$ and pairs $\theta_{i,j}$ (for $j \leq i \leq r$) into Σ-contexts $c_2, \ldots c_r$, look-ahead states $p_2, \ldots, p_{r+1}$ and pairs $\theta_{i+1,j+1}$ (for $j \leq i \leq r$). Then $c'_{i+1} \cdots c'_n = c_2 \cdots c_r$.

Since $\theta_{2,1} \to_{c_2 \cdots c_r} w''$ with $|w''| \geq 1$, there are pairs $\theta_{3,1}, \ldots, \theta_{n,1}$ such that $\theta_{n,1}$ appears in the sequence w'' and, for all $i \in [r]$ with $i \geq 2$, $\theta_{i,1} \to_{c_i} w'_i\, \theta_{i+1,1}\, w''_i$ with $w'_i, w''_i \in \Theta^*$. To conclude, we choose $c_1 = c'_1 \ldots c'_i$, $p_1 = h(c_1)$ and $\theta_{1,1} = \theta$. ◁

In order to find a nesting loop, we need two indexes i and j with:



Formally, we require two indexes i, j with $i < j < r$ which share
- the same look-ahead state $h(c_{n,i}) = h(c_{n,j})$,
- the same pair $\theta_{i,i} = \theta_{j,j} \in \Theta$ and
- the same set of pairs $\{\theta_{i,\ell}\}_{0\leq\ell\leq i} = \{\theta_{j,\ell}\}_{0\leq\ell\leq j}$.

We ensure the existence of such i, j by choosing $r \geq |P|\,|Q|\,(m+1)\,2^{|Q|(m+1)} + 1$ where m is the maximum arity of states. We now show how to build the nesting loop from indexes i, j. We note $p = h(c_{n,i}) = h(c_{n,j})$, $(q_1, k_1) = \theta_{i,i} = \theta_{j,j}$ and $S = \{\theta_{i,\ell}\}_{0\leq\ell\leq i} = \{\theta_{j,\ell}\}_{0\leq\ell\leq j}$. We note $c' = c_{n,i}.c_{n,i+1}.\dots.c_{n,j-1}$. Note that c' has a leaf labeled p and $h(c') = p$.

We need the sets $\{\theta_{i,\ell}\}_{0\leq\ell\leq i}$ and $\{\theta_{j,\ell}\}_{0\leq\ell\leq j}$ to be equal so that the pairs $\theta_{i,k}$ for $k \leq i$ loop on each other through the loop c'. Formally, noting $\theta'_0 = \theta_{j,i}$, for all $\theta'_k \in S$ for $k \in \mathbb{N}$, there exists $\theta'_{k+1} \in S$ such that $\theta'_k \to'_c \sim \theta'_{k+1} \sim$. Since $S \subseteq \Theta$ is finite, there must be $n, m \in \mathbb{N}$ such that $\theta'_n = \theta'_{n+m}$ (with $m \geq 1$), so $\theta'_n \to_{c'^m} \sim \theta'_n \sim$. Also $(q_1, k_1) \to_{c'} \sim \theta'_0 \sim (q_1, k_1) \sim$ and $\theta'_0 \to_{c'^n} x_n$, so $(q_1, k_1) \to_{c'^{n+1}} \sim \theta'_n \sim (q_1, k_1) \sim$ and, for all $m' \in \mathbb{N}$: $(q_1, k_1) \to_{c'^{m'}} \sim (q_1, k_1) \sim \to_{c'^{n+1}} \sim \theta'_n \sim (q_1, k_1) \sim$. Finally, for $c = c'^{m(n+1)}$, we have $(q_1, k_1) \to_c \sim \theta'_n \sim (q_1, k_1) \sim$ and $\theta'_n \to_c \theta'_n$. So we have a *nesting loop*.

In conclusion, if M is not finite nesting, then it is *nested input pumpable*, and so it does not have linear size-to-height increase.

