Output String Languages of Compositions of Deterministic Macro Tree Transducers

Joost Engelfriet and Sebastian Maneth

Leiden University, LIACS, PO Box 9512, 2300 RA Leiden, The Netherlands
E-mail: {engelfri, maneth}@liacs.nl

Technical Report 2001-04

Abstract. The composition of total deterministic macro tree transducers gives rise to a proper hierarchy with respect to their output string languages (these are the languages obtained by taking the yields of the output trees). There is a language not in this hierarchy which can be generated by a (quite restricted) nondeterministic string transducer, namely, a two-way generalized sequential machine. Similar results hold for attributed tree transducers, for controlled EDT0L systems, and for YIELD mappings (which proves properness of the IO-hierarchy). Witnesses for the properness of the macro tree transducer hierarchy can already be found in the latter three hierarchies.

1 Introduction

Macro tree transducers [Eng80,CF82,EV85] are a model of syntax-directed semantics (see [FV98] for a survey) which combine top-down tree transducers and macro grammars, i.e., they are finite state transducers, the states of which are equipped with parameters that allow to handle context information.

A macro tree transducer M can be used as a string language generator as follows. The tree translation of M is applied to a tree language, which typically is the set of derivation trees of a context-free grammar, or, in general, a regular tree language. This generates an output tree language of M, and taking the yields of these trees generates an output string language of M. In this way one can also view M as a controlled (tree) grammar, where the generation of the output trees is controlled by the input trees. Then, the iteration of control corresponds to the composition of the tree translations. The string languages generated by the composition closure of macro tree transducers form a very large class with nice properties: it is a full AFL, and membership, emptiness, and finiteness of its languages are decidable [DE98]. Because of their special relevance to syntax-directed semantics we here investigate total deterministic macro tree transducers (for short, MTTs) only; they are a combination of total deterministic top-down tree transducers and IO (inside-out) macro grammars.

The question arises, whether composition of MTTs gives rise to a proper hierarchy of output string languages. For the two ingredients of MTTs the situation is as follows. Since (total deterministic) top-down tree transducers are closed under composition [Rou70], they do not form a proper hierarchy of output string languages (note that composition of nondeterministic top-down tree transducers does yield such a hierarchy [Eng82]). The iteration of IO macro grammars by the concept of n-level grammars gives rise to a proper and to an infinite hierarchy [Dam82], for the generated tree and string languages, respectively: the so-called IO-hierarchies (see, e.g., [ES78]). With respect to the translations it is well known that composition of MTTs (which corresponds to the n-level tree transducers of [EV88]) yields a proper hierarchy, that is, the class of translations realized by the composition of n MTTs is properly included in the one realized by the composition of $n+1$ MTTs (cf. [EV85]). The proof relies on the fact that the height of the output tree of an MTT is exponentially bounded by the height of the input tree. In [Dam82] it is proved that also the output tree languages form a proper hierarchy. With respect to the output string languages, composition of MTTs yields an infinite hierarchy; the proof in [Dam82] combines the above exponential bound with the concept of rational index [BCN81]. To prove properness of this hierarchy (at each level) we use instead a so-called "bridge-theorem" (cf. [Eng82], and the section on translational techniques in [Gre81]).

Let us discuss the bridge theorem in more detail. Consider two languages L' and L such that L' is of some special form, depending on L; in applications of the bridge theorem, L' will typically be obtained from L by some kind of string insertion. Now if L' is the output string language of an MTT, then the special form of L' forces the language L to be an output string language of an MTT M which has certain restricted properties. To be precise, these properties require that in the rules of M (i) no parameter is copied and (ii) no parameter is deleted. An MTT satisfying (i) and (ii) is called *simple in the parameters* (for short sp). The proof of this bridge theorem is a generalization of the proof of Theorem 3.4.3 in [Fis68], where Fischer proves for a specific IO macro language L' that L can be produced by an IO macro grammar that is sp. For an MTT M that is sp we show that, with respect to the output string language, parameters are not needed at all; that is, we can construct a top-down tree transducer which has the same output string language as M. Since MTTs are closed under composition with top-down tree transducers, this result will allow us to use the bridge theorem to step down from the composition of $n+1$ MTTs to that of n MTTs.

We apply the bridge theorem to three different types of string insertions to obtain the following results:

(1) There is a language L' which is not the output string language of any composition of MTTs, but which can be generated by a nondeterministic two-way generalized sequential machine. Here, L' is obtained from L by the nondeterministic insertion of two new symbols, where L is a language that cannot be generated by a top-down tree transducer. Intuitively, the result shows that nondeterminism (present in a very simple type of insertion) is more powerful than determinism (present in an MTT).
As another example of this phenomenon we prove that there is a context-free language which cannot be generated as output by the composition closure of MTTs, taking monadic tree languages as initial input. The latter class of languages is of interest because it contains the EDT0L-hierarchy, generated by the iteration of controlled EDT0L systems. The EDT0L system is the deterministic version of the ET0L system [Roz73] (see [ERS80] for the relationship of these systems to top-down tree transducers and two-way machines). In particular we show that languages generated by the iteration of $n+1$ controlled EDT0L systems can be generated by the composition of n MTTs.

(2) Composition of MTTs yields a proper hierarchy with respect to their output string languages, i.e., there is a language L' which is output string language of the composition of $n+1$ MTTs, but which cannot be generated as output by the composition of n MTTs. Here L' is obtained from a language L at the previous level, by inserting a sequence of b's before each symbol of a string in L (for a new symbol b), viz., b^i before the i-th symbol from the right.
In fact, we use the relationship with EDT0L systems mentioned in point (1) and show that L' can be generated by the iteration of $n+2$ controlled EDT0L systems. This implies properness of the EDT0L-hierarchy. In [Eng82] properness of the ET0L-hierarchy is proved, but it is mentioned as open whether the EDT0L-hierarchy is proper.

(3) There is an $(n+1)$-level IO macro language L' which cannot be generated as output by the composition of n MTTs. Here, L' is obtained from L (at the previous level) by inserting, before each symbol of a string w in L, a string in $\{1,2\}^*$ that represents (in Dewey notation) the corresponding leaf of some binary tree with yield w. Since every n-level IO macro language can be generated as output by the composition of n MTTs, this proves the properness of the IO-hierarchy of string languages, which was left open in [Dam82].
Since every n-level IO macro language can also be generated as output by the composition of n attributed tree transducers [Fül81,FV98] (ATTs), and ATTs can be simulated by MTTs, we also obtain that composition of ATTs yields a proper hierarchy of output string languages.

This paper is structured as follows. Section 2 contains basic notions concerning trees, tree substitution, tree translations, and finite state relabelings. In Section 3, the definition of macro tree transducers is given and some basic lemmas are recalled. Furthermore, the sp (simple in the parameters) property is defined. In Section 4 it is proved that MTTs are closed under composition with finite state relabelings. This also implies the closure of MTTs under composition with top-down tree transducers with regular look-ahead, which is mentioned as an open problem in the Conclusions of [EV85]. Finally, it is proved that MTTs that are simple in the parameters generate the same class of output string languages as top-down tree transducers. Section 5 contains the detailed proof of the bridge theorem, together with two particular versions of it. Using these theorems it is proved in Section 6 that composition of MTTs yields a proper hierarchy of output string languages (the yMTT-hierarchy) and that the EDT0L-hierarchy is proper. Moreover, it is shown that there are "non-deterministic" languages not in the yMTT- and the EDT0L-hierarchies, which can be generated by a nondeterministic two-way generalized sequential machine and a context-free grammar, respectively. The properness of the IO-hierarchy is proved in Section 7, and it is shown that the EDT0L-hierarchy is included in the IO-hierarchy. Finally, Section 8 contains the hierarchy result for ATTs and a summary of relations between the various hierarchies discussed in this paper; it also mentions some open problems.

Some of the results of this paper were presented in [Man99].

2 Preliminaries

The set $\{0,1,\dots\}$ of natural numbers is denoted by $\mathbb{N}$. The empty set is denoted by $\varnothing$. For $k \in \mathbb{N}$, $[k]$ denotes the set $\{1,\dots,k\}$; thus $[0]=\varnothing$. For a set A, $|A|$ is its cardinality, and A^* is the set of all strings over A. An alphabet is a finite set A. The empty string is denoted by ε. The length of a string w is denoted by $|w|$, and the i-th symbol in w is denoted by $w(i)$. For a string $w = a_1 \cdots a_n$, its reverse $a_n \cdots a_1$ is denoted by w^r. For strings $v, w_1, \dots, w_n \in A^*$ and distinct $a_1,\dots,a_n \in A$, we denote by $v[a_1 \leftarrow w_1, \dots, a_n \leftarrow w_n]$ the result of (simultaneously) substituting w_i for every occurrence of a_i in v. Note that $[a_1 \leftarrow w_1, \dots, a_n \leftarrow w_n]$ is a homomorphism on strings. For a condition P on a and w we use, similar to set notation, $[a \leftarrow w \mid P]$ to denote the substitution $[L]$, where L is the list of all $a \leftarrow w$ for which condition P holds. By *REG* and *CF* we denote the classes of regular and context-free languages, respectively.