The proof of Statement (2) can be given in a very similar way as for (1), here we only outline the changes to the notations which allow to adapt the proof of (1) to (2). We replace Σ-contexts with elements of the set $T_\Sigma(P)$ containing possibly several leafs labeled in P. The rest of the notational changes are consequences of this change. Given a $s \in T_\Sigma(P)$, we now consider the nesting of state calls called on distinct subtrees of s with potentially distinct look-ahead states. We augment pairs in Θ so as to include the look-ahead, so $\Theta = \{(q, k, p) \mid q \in Q^{(m)}, k \in [m] \cup \{\bot\}, p \in P\}$. The notation $(q, k, p) \to_s (q_1, k_1, p_1) \dots (q_n, k_n, p_n)$ means that $h(s) = p$ and calls to states $q_1, \dots, q_n$ on leafs of s labeled $p_1, \dots, p_n$ resp. are nested on parameters $y_{k_1}, \dots, y_{k_n}$ along a path in $\widehat{M}_q(s)$. This means that, when concatenating contexts, we write $s(s_1, \dots, s_m)$ instead of $s \cdot s'$.

For (2), similarly to nesting loops for (1), we define a *yield nesting loop* as given by contexts $s_1, s_2 \in T_\Sigma(P)$, look-ahead states $p_1, p_2 \in P$ and triplets $(q_1, k_1, p_1), (q_2, k_2, p_2) \in \Theta$ such that:
- $h(s_1) = p_1$, $h(s_2) = p_2$, s_1 has two leafs labeled p_1 and p_2, s_2 has one leaf labeled p_2,
- $\langle q_1, p_1\rangle$ is reachable,
- either $(q_1, k_1, p_1) \to_{s_1} \sim (q_2, k_2, p_2) \sim (q_1, k_1, p_1) \sim$
  or $(q_1, k_1, p_1) \to_{s_1} \sim (q_1, k_1, p_1) \sim (q_2, k_2, p_2) \sim$,
- $(q_2, k_2, p_2) \to_{s_2} \sim (q_2, k_2, p_2) \sim$.

We say that M is *yield nested input pumpable* when it has either a *yield nesting loop* or a *nesting loop*. Note that the existence of either of these loops contradicts the linear height increase property. To prove (2) we prove that infinite yield nesting implies the existence of either a yield nesting loop or a nesting loop. That proof works similarly to (1): M is not fynest so we can find large enough nesting sequences (but with the new definition of $\to_s$), find a repeating triplet (q_1, k_1, p_1), pump the loop enough times that a triplet (q_2, k_2, p_2) loops onto itself. Note that if the nested calls to (q_1, k_1, p_1) and (q_2, k_2, p_2) in $(q_1, k_1, p_1) \to_{s_1} \sim (q_2, k_2, p_2) \sim (q_1, k_1, p_1) \sim$ are on the same leaf in s_1 (with $p_1 = p_2$), then we get a nesting loop (otherwise we get a yield nesting loop). ◀

From Theorem 11 and Lemmas 12, 13, and 14 we obtain our following main theorem.

▶ **Theorem 17.** *Let M be an mttr. Then (1) it is decidable whether or not M has linear size-to-height increase (2) it is decidable whether or not M is linear height-increase.*

5 Conclusions

We have proven that for a given macro tree transducer (with look-ahead) it is decidable whether or not it has linear height increase (LHI) and, whether or not it has linear size-to-height increase (LSHI). Both decision procedures rely on a novel normal form that is called "depth-proper" normal form. Roughly speaking the normal form requires that each parameter of every state of the transducer appears at arbitrary large depths in output trees generated by that state (and for a given look-ahead state).

One major open problem in the field is to prove a Conjecture of Joost Engelfriet (from around the year 2000), that the translation of an mttr can be defined by an attributed tree transducer (atts) if and only if the translation has "linear size to number of distinct output subtrees" increase. Note that deciding such property is out of reach (it is at least as difficult as deciding equivalence of atts). To prove this characterization, different loops need to be considered which produce unbounded numbers of states in (partial) output trees. We believe that the depth normal form will be instrumental in reducing the number of different such loops that must be considered and therefore will be of great help in proving the conjecture.

References

1 Alfred V. Aho and Jeffrey D. Ullman. Translations on a context-free grammar. *Inf. Control.*, 19(5):439–475, 1971. doi:10.1016/S0019-9958(71)90706-6.

2 Rajeev Alur and Loris D'Antoni. Streaming tree transducers. *J. ACM*, 64(5):31:1–31:55, 2017. doi:10.1145/3092842.

3 Mikolaj Bojanczyk and Amina Doumane. First-order tree-to-tree functions. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 252–265. ACM, 2020. doi:10.1145/3373718.3394785.