For functions $f\colon A \to B$ and $g\colon B \to C$ their composition is $(f \circ g)(x) = g(f(x))$; note that the order of f and g is nonstandard. For sets of functions F and G their composition is $F \circ G = \{f \circ g \mid f \in F, g \in G\}$, and $F^n = F \circ \cdots \circ F$ (n times). For a binary relation $\Rightarrow$, its transitive reflexive closure is denoted by $\Rightarrow^*$.

Let A and B be disjoint alphabets. For $w \in (A \cup B)^*$ we denote by $\mathrm{res}_A(w)$ the restriction of w to letters in A, i.e., res_A is the homomorphism from $(A \cup B)^*$ to A^* defined by $\mathrm{res}_A(a) = a$ for $a \in A$ and $\mathrm{res}_A(a) = \varepsilon$ for $a \in B$.

2.1 Ranked Sets and Trees

A set Σ together with a mapping $\mathrm{rank}_\Sigma : \Sigma \to \mathbb{N}$ is called a *ranked set*. For $k \geq 0$, $\Sigma^{(k)}$ is the set $\{\sigma \in \Sigma \mid \mathrm{rank}_\Sigma(\sigma) = k\}$; we also write $\sigma^{(k)}$ to indicate that $\mathrm{rank}_\Sigma(\sigma) = k$. If $\Sigma = \Sigma^{(1)} \cup \Sigma^{(0)}$, then Σ is *monadic*. For a set A, $\langle \Sigma, A \rangle$ is the ranked set $\Sigma \times A$ with $\mathrm{rank}_{\langle \Sigma, A\rangle}(\langle \sigma, a \rangle) = \mathrm{rank}_\Sigma(\sigma)$ for every $\langle \sigma, a\rangle \in \langle \Sigma, A \rangle$.

For the rest of this paper we choose the *set of input variables* to be $X = \{x_1, x_2, \dots\}$ and the *set of parameters* to be $Y = \{y_1, y_2, \dots\}$. For $k \geq 0$, $X_k = \{x_1, \dots, x_k\}$ and $Y_k = \{y_1, \dots, y_k\}$.

Let Σ be a ranked set. The *set of trees over Σ*, denoted by T_Σ, is the smallest set of strings $T \subseteq (\Sigma \cup \{(,),,\})^*$ such that $\Sigma^{(0)} \subseteq T$ and if $\sigma \in \Sigma^{(k)}$, $k \geq 1$, and $t_1, \dots, t_k \in T$, then $\sigma(t_1, \dots, t_k) \in T$. For $\alpha \in \Sigma^{(0)}$ we denote the tree α also by $\alpha()$. If Σ is monadic, then $t \in T_\Sigma$ is a *monadic tree*. For a set A, the *set of trees over Σ indexed by A*, denoted by $T_\Sigma(A)$, is the set $T_{\Sigma \cup A}$, where for every $a \in A$, $\mathrm{rank}_A(a) = 0$.

For every tree $t \in T_\Sigma$, the *set of nodes of t*, denoted by $V(t)$, is a subset of $\mathbb{N}^*$ which is inductively defined as follows: if $t = \sigma(t_1, \dots, t_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and for all $i \in [k], t_i \in T_\Sigma$, then $V(t) = \{\varepsilon\} \cup \{iu \mid u \in V(t_i), i \in [k]\}$. Thus, ε represents the root of a tree and for a node u the i-th child of u is represented by ui. The *label of t at node u* is denoted by $t[u]$; we also say that $t[u]$ occurs in t (at u). The node u is a *leaf* if it has no children, i.e., if $t[u] \in \Sigma^{(0)}$. The *subtree of t at node u* is denoted by t/u. The *substitution of the tree $s \in T_\Sigma$ at node u in t* is denoted by $t[u \leftarrow s]$; it means that the subtree t/u is replaced by s. Formally, these notions can be defined as follows: $t[\varepsilon]$ is the first symbol of t (in Σ), $t/\varepsilon = t$, $t[\varepsilon \leftarrow s] = s$, and if $t = \sigma(t_1, \dots, t_k)$, $i \in [k]$, and $u \in V(t_i)$, then $t[iu] = t_i[u]$, $t/iu = t_i/u$, and $t[iu \leftarrow s] = \sigma(t_1, \dots, t_i[u \leftarrow s], \dots, t_k)$. The pre-order of the nodes of t is the lexicographical order on $\mathbb{N}^*$; thus, $\varepsilon < iu$, if $u < v$ then $iu < iv$, and if $i < j$ then $iu < jv$.

For a tree $t \in T_\Sigma$, yt denotes the *yield of t*, i.e., the string in $(\Sigma^{(0)})^*$ obtained by reading the leaves of t in pre-order, omitting nodes labeled by the special symbol e. Thus, $yt = t[\rho_1] \cdots t[\rho_m]$, where $\rho_1, \dots, \rho_m$ are all leaves ρ of t with $t[\rho] \neq e$, in pre-order (e.g., for $t = \sigma(a, \sigma(e, b))$, $yt = t[1]t[22] = ab$). The string yt can be obtained recursively as follows; if $t = e$ then $yt = \varepsilon$, if $t \in \Sigma^{(0)} - \{e\}$ then $yt = t$, and if $t = \sigma(t_1, \dots, t_k)$, $k \geq 1$, $\sigma \in \Sigma^{(k)}$, and $t_1, \dots, t_k \in T_\Sigma$, then $yt = yt_1 \cdots yt_k$.

Let A be an alphabet and let $w \in A^*$. For a binary symbol $\sigma \notin A$, the tree $\mathrm{comb}_\sigma(w) \in T_{\{\sigma\}}(A)$ is recursively defined as follows; if $w = \varepsilon$ then $\mathrm{comb}_\sigma(w) = e$, and if $w = aw'$ with $a \in A$ and $w' \in A^*$, then $\mathrm{comb}_\sigma(w) = \sigma(a, \mathrm{comb}_\sigma(w'))$. Clearly, $y\mathrm{comb}_\sigma(w) = w$. The monadic tree $\mathrm{sm}(w) \in T_\Gamma$ with $\Gamma = \{a^{(1)} \mid a \in A\} \cup \{e^{(0)}\}$ is recursively defined as e if $w = \varepsilon$ and as $a(\mathrm{sm}(w'))$ if $w = aw'$ with $a \in A$ and $w' \in A^*$. As an example, $\mathrm{comb}_\sigma(acc) = \sigma(a, \sigma(c, \sigma(c, e)))$ and $\mathrm{sm}(acc) = a(c(c(e)))$.

2.2 Second-Order Tree Substitution

Note that trees are particular strings and that string substitution as defined in the beginning of this section is applicable to a tree to replace symbols of rank zero; we refer to this type of substitution as "first-order tree substitution".

Let Σ be a ranked alphabet, let $\sigma_1, \dots, \sigma_n$ be distinct elements of Σ, $n \geq 1$, and for each $i \in [n]$ let s_i be a tree in $T_{\Sigma - Y}(Y_k)$, where $k = \mathrm{rank}_\Sigma(\sigma_i)$. For $t \in T_\Sigma$, the *second-order tree substitution of σ_i by s_i in t*, denoted by

$$t[\![\sigma_1 \leftarrow s_1, \dots, \sigma_n \leftarrow s_n]\!]$$

is inductively defined as follows (abbreviating $[\![\sigma_1 \leftarrow s_1, \dots, \sigma_n \leftarrow s_n]\!]$ by $[\![\dots]\!]$). For $t = \sigma(t_1, \dots, t_m)$ with $\sigma \in \Sigma^{(m)}$, $m \geq 0$, and $t_1, \dots, t_m \in T_\Sigma$, (i) if $\sigma = \sigma_i$ for an $i \in [n]$, then $t[\![\dots]\!] = s_i[y_j \leftarrow t_j[\![\dots]\!] \mid j \in [k]]$ and (ii) otherwise $t[\![\dots]\!] = \sigma(t_1[\![\dots]\!], \dots, t_m[\![\dots]\!])$. Note that $[\![\sigma_1 \leftarrow s_1, \dots, \sigma_n \leftarrow s_n]\!]$ is a tree homomorphism [GS84] and that (just as ordinary substitution) second-order tree substitution is associative (by the closure of tree homomorphisms under composition, cf. Theorem IV.3.7 of [GS84]), i.e., $t[\![\sigma \leftarrow s]\!][\![\sigma \leftarrow s']\!] = t[\![\sigma \leftarrow s[\![\sigma \leftarrow s']\!]]\!]$ and if $\sigma' \neq \sigma$ then $t[\![\sigma \leftarrow s]\!][\![\sigma' \leftarrow s']\!] = t[\![\sigma' \leftarrow s', \sigma \leftarrow s[\![\sigma' \leftarrow s']\!]]\!]$, and similarly for the general case (cf. Sections 3.4 and 3.7 of [Cou83]). For a condition P on σ and s we use $[\![\sigma \leftarrow s \mid P]\!]$ to denote the substitution $[\![L]\!]$, where L is the list of all $\sigma \leftarrow s$ for which condition P holds.

The following small lemma says that if we are considering the yield of a tree to which a (first- or second-order) tree substitution is applied, then inside the substitution merely the yields of the trees that are substituted are relevant.

Lemma 1. Let Σ be a ranked alphabet, $\alpha_1, \dots, \alpha_n \in \Sigma^{(0)} - \{e\}$, and $\sigma_1, \dots, \sigma_n \in \Sigma$. Let $t, t', s_1, s_1', \dots, s_n, s_n' \in T_\Sigma(Y)$ such that $ys_i = ys_i'$ for every $i \in [n]$.

(a) If $yt = yt'$, then $y(t[\alpha_1 \leftarrow s_1, \dots, \alpha_n \leftarrow s_n]) = y(t'[\alpha_1 \leftarrow s_1', \dots, \alpha_n \leftarrow s_n'])$.
(b) $y(t[\![\sigma_1 \leftarrow s_1, \dots, \sigma_n \leftarrow s_n]\!]) = y(t[\![\sigma_1 \leftarrow s_1', \dots, \sigma_n \leftarrow s_n']\!])$.

Proof. (a) Clearly, $y(t[\alpha_1 \leftarrow s_1, \dots, \alpha_n \leftarrow s_n])$ equals $(yt)[\alpha_i \leftarrow ys_i \mid i \in [n]]$ (note that here the substitution is on strings). Since $yt = yt'$ and $ys_i = ys_i'$, this equals $(yt')[\alpha_i \leftarrow ys_i' \mid i \in [n]] = y(t'[\alpha_1 \leftarrow s_i', \dots, \alpha_n \leftarrow s_n'])$.

(b) This part is proved by induction on the structure of t. Let $t = \sigma(t_1, \dots, t_k)$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \dots, t_k \in T_\Sigma$. Let $[\![\sigma_1 \leftarrow s_1, \dots, \sigma_n \leftarrow s_n]\!]$ be denoted by $[\![\dots]\!]$ and let $[\![\sigma_1 \leftarrow s_1', \dots, \sigma_n \leftarrow s_n']\!]$ be denoted by $[\![\_]\!]$.

(i) If $\sigma = \sigma_i$ for an $i \in [n]$, then $y(t[\![\dots]\!]) = y(s_i[y_j \leftarrow t_j[\![\dots]\!] \mid j \in [k]])$. By induction, $y(t_j[\![\dots]\!]) = y(t_j[\![\_]\!])$ for $j \in [k]$. Hence, by (a) (for $t = s_i$, $t' = s_i'$, $\alpha_j = y_j$, $s_j = t_j[\![\dots]\!]$, and $s_j' = t_j[\![\_]\!]$), this equals $y(s_i'[y_j \leftarrow t_j[\![\_]\!] \mid j \in [k]]) = y(t[\![\_]\!])$.

(ii) Otherwise, $y(t[\![\dots]\!]) = y\sigma(t_1[\![\dots]\!], \dots, t_k[\![\dots]\!]) = y(t_1[\![\dots]\!]) \cdots y(t_k[\![\dots]\!])$. By the induction hypothesis we get $y(t_1[\![\_]\!]) \cdots y(t_k[\![\_]\!]) = y(t[\![\_]\!])$. □

2.3 Tree Translations and Relabelings

Let Σ and Δ be ranked alphabets. A subset L of T_Σ is called a *tree language.* A (total) function $\tau : T_\Sigma \to T_\Delta$ is called a *tree translation* or simply *translation.* For a tree language $L \subseteq T_\Sigma$, $\tau(L)$ denotes the set $\{t \in T_\Delta \mid t = \tau(s) \text{ for some } s \in L\}$ and $yL = \{yt \mid t \in L\}$. For a class $\mathcal{T}$ of tree translations and a class $\mathcal{L}$ of tree languages, $\mathcal{T}(\mathcal{L})$ denotes the class of tree languages $\{\tau(L) \mid \tau \in \mathcal{T}, L \in \mathcal{L}\}$ and $y\mathcal{L} = \{yL \mid L \in \mathcal{L}\}$.

A tree language is *regular* (or recognizable) if there is a finite state tree automaton recognizing it, or, equivalently, there is a regular tree grammar generating it. The class of regular tree languages is denoted by $REGT$. Note that $\mathrm{sm}(REG) \subseteq REGT$. The reader is assumed to be familiar with the basic properties of the regular tree languages (see, e.g., [GS84,GS97]).

A (total deterministic) *finite state relabeling* M is a tuple (Q, Σ, Δ, R), where Q is a finite set of *states*, Σ and Δ are ranked alphabets of *input* and *output symbols*, respectively, and R is a finite set of rules such that for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $q_1, \dots, q_k \in Q$, R contains exactly one rule of the form $\sigma(\langle q_1, x_1\rangle, \dots, \langle q_k, x_k\rangle) \to \langle q, \delta(x_1, \dots, x_k)\rangle$, where $q \in Q$ and $\delta \in \Delta^{(k)}$. The rules of M are used as term rewriting rules, and the derivation relation induced by M is denoted by $\Rightarrow_M$; more formally, for $\xi, \xi' \in T_{\langle Q, T_\Delta\rangle \cup \Sigma}$, $\xi \Rightarrow_M \xi'$ if and only if

- there is a subtree $\sigma(\langle q_1, t_1\rangle, \dots, \langle q_k, t_k\rangle)$ (rooted at node u) of ξ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, $q_1, \dots, q_k \in Q$, and $t_1, \dots, t_k \in T_\Delta$, and

– $\xi' = \xi[u \leftarrow \langle q, \delta(t_1, \ldots, t_k)\rangle]$, where $\sigma(\langle q_1, x_1\rangle, \ldots, \langle q_k, x_k\rangle) \to \langle q, \delta(x_1, \ldots, x_k)\rangle$ is a rule in R.

If we are only interested in the state q in which M arrives for input s, then we write $s \Rightarrow_M^* \langle q, \_\rangle$ (to mean that $s \Rightarrow_M^* \langle q, t\rangle$ for some tree t). Note that for each $q \in Q$, $\{s \in T_\Sigma \mid s \Rightarrow_M^* \langle q, \_\rangle\}$ is a regular tree language. The translation τ_M realized by M is $\{(s,t) \in T_\Sigma \times T_\Delta \mid s \Rightarrow_M^* \langle q, t\rangle, q \in Q\}$. The class of all translations that can be realized by finite state relabelings is denoted by $D_t QRELAB$.

3 Macro Tree Transducers

In this section macro tree transducers are defined and some results which will often be used throughout the paper are recalled. Furthermore, the nondeleting and sp (simple in the parameters) properties are defined.

Definition 2. A (total deterministic) *macro tree transducer* (for short, MTT) is a tuple $M = (Q, \Sigma, \Delta, q_0, R)$, where Q is a ranked alphabet of *states*, Σ and Δ are ranked alphabets of *input* and *output symbols*, respectively, $\Delta \cap Y = \varnothing$, $q_0 \in Q^{(0)}$ is the *initial state*, and R is a finite set of *rules* of the following form. For every $q \in Q^{(m)}$ and $\sigma \in \Sigma^{(k)}$ with $m, k \geq 0$ there is exactly one rule of the form

$$\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \to \zeta \qquad (*)$$

in R, where $\zeta \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$. □

A rule r of the form $(*)$ is called the (q,σ)-rule of M and its right-hand side ζ is denoted by $\mathrm{rhs}_M(q,\sigma)$; it is also called a q-rule. The rules of M can be viewed as term rewriting rules in the obvious way, with the input variables x_i ranging over T_Σ and the parameters y_j ranging over T_Δ. Then M induces a derivation relation $\Rightarrow_M$ on $T_{\langle Q, T_\Sigma\rangle \cup \Delta}$ and an input tree $s \in T_\Sigma$ is translated by M into the unique tree $t \in T_\Delta$ with $\langle q_0, s\rangle \Rightarrow_M^* t$. Instead of using the derivation relation $\Rightarrow_M$ to define the translation realized by M, we use the following recursive definition of q-translations, which is based on second-order tree substitution as defined in Section 2.2.