4 Hubert Comon-Lundh, Max Dauchet, Rémi Gilleron, Cristof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. URL: https://jacquema.gitlabpages.inria.fr/files/tata.pdf, November 2007.

5 Frank Drewes and Joost Engelfriet. Decidability of the finiteness of ranges of tree transductions. *Inf. Comput.*, 145(1):1–50, 1998. doi:10.1006/inco.1998.2715.

6 Joost Engelfriet. Bottom-up and top-down tree transformations – A comparison. *Math. Syst. Theory*, 9(3):198–231, 1975. doi:10.1007/BF01704020.

7 Joost Engelfriet. Context-free grammars with storage. *CoRR*, abs/1408.0683, 2014. arXiv:1408.0683.

8 Joost Engelfriet, Kazuhiro Inaba, and Sebastian Maneth. Linear-bounded composition of tree-walking tree transducers: linear size increase and complexity. *Acta Informatica*, 58(1-2):95–152, 2021. doi:10.1007/s00236-019-00360-8.

9 Joost Engelfriet and Sebastian Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.*, 154(1):34–91, 1999. doi:10.1006/inco.1999.2807.

10 Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003. doi:10.1137/S0097539701394511.

11 Joost Engelfriet and Erik Meineche Schmidt. IO and OI. I. *J. Comput. Syst. Sci.*, 15(3):328–353, 1977. doi:10.1016/S0022-0000(77)80034-2.

12 Joost Engelfriet and Erik Meineche Schmidt. IO and OI. II. *J. Comput. Syst. Sci.*, 16(1):67–99, 1978. doi:10.1016/0022-0000(78)90051-X.

13 Joost Engelfriet and Heiko Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31(1):71–146, 1985. doi:10.1016/0022-0000(85)90066-2.

14 Emmanuel Filiot, Sebastian Maneth, Pierre-Alain Reynier, and Jean-Marc Talbot. Decision problems of tree transducers with origin. *Inf. Comput.*, 261:311–335, 2018. doi:10.1016/j.ic.2018.02.011.

15 M. J. Fischer. *Grammars with Macro like Productions.* PhD thesis, Harvard University, 1968. See also *Proc. 9th Sympos. on SWAT*, pp. 131-142, 1968.

16 Zoltán Fülöp. On attributed tree transducers. *Acta Cybern.*, 5(3):261–279, 1981. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3218.

17 Zoltán Fülöp and Heiko Vogler. *Syntax-Directed Semantics – Formal Models Based on Tree Transducers.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1998. doi:10.1007/978-3-642-72248-6.

18 Paul Gallot, Sebastian Maneth, Keisuke Nakano, and Charles Peyrat. Deciding linear height and linear size-to-height increase for macro tree transducers, 2024. arXiv:2307.16500.

19 Ferenc Gécseg and Magnus Steinby. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 1–68. Springer, 1997. doi:10.1007/978-3-642-59126-6_1.

20 Donald E. Knuth. Semantics of context-free languages. *Math. Syst. Theory*, 2(2):127–145, 1968. doi:10.1007/BF01692511.

21 Donald E. Knuth. Correction: Semantics of context-free languages. *Math. Syst. Theory*, 5(1):95–96, 1971. doi:10.1007/BF01702865.

22 Thomas Perst and Helmut Seidl. Macro forest transducers. *Inf. Process. Lett.*, 89(3):141–149, 2004. doi:10.1016/j.ipl.2003.05.001.

23 William C. Rounds. Mappings and grammars on trees. *Math. Syst. Theory*, 4(3):257–287, 1970. doi:10.1007/BF01695769.

24 James W. Thatcher. Generalized sequential machine maps. *J. Comput. Syst. Sci.*, 4(4):339–367, 1970. doi:10.1016/S0022-0000(70)80017-4.

25 James W. Thatcher. There's a lot more to finite automata theory than you would have thought. In *Proc. 4th Ann. Princeton Conf. on Informations Sciences and Systems*, pages 263–276, 1970. Also published in revised form under the title "Tree automata: an informal survey" in Currents in the Theory of Computing (ed. A. V. Aho), Prentice-Hall, 1973, 143–172.