Definition 3. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT and let $q \in Q^{(m)}$ be a state of M. The *q-translation of M* is the total function $M_q : T_\Sigma \to T_\Delta(Y_m)$ defined as follows. For every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$,

$$M_q(\sigma(s_1, \ldots, s_k)) = \mathrm{rhs}_M(q,\sigma)[\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!].$$

The *translation realized by* M, denoted by τ_M, is the q_0-translation M_{q_0} of M. □

Note that the q-translation of M can also be obtained using the derivation relation $\Rightarrow_M$ discussed above, i.e., for every input tree s of M, $\langle q, s\rangle(y_1, \ldots, y_m) \Rightarrow_M^* M_q(s)$ (cf. Lemma 4.8 of [EV94]). In proofs we will always use the q-translations of M, but our intuition is often based on the derivation relation $\Rightarrow_M$. For an example of an MTT M, and the way it works, see Example 10 at the end of this section.

The class of all translations which can be realized by MTTs is denoted by MTT. A *top-down tree transducer* is an MTT all states of which are of rank zero. The class of all translations which can be realized by top-down tree transducers is denoted by T. If a top-down tree transducer has only one state, then it is a *tree homomorphism*. Note that every tree homomorphism is a second-order tree substitution, and vice versa.

The following two results are often used in this paper.

Lemma 4. (Corollary 4.10 of [EV85]) $T \circ MTT \subseteq MTT$.

Lemma 5. (Theorem 4.12 of [EV85]) $MTT \circ T \subseteq MTT$.

Since regular look-ahead can be simulated by finite state relabelings (see Corollary IV.6.7 in [GS84]), the fact that MTT is closed under regular look-ahead (Theorem 4.21 of [EV85]) can be stated as follows.

Lemma 6. $D_t QRELAB \circ MTT \subseteq MTT$.

Recall from Section 2.1 that for a string $w = a_1 \cdots a_n$, $\mathrm{sm}(w)$ is the monadic tree $a_1(a_2(\cdots a_n(e)\cdots))$. The next lemma shows that an MTT can turn the yield ys of its input tree s into the monadic tree $\mathrm{sm}(ys)$.

Lemma 7. Let Σ be a ranked alphabet. There is an MTT M_Σ with input alphabet Σ such that for every $s \in T_\Sigma$, $\tau_{M_\Sigma}(s) = \mathrm{sm}(ys)$.

Proof. Define $M_\Sigma = (\{q_0^{(0)}, q^{(1)}\}, \Sigma, \Gamma, q_0, R)$ with $\Gamma = \{a^{(1)} \mid a \in \Sigma^{(0)}, a \neq e\} \cup \{e^{(0)}\}$. For every $\sigma \in \Sigma^{(k)}, k \geq 1$, let the rules

$$\begin{aligned} \langle q_0, \sigma(x_1, \ldots, x_k)\rangle \quad &\to \langle q, x_1\rangle(\langle q, x_2\rangle(\ldots(\langle q, x_k\rangle(e))\ldots)) \\ \langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1) &\to \langle q, x_1\rangle(\langle q, x_2\rangle(\ldots(\langle q, x_k\rangle(y_1))\ldots)) \end{aligned}$$

be in R, for every $a \in \Sigma^{(0)} - \{e\}$ let $\langle q_0, a\rangle \to a(e)$ and $\langle q, a\rangle(y_1) \to a(y_1)$ be in R, and let $\langle q_0, e\rangle \to e$ and $\langle q, e\rangle(y_1) \to y_1$ be in R.

We now show that $\tau_{M_\Sigma} = y \circ \mathrm{sm}$. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. Then $\tau_{M_\Sigma}(s) = M_{q_0}(s)$ which, by Definition 3, equals $\mathrm{rhs}_M(q_0, \sigma)[\![\ldots]\!]$ with $[\![\ldots]\!] = [\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in \{q_0, q\}, i \in [k]]\!]$. By the definition of the rules of M this equals $\mathrm{rhs}_M(q, \sigma)[y_1 \leftarrow e][\![\ldots]\!] = \mathrm{rhs}_M(q, \sigma)[\![\ldots]\!][y_1 \leftarrow e] = M_q(s)[y_1 \leftarrow e]$, which, by the following claim, is equal to $\mathrm{sm}(ys)$.

<u>*Claim:*</u> For every $s \in T_\Sigma$, $M_q(s) = \mathrm{sm}(ys)[e \leftarrow y_1]$.

The proof is by induction on the structure of s. If $s = e$ then $M_q(s) = y_1 = \mathrm{sm}(\varepsilon)[e \leftarrow y_1] = \mathrm{sm}(ys)[e \leftarrow y_1]$, and if $s = a \in \Sigma^{(0)} - \{e\}$ then $M_q(s) = a(y_1) = \mathrm{sm}(ys)[e \leftarrow y_1]$. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 1$, and $s_1, \ldots, s_k \in T_\Sigma$. It follows from Definition 3 that $M_q(s) = \langle q, x_1\rangle(\cdots\langle q, x_k\rangle(y_1)\cdots)\ [\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in \{q_0, q\}, i \in [k]]\!]$. Applying the induction hypothesis (and combining the substitution of y_1) we get

$$\mathrm{sm}(ys_1)[e \leftarrow \mathrm{sm}(ys_2)[\cdots[e \leftarrow \mathrm{sm}(ys_k)[e \leftarrow y_1]]\cdots]].$$

Clearly, $[e \leftarrow y_1]$ can be moved out of the substitutions. By the fact that $\mathrm{sm}(w)[e \leftarrow \mathrm{sm}(w')] = \mathrm{sm}(ww')$, we get $\mathrm{sm}(ys_1 \cdots ys_k)[e \leftarrow y_1] = \mathrm{sm}(ys)[e \leftarrow y_1]$. $\square$

A macro tree transducer M is *nondeleting*, if in the right-hand side of every q-rule, for every state q of rank $m \geq 1$, each parameter y_j, $j \in [m]$, occurs at least once. This property makes sure that the output generated in a parameter position cannot be deleted. First, let us prove a small lemma which says that also in the translations $M_q(s)$, every parameter of q occurs. This is similar to Lemma 6.7 of [EM99], which says that if every parameter y_j occurs exactly once in a right-hand side (for all rules of M), then y_j also occurs exactly once in $M_q(s)$.

Lemma 8. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a nondeleting MTT, $q \in Q^{(m)}$, $m \geq 1$, and $s \in T_\Sigma$. Then for every $j \in [m]$, y_j occurs in $M_q(s)$.

Proof. Let $j \in [m]$. The proof is by induction on the structure of s. The induction hypothesis is denoted by IH1. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. By Definition 3, $M_q(s) = \mathrm{rhs}_M(q, \sigma)[\![\ldots]\!]$ with $[\![\ldots]\!] = [\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!]$. Since M is nondeleting, y_j occurs in $t = \mathrm{rhs}_M(q, \sigma)$ and, by the following claim, y_j occurs in $t[\![\ldots]\!]$.

Claim: Let $t \in T_{\langle Q,X_k\rangle\cup\Delta}(Y_m)$. If y_j occurs in t, then it occurs in $t[\![\dots]\!]$.

The claim is proved by induction on the structure of t. The induction hypothesis is denoted by IH2. If $t = y_j$, then $t[\![\dots]\!] = y_j$. Let $l \geq 1$ and $t_1,\dots,t_l \in T_{\langle Q,X_k\rangle\cup\Delta}(Y_m)$. If $t = \delta(t_1,\dots,t_l)$ with $\delta \in \Delta^{(l)}$, then $t[\![\dots]\!] = \delta(t_1[\![\dots]\!],\dots,t_l[\![\dots]\!])$. Since y_j occurs in t, it occurs in t_ν for some $\nu \in [l]$. By IH2, y_j occurs in $t_\nu[\![\dots]\!]$ and thus in $t[\![\dots]\!]$. If $t = \langle q',x_i\rangle(t_1,\dots,t_l)$ with $\langle q',x_i\rangle \in \langle Q,X_k\rangle^{(l)}$, then $t[\![\dots]\!] = M_{q'}(s_i)[y_\nu \leftarrow t_\nu[\![\dots]\!] \mid \nu \in [l]]$. By the fact that y_j occurs in t, and by IH2, y_j occurs in $t_\nu[\![\dots]\!]$ for some $\nu \in [l]$. By IH1, y_ν occurs in $M_{q'}(s_i)$ and thus $t_\nu[\![\dots]\!]$ is a subtree of $t[\![\dots]\!]$. □

It was proved in Lemma 6.6 of [EM99] that every MTT M with regular look-ahead can be turned into a nondeleting one which realizes the same translation as M. This can be stated in the following way (cf. Lemma 6), where MTT_{nd} denotes the class of all translations realized by nondeleting MTTs.

Lemma 9. *$MTT \subseteq D_tQRELAB \circ MTT_{\mathrm{nd}}$.*

A macro tree transducer M is *simple in the parameters* (for short *sp*), if in the right-hand side of every q-rule, for every state q of rank $m \geq 1$, each parameter y_j, $j \in [m]$, occurs exactly once (i.e., the rules of M are linear and nondeleting in Y_m); we say that M is an MTT$_{\mathrm{sp}}$. The class of all translations that can be realized by MTT$_{\mathrm{sp}}$s is denoted by MTT_{sp}. Note that in [EM99], sp macro tree transducers are said to be 'nondeleting surp'.

Let us finally consider an example of an MTT$_{\mathrm{sp}}$.

Example 10. Let $M = (Q,\Sigma,\Sigma,q_0,R)$ be the MTT with $Q = \{q_0^{(0)}, q^{(2)}\}$, $\Sigma = \{\sigma^{(2)}, a^{(0)}, b^{(0)}\}$, and R consisting of the following rules.

$$\begin{array}{ll}
\langle q_0,\sigma(x_1,x_2)\rangle & \to \langle q,x_2\rangle(\langle q_0,x_1\rangle,\langle q_0,x_1\rangle)\\
\langle q,\sigma(x_1,x_2)\rangle(y_1,y_2) & \to \langle q,x_2\rangle(\sigma(y_1,\langle q_0,x_1\rangle),\sigma(\langle q_0,x_1\rangle,y_2))\\
\langle q_0,a\rangle & \to a\\
\langle q,a\rangle(y_1,y_2) & \to \sigma(y_1,y_2)\\
\langle q_0,b\rangle & \to b\\
\langle q,b\rangle(y_1,y_2) & \to \sigma(y_2,y_1)
\end{array}$$

Note that M is sp because both y_1 and y_2 appear exactly once in the right-hand side of each q-rule of M. Consider the input tree $t = \sigma(a,\sigma(b,\sigma(b,b)))$. Then a derivation by M looks as follows.

$$\begin{array}{rl}
\langle q_0,t\rangle \Rightarrow_M & \langle q,\sigma(b,\sigma(b,b))\rangle(\langle q_0,a\rangle,\langle q_0,a\rangle)\\
\Rightarrow_M^* & \langle q,\sigma(b,\sigma(b,b))\rangle(a,a)\\
\Rightarrow_M & \langle q,\sigma(b,b)\rangle(\sigma(a,\langle q_0,b\rangle),\sigma(\langle q_0,b\rangle,a))\\
\Rightarrow_M^* & \langle q,\sigma(b,b)\rangle(\sigma(a,b),\sigma(b,a))\\
\Rightarrow_M^* & \langle q,b\rangle(\sigma(\sigma(a,b),b),\sigma(b,\sigma(b,a)))\\
\Rightarrow_M & \sigma(\sigma(b,\sigma(b,a)),\sigma(\sigma(a,b),b))
\end{array}$$

Thus, $\tau_M(t) = \sigma(\sigma(b,\sigma(b,a)),\sigma(\sigma(a,b),b))$. This tree can be computed in terms of q-translations and q_0-translations as follows. First, $M_{q_0}(b) = b$ and $M_q(b) = \sigma(y_2,y_1)$. Hence

$$\begin{array}{l}
M_q(\sigma(b,b))\\
= \mathrm{rhs}_M(q,\sigma)[\![\langle q',x_i\rangle \leftarrow M_{q'}(b) \mid q' \in \{q_0,q\}, 1 \leq i \leq 2]\!]\\
= \langle q,x_2\rangle(\sigma(y_1,\langle q_0,x_1\rangle),\sigma(\langle q_0,x_1\rangle,y_2))[\![\langle q,x_2\rangle \leftarrow \sigma(y_2,y_1),\langle q_0,x_1\rangle \leftarrow b]\!]\\
= \sigma(y_2,y_1)[y_1 \leftarrow \sigma(y_1,b), y_2 \leftarrow \sigma(b,y_2)]\\
= \sigma(\sigma(b,y_2),\sigma(y_1,b)).
\end{array}$$

Next,

$$\begin{aligned}M_q(\sigma(b,\sigma(b,b))) &= \mathrm{rhs}_M(q,\sigma)[\![\langle q,x_2\rangle \leftarrow M_q(\sigma(b,b)), \langle q_0,x_1\rangle \leftarrow M_{q_0}(b)]\!] \\ &= M_q(\sigma(b,b))[y_1 \leftarrow \sigma(y_1,b), y_2 \leftarrow \sigma(b,y_2)] \\ &= \sigma(\sigma(b,y_2),\sigma(y_1,b))[y_1 \leftarrow \sigma(y_1,b), y_2 \leftarrow \sigma(b,y_2)] \\ &= \sigma(\sigma(b,\sigma(b,y_2)),\sigma(\sigma(y_1,b),b)).\end{aligned}$$

Finally, $M_{q_0}(a) = a$ and

$$\begin{aligned}\tau_M(t) &= M_{q_0}(t) \\ &= \mathrm{rhs}_M(q_0,\sigma)[\![\langle q,x_2\rangle \leftarrow M_q(\sigma(b,\sigma(b,b))), \langle q_0,x_1\rangle \leftarrow M_{q_0}(a)]\!] \\ &= M_q(\sigma(b,\sigma(b,b))[y_1 \leftarrow a, y_2 \leftarrow a] \\ &= \sigma(\sigma(b,\sigma(b,a)),\sigma(\sigma(a,b),b)).\end{aligned}$$

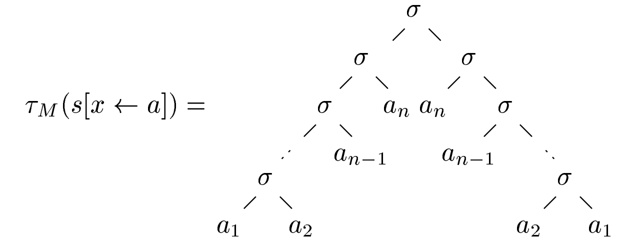

Fig. 1. Translations of M for the input trees $s[x \leftarrow a]$

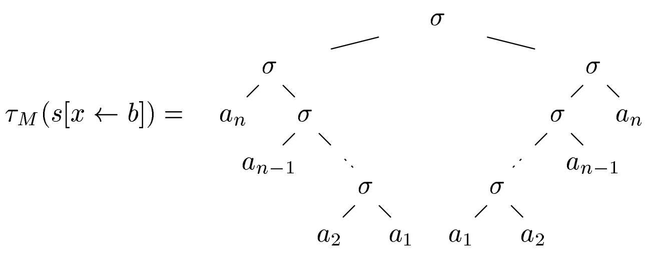

Fig. 2. Translations of M for the input trees $s[x \leftarrow b]$

In Figures 1 and 2 it is shown how the translations for input trees of the form

$$s = \sigma(a_1, \sigma(a_2, \ldots \sigma(a_n, x) \ldots))$$

with $a_1, \ldots, a_n \in \Sigma^{(0)}$, $n \geq 1$, and $x = a$ and $x = b$, respectively, look like. If $x = a$ then $y\tau_M(s) = ww^r$ and if $x = b$ then $y\tau_M(s) = w^r w$, where $w = a_1 \cdots a_n$ (and recall from the Preliminaries that w^r denotes the reverse of w). □

4 Closure Properties

In this section we prove two closure properties of MTTs. First, that the class *MTT* of macro tree translations is closed under composition with finite state relabelings, and second, that, with respect to output string languages, the class $MTT(\mathcal{L})$, for an arbitrary class $\mathcal{L}$ of tree languages, is closed under translations realized by $\mathrm{MTT_{sp}}$s. To prove the second closure property, it will be shown in Theorem 15 that, when applied to a class of tree languages closed under finite state relabelings, $\mathrm{MTT_{sp}}$s generate the same class of string languages as top-down tree transducers.

Let us move to the first closure property. We want to show that for an MTT M and a finite state relabeling N there is an MTT M' with $\tau_{M'} = \tau_M \circ \tau_N$ (cf. Lemma 6, which proves this for the opposite order of the composition, i.e., that $\tau_N \circ \tau_M$ can be realized by an MTT). In fact, the result $MTT \circ D_t QRELAB \subseteq MTT$ can also be obtained from known results as follows. By Theorem 4.8 of [EV85], $MTT = T \circ YIELD$ (for $YIELD$, see Section 7). Thus, $MTT \circ D_t QRELAB$ equals $T \circ YIELD \circ D_t QRELAB$. By Lemma 3.11 of [DE98] this is included in $T \circ QRELAB \circ YIELD$, where $QRELAB$ denotes the class of nondeterministic finite state relabelings. More precisely, we only need to consider total functions in $T \circ QRELAB \circ YIELD$, because MTT$\circ D_t QRELAB$ consists of total functions only. Thus, $MTT \circ D_t QRELAB \subseteq (T \circ QRELAB \circ YIELD) \cap \mathcal{F}$, where $\mathcal{F}$ is the class of all total functions. From the theorem of [Eng78] it follows that, for every function f in $T \circ QRELAB \circ YIELD$, there is a top-down tree transducer M with regular look-ahead such that $f \in \tau_M \circ YIELD$. Since the look-ahead can be simulated by a relabeling in $D_t QRELAB$ we obtain $f \in D_t QRELAB \circ T \circ YIELD = D_t QRELAB \circ MTT$, which is in MTT by Lemma 6. Hence, $MTT \circ D_t QRELAB \subseteq MTT$. We now give an elementary proof of this fact.

Lemma 11. *$MTT \circ D_t QRELAB \subseteq MTT$.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT and let $N = (Q_N, \Delta, \Omega, R_N)$ be a finite state relabeling. We will construct a finite state relabeling N' and an MTT M' such that $\tau_{N'} \circ \tau_{M'} = \tau_M \circ \tau_N$. By Lemma 6, $\tau_M \circ \tau_N \in MTT$.

The (standard) idea is to construct the MTT M' from M by running the finite state relabeling N on the right-hand sides ζ of the rules of M. To do this we need to know, for y_j occurring in ζ, in which state the relabeling N arrives after processing the tree that will be substituted for y_j. This (top-down) information can be represented by a mapping $\varphi : [m] \to Q_N$ (if ζ is the right-hand side of a q-rule and q is of rank m) and can be coded into the states of M'. More precisely, we choose the set Q' of states of M' as

$$Q' = \{(q, \varphi)^{(m)} \mid q \in Q^{(m)}, m \geq 0, \varphi : [m] \to Q_N\}.$$

Similarly, for a subtree $\langle q', x_i\rangle(t_1, \ldots, t_l)$ of ζ we need to know, given that N arrives in state p_ν after processing t_ν for $\nu \in [l]$, in which state N arrives after processing the tree $M_{q'}(s_i)$ that will be substituted for $\langle q', x_i\rangle$. This information can be represented by a function ϕ_i for $i \in [k]$ which associates with every $q' \in Q^{(l)}$ a mapping of type $Q_N^l \to Q_N$. We use the (bottom-up) finite state relabeling N' to replace every symbol σ by the new symbol $(\sigma, \phi_1, \ldots, \phi_k)$, where ϕ_i is the corresponding function determining the state change of N on the trees $M_{q'}(s_i)$.

In order to translate the right-hand side of a (q, σ)-rule of M, with $q \in Q^{(m)}$, $\sigma \in \Sigma^{(k)}$, and $m, k \geq 0$, the finite state relabeling N is extended as follows. Let $\phi_1, \ldots, \phi_k$ be functions which associate with every $q' \in Q^{(l)}$ a mapping $\phi(q') : Q_N^l \to Q_N$ and let $\varphi : [m] \to Q_N$. Then $N_{\varphi,(\phi_1,\ldots,\phi_k)} = (Q_N, \langle Q, X_k\rangle \cup \Delta \cup P, \langle Q', X_k\rangle \cup \Omega \cup P, R_N \cup S)$ is the extension of N to input trees in $T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$, where $P = \{y_j^{(0)} \mid j \in [m]\}$ and the set S of additional rules is defined as follows. For every $j \in [m]$, $y_j \to \langle \varphi(j), y_j\rangle$ is in S, and for every $\langle q', x_i\rangle \in \langle Q, X_k\rangle^{(l)}$, $l \geq 0$, and $p_1, \ldots, p_l \in Q_N$ the rule

$$\langle q', x_i\rangle(\langle p_1, x_1\rangle, \ldots, \langle p_l, x_l\rangle) \to \langle \phi_i(q')(p_1, \ldots, p_l), \langle (q', \varphi'), x_i\rangle(x_1, \ldots, x_l)\rangle$$

is in S, with $\varphi' : [l] \to Q_N$ and $\varphi'(\nu) = p_\nu$ for all $\nu \in [l]$.

Define $N' = (Q_{N'}, \Sigma, \Gamma, R_{N'})$, where

– $Q_{N'}$ is the set of all functions ϕ which assign to every $q \in Q^{(l)}$ with $l \geq 0$ a mapping $\phi(q) : Q_N^l \to Q_N$.

– $\Gamma = \{(\sigma, \phi_1, \ldots, \phi_k)^{(k)} \mid \sigma \in \Sigma^{(k)},\ k \geq 0,\ \phi_1, \ldots, \phi_k \in Q_{N'}\}$ and
– for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $\phi_1, \ldots, \phi_k \in Q_{N'}$, the rule

$$\sigma(\langle \phi_1, x_1 \rangle, \ldots, \langle \phi_k, x_k \rangle) \to \langle \phi, (\sigma, \phi_1, \ldots, \phi_k)(x_1, \ldots, x_k) \rangle$$

is in $R_{N'}$, where ϕ is defined as follows. For every $q \in Q^{(m)}$, $m \geq 0$, and $p_1, \ldots, p_m \in Q_N$, $\phi(q)(p_1, \ldots, p_m) = p$, with $p \in Q_N$ such that

$$\mathrm{rhs}_M(q, \sigma) \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \_ \rangle$$

and $\varphi : [m] \to Q_N$ with $\varphi(j) = p_j$ for every $j \in [m]$. Recall from Section 2.3 that $t \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \_ \rangle$ means that there is a t' such that $t \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, t' \rangle$.

We now define the MTT $M' = (Q', \Gamma, \Omega, q_0', R')$ with Q' as above, $q_0' = (q_0, \varnothing)$, and R' as follows. For every $(q, \varphi) \in Q'^{(m)}$, $(\sigma, \phi_1, \ldots, \phi_k) \in \Sigma'^{(k)}$, and $m, k \geq 0$,

$$\langle (q, \varphi), (\sigma, \phi_1, \ldots, \phi_k)(x_1, \ldots, x_k) \rangle (y_1, \ldots, y_m) \to \zeta$$

is a rule in R', where $\mathrm{rhs}_M(q, \sigma) \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \zeta \rangle$ for some $p \in Q_N$, i.e., $\zeta = \tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(\mathrm{rhs}_M(q, \sigma))$.

Let us now prove the correctness of this construction. For $m \geq 0$ and $\varphi : [m] \to Q_N$ let $N_\varphi = N_{\varphi,(\phi_1,\ldots,\phi_k)}$ with $k = 0$, i.e., N_φ is the extension of N to trees in $T_\Delta(Y_m)$ obtained by adding the rules $y_j \to \langle \varphi(j), y_j \rangle$ for every $j \in [m]$. The correctness, i.e., that $\tau_{M'}(\tau_{N'}(s)) = \tau_N(\tau_M(s))$ for every $s \in T_\Sigma$, follows from Claim 1(a) for $(q, \varphi) = (q_0, \varnothing)$, because $N_\varnothing = N$.

Claim 1: For every $s \in T_\Sigma$, $(q, \varphi) \in Q'^{(m)}$ with $m \geq 0$, and $\phi \in Q_{N'}$,

(a) $M'_{(q,\varphi)}(\tau_{N'}(s)) = \tau_{N_\varphi}(M_q(s))$ and
(b) if $s \Rightarrow^*_{N'} \langle \phi, \_ \rangle$, then $M_q(s) \Rightarrow^*_{N_\varphi} \langle \phi(q)(\varphi(1), \ldots, \varphi(m)), \_ \rangle$.

The claim is proved by induction on the structure of s. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. The induction hypothesis is denoted by IH1. Let $\phi_1, \ldots, \phi_k \in Q_{N'}$ such that $s_i \Rightarrow^*_{N'} \langle \phi_i, \_ \rangle$ for every $i \in [k]$.

First, part (a) of the claim is proved. It follows from the definition of N' that $M'_{(q,\varphi)}(\tau_{N'}(s)) = M'_{(q,\varphi)}((\sigma, \phi_1, \ldots, \phi_k)(\tau_{N'}(s_1), \ldots, \tau_{N'}(s_k)))$. By Definition 3 this equals $\mathrm{rhs}_{M'}((q, \varphi), (\sigma, \phi_1, \ldots, \phi_k))[\![_{M'}]\!]$, where $[\![_{M'}]\!]$ denotes the substitution $[\![\langle (q', \varphi'), x_i \rangle \leftarrow M'_{(q',\varphi')}(\tau_{N'}(s_i)) \mid \langle (q', \varphi'), x_i \rangle \in \langle Q', X_k \rangle]\!]$. By the definition of M' and IH1(a) this equals $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(\mathrm{rhs}_M(q, \sigma))[\![_{NM}]\!]$, where $[\![_{NM}]\!] = [\![\langle (q', \varphi'), x_i \rangle \leftarrow \tau_{N_{\varphi'}}(M_{q'}(s_i)) \mid \langle (q', \varphi'), x_i \rangle \in \langle Q', X_k \rangle]\!]$. It follows from Claim 2(a) below, for $t = \mathrm{rhs}_M(q, \sigma)$, that $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(\mathrm{rhs}_M(q, \sigma))[\![_{NM}]\!]$ equals $\tau_{N_\varphi}(\mathrm{rhs}_M(q, \sigma)[\![_M]\!])$ with $[\![_M]\!] = [\![\langle q', x_i \rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i \rangle \in \langle Q, X_k \rangle]\!]$. This equals $\tau_{N_\varphi}(M_q(s))$.

For the (b) part, if $s \Rightarrow^*_{N'} \langle \phi, \_ \rangle$, then there are $\zeta_1, \ldots, \zeta_k, \zeta \in T_{\Sigma'}$ with $s_i \Rightarrow^*_{N'} \langle \phi_i, \zeta_i \rangle$ for all $i \in [k]$ and $\sigma(\langle \phi_1, \zeta_1 \rangle, \ldots, \langle \phi_k, \zeta_k \rangle) \Rightarrow_{N'} \langle \phi, \zeta \rangle$. By the definition of N', if $\sigma(\langle \phi_1, \zeta_1 \rangle, \ldots, \langle \phi_k, \zeta_k \rangle) \Rightarrow_{N'} \langle \phi, \zeta \rangle$, then, for every $(q, \varphi) \in Q'^{(m)}$ and $m \geq 0$, $\mathrm{rhs}_M(q, \sigma) \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle \phi(q)(\varphi(1), \ldots, \varphi(m)), \_ \rangle$. By Claim 2(b) for $t = \mathrm{rhs}_M(q, \sigma)$ and $p = \phi(q)(\varphi(1), \ldots, \varphi(m))$: $M_q(s) = \mathrm{rhs}_M(q, \sigma)[\![_M]\!] \Rightarrow^*_{N_\varphi} \langle \phi(q)(\varphi(1), \ldots, \varphi(m)), \_ \rangle$. This concludes the proof of Claim 1.

Claim 2: For every $m \geq 0$, $\varphi : [m] \to Q_N$, $p \in Q_N$, and $t \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$,

(a) $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t)[\![_{NM}]\!] = \tau_{N_\varphi}(t[\![_M]\!])$ and
(b) if $t \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \_ \rangle$, then $t[\![_M]\!] \Rightarrow^*_{N_\varphi} \langle p, \_ \rangle$.

Claim 2 is proved by induction on the structure of t. We denote the induction hypothesis by IH2.

If $t = y_j \in Y_m$, then $t \Rightarrow_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle \varphi(j), t\rangle$ and $t[\![_{NM}]\!] = t$, and $t[\![_M]\!] = t \Rightarrow_{N_\varphi} \langle \varphi(j), t\rangle$, by the definition of $N_{\varphi,(\phi_1,\ldots,\phi_k)}$ and N_φ, respectively. This proves both (a) and (b). Let $l \geq 0$ and $t_1, \ldots, t_l \in T_{\langle Q,X_k\rangle \cup \Delta}(Y_m)$.

If $t = \delta(t_1,\ldots,t_l)$ with $\delta \in \Delta^{(l)}$, then $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t)[\![_{NM}]\!] = \gamma(\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_1), \ldots, \tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_l))[\![_{NM}]\!]$, with $\gamma \in \Gamma$ such that $\delta(\langle p_1,x_1\rangle,\ldots,\langle p_k,x_k\rangle) \to \langle p, \gamma(x_1, \ldots, x_k)\rangle$ is a rule of N (and thus of $N_{\varphi,(\phi_1,\ldots,\phi_k)}$) and $t_\nu \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p_\nu, \_\rangle$ for $\nu \in [k]$. By IH2(a), $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_\nu)[\![_{NM}]\!]$ equals $\tau_{N_\varphi}(t_\nu[\![_M]\!])$ and by IH2(b) $t_\nu[\![_M]\!] \Rightarrow^*_{N_\varphi} \langle p_\nu, \_\rangle$. Since N_φ has the same rule $\delta(\langle p_1,x_1\rangle,\ldots,\langle p_k,x_k\rangle) \to \langle p, \gamma(x_1,\ldots,x_k)\rangle$ of N it follows that the derivations by $\Rightarrow_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}$ and $\Rightarrow_{N_\varphi}$ both end with $\langle p, \_\rangle$ which shows the (b) part, and $\gamma(\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_1),\ldots,\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_l))[\![_{NM}]\!] = \gamma(\tau_{N_\varphi}(t_1[\![_M]\!]),\ldots,\tau_{N_\varphi}(t_l[\![_M]\!])) = \tau_{N_\varphi}(\delta(t_1[\![_M]\!],\ldots,t_l[\![_M]\!])) = \tau_{N_\varphi}(t[\![_M]\!])$, which shows the (a) part.

Finally, let $t = \langle q', x_i\rangle(t_1,\ldots,t_l)$ with $\langle q', x_i\rangle \in \langle Q, X_k\rangle^{(l)}$. If $t \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \_\rangle$, then there are $p_1,\ldots,p_l \in Q_N$ and $\zeta_1,\ldots,\zeta_l,\zeta \in T_{\langle Q,X_k\rangle\cup\Delta}(Y_m)$ such that $t_\nu \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p_\nu, \zeta_\nu\rangle$ for $\nu \in [l]$ and $\langle q', x_i\rangle(\langle p_1,\zeta_1\rangle,\ldots,\langle p_l,\zeta_l\rangle) \Rightarrow_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p, \zeta\rangle$, which, by definition of $N_{\varphi,(\phi_1,\ldots,\phi_k)}$, implies that $p = \phi_i(q')(p_1,\ldots,p_l)$. Now $t_\nu \Rightarrow^*_{N_{\varphi,(\phi_1,\ldots,\phi_k)}} \langle p_\nu, \_\rangle$ implies $t_\nu[\![_M]\!] \Rightarrow^*_{N_\varphi} \langle p_\nu, \_\rangle$ by IH2(b), i.e., $t_\nu[\![_M]\!] \Rightarrow^*_{N_\varphi} \langle p_\nu, \tau_{N_\varphi}(t_\nu[\![_M]\!])\rangle$, and so

$$t[\![_M]\!] \Rightarrow^*_{N_\varphi} M_{q'}(s_i)[y_\nu \leftarrow \langle p_\nu, \tau_{N_\varphi}(t_\nu[\![_M]\!])\rangle \mid \nu \in [l]] = \xi,$$

by the definition of $[\![_M]\!]$. Now let $\varphi' : [l] \to Q_N$ with $\varphi'(\nu) = p_\nu$ for every $\nu \in [l]$. Since $s_i \Rightarrow^*_{N'} \langle \phi_i, \_\rangle$, it follows from IH1(b) and the definition of φ' that $M_{q'}(s_i) \Rightarrow^*_{N_{\varphi'}} \xi' = M_{q'}(s_i)[y_\nu \leftarrow \langle p_\nu, y_\nu\rangle \mid \nu \in [l]]$ and $\xi' \Rightarrow^*_{N_{\varphi'}} \langle p, \tau_{N_{\varphi'}}(M_{q'}(s_i))\rangle$. Clearly, the latter derivation also holds for N_φ, and so $\xi \Rightarrow^*_{N_\varphi} \langle p, \zeta\rangle$, where $\zeta = \tau_{N_{\varphi'}}(M_{q'}(s_i))[y_\nu \leftarrow \tau_{N_\varphi}(t_\nu[\![_M]\!]) \mid \nu \in [l]]$. Hence $t[\![_M]\!] \Rightarrow^*_{N_\varphi} \xi \Rightarrow^*_{N_\varphi} \langle p, \zeta\rangle$. This proves part (b).

By IH2(a), $\tau_{N_\varphi}(t_\nu[\![_M]\!]) = \tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_\nu)[\![_{NM}]\!]$ for every $\nu \in [l]$. Thus, $\zeta = \tau_{N_{\varphi'}}(M_{q'}(s_i))[y_\nu \leftarrow \tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_\nu)[\![_{NM}]\!] \mid \nu \in [l]]$. By the definition of $[\![_{NM}]\!]$ this equals $\langle (q',\varphi'), x_i\rangle(\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_1),\ldots,\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t_l))[\![_{NM}]\!]$ which, by the definition of $N_{\varphi,(\phi_1,\ldots,\phi_k)}$, equals $\tau_{N_{\varphi,(\phi_1,\ldots,\phi_k)}}(t)[\![_{NM}]\!]$. This ends the proof of Claim 2. □

It was mentioned in the Conclusions of [EV85] as an open problem whether the class of macro tree translations is closed under composition with T^R, the class of top-down tree translations with regular look-ahead. Since T^R equals $D_t\,QRELAB \circ T$ (see Theorem 2.6 of [Eng77]) it follows from Lemma 11 that $MTT \circ T^R \subseteq MTT \circ D_t\,QRELAB \circ T \subseteq MTT \circ T$, and by Lemma 5, $MTT \circ T \subseteq MTT$.

Corollary 12. $MTT \circ T^R \subseteq MTT$.

We now move to the second closure property. The main part of the proof of this closure property consists of proving Theorem 15 which says that, for a class $\mathcal{L}$ of tree languages closed under finite state relabelings, $yMTT_{\rm sp}(\mathcal{L}) = yT(\mathcal{L})$. In essence this is proved in the following lemma, which shows how to generate by a top-down tree transducer the string language generated by an MTT$_{\rm sp}$.

Lemma 13. $MTT_{\rm sp} \circ y \subseteq D_t\,QRELAB \circ T \circ y$.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT$_{\rm sp}$. We will construct a finite state relabeling N and a top-down tree transducer M' such that for every $s \in T_\Sigma$,

$y(\tau_{M'}(\tau_N(s))) = y\tau_M(s)$. The idea is as follows. Let $q \in Q^{(m)}$ and $s \in T_\Sigma$. Then, since M is sp, $yM_q(s)$ is of the form

$$w = w_0 y_{j_1} w_1 y_{j_2} w_2 \cdots y_{j_m} w_m,$$

where $j_1, \ldots, j_m \in [m]$ are pairwise different and $w_0, \ldots, w_m \in (\Delta^{(0)})^*$. For a string of the form w and for $0 \leq \nu \leq m$ we denote by $\mathrm{part}_\nu(w)$ the string w_ν. For every ν the top-down tree transducer M' has a state (q, ν) which computes w_ν. The information on the order of the parameters, i.e., the string $\mathrm{res}_Y(yM_q(s)) \in Y_m^*$, will be determined by the finite state relabeling N in such a way that $\sigma \in \Sigma^{(k)}$ is relabeled by $(\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)$, where for each $i \in [k]$, per_i is a mapping associating with every $q \in Q^{(m)}$ a permutation of the string $y_1 \cdots y_m$. For instance, if s_i equals the tree s from above, then the σ in $\sigma(s_1, \ldots, s_i, \ldots, s_k)$ is relabeled by $(\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)$ and $\mathrm{per}_i(q) = \mathrm{res}_Y(w) = y_{j_1} \cdots y_{j_m}$.

Formally, $N = (Q_N, \Sigma, \Gamma, R_N)$, where

- Q_N is the set of all mappings per which associate with every $q \in Q^{(m)}$ a string in Y_m^* which is a permutation of $y_1 \cdots y_m$.
- $\Gamma = \{(\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)^{(k)} \mid \sigma \in \Sigma^{(k)}, k \geq 0, \mathrm{per}_1, \ldots, \mathrm{per}_k \in Q_N\}$.
- For every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $\mathrm{per}_1, \ldots, \mathrm{per}_k \in Q_N$ let

  $$\sigma(\langle \mathrm{per}_1, x_1\rangle, \ldots, \langle \mathrm{per}_k, x_k\rangle) \to \langle \mathrm{per}, (\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)(x_1, \ldots, x_k)\rangle$$

  be in R_N, where for every $q \in Q^{(m)}$, $\mathrm{per}(q) = \mathrm{res}_Y(y(\mathrm{rhs}_M(q, \sigma)\Theta))$ and Θ denotes the second-order substitution

  $$[\![\langle q', x_i\rangle \leftarrow \mathrm{comb}_b(\mathrm{per}_i(q')) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!],$$

  where b is an arbitrary binary symbol (see Section 2.1 for comb_b).

It follows from Claim 1 that N realizes the relabeling as described.

Claim 1: Let $q \in Q^{(m)}$, $m \geq 0$, and $s \in T_\Sigma$. If $s \Rightarrow_N^* \langle \mathrm{per}, \_\rangle$, then $\mathrm{per}(q) = \mathrm{res}_Y(yM_q(s))$.

The proof of this claim is by induction on the structure of s. The induction hypothesis is denoted by IH1. Let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. Then $s \Rightarrow_N^* \langle \mathrm{per}, \_\rangle$ if there are $\mathrm{per}_1, \ldots, \mathrm{per}_k \in Q_N$ such that $s_i \Rightarrow_N^* \langle \mathrm{per}_i, \tau_N(s_i)\rangle$ for $i \in [k]$ and $\sigma(\langle \mathrm{per}_1, \tau_N(s_1)\rangle, \ldots, \langle \mathrm{per}_k, \tau_N(s_k)\rangle) \Rightarrow_N \langle \mathrm{per}, \_\rangle$, where $\mathrm{per}(q) = \mathrm{res}_Y(y(t\Theta))$, $t = \mathrm{rhs}_M(q, \sigma)$, and Θ as in the definition of N. Let $[\![\ldots]\!] = [\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!]$. By Claim 2, $\mathrm{res}_Y(y(t\Theta)) = \mathrm{res}_Y(y(t[\![\ldots]\!])) = \mathrm{res}_Y(yM_q(s))$.

Claim 2: For every $t \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$, $\mathrm{res}_Y(y(t\Theta)) = \mathrm{res}_Y(y(t[\![\ldots]\!]))$.

This claim is proved by induction on the structure of t. The induction hypothesis is denoted by IH2. If $t = y_j \in Y_m$, then $\mathrm{res}_Y(y(t\Theta)) = \mathrm{res}_Y(yt) = \mathrm{res}_Y(y(t[\![\ldots]\!]))$. Let $l \geq 0$ and $t_1, \ldots, t_l \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$.

If $t = \delta(t_1, \ldots, t_l)$, then $\mathrm{res}_Y(y(t\Theta)) = \mathrm{res}_Y(y(\delta(t_1\Theta, \ldots, t_l\Theta))) = \mathrm{res}_Y(y(t_1\Theta) \cdots y(t_l\Theta)) = \mathrm{res}_Y(y(t_1\Theta)) \cdots \mathrm{res}_Y(y(t_l\Theta))$. By IH2 this equals $\mathrm{res}_Y(y(t_1[\![\ldots]\!])) \cdots \mathrm{res}_Y(y(t_l[\![\ldots]\!])) = \mathrm{res}_Y(y(t_1[\![\ldots]\!]) \cdots y(t_l[\![\ldots]\!])) = \mathrm{res}_Y(t[\![\ldots]\!])$.

If $t = \langle q', x_i\rangle(t_1, \ldots, t_l)$, then $\mathrm{res}_Y(y(t\Theta)) = \mathrm{res}_Y(y(\mathrm{comb}_b(\mathrm{per}_i(q'))[y_j \leftarrow t_j\Theta \mid j \in [l]]))$. By applying yield we get $\mathrm{res}_Y(\mathrm{per}_i(q')[y_j \leftarrow y(t_j\Theta) \mid j \in [l]])$ and application of res_Y gives $\mathrm{per}_i(q')[y_j \leftarrow \mathrm{res}_Y(y(t_j\Theta)) \mid j \in [l]]$. By IH1 and IH2 this equals $\mathrm{res}_Y(yM_{q'}(s_i))[y_j \leftarrow \mathrm{res}_Y(y(t_j[\![\ldots]\!])) \mid j \in [l]] = \mathrm{res}_Y(y(M_{q'}(s_i)[y_j \leftarrow t_j[\![\ldots]\!] \mid j \in [l]])) = \mathrm{res}_Y(y(t[\![\ldots]\!]))$.

We now define the top-down tree transducer $M' = (Q', \Gamma, \Delta', (q_0, 0), R')$, where

- $Q' = \{(q, \nu)^{(0)} \mid q \in Q^{(m)}, 0 \leq \nu \leq m\}$,

– $\Delta' = \Delta^{(0)} \cup \{b^{(2)}, e^{(0)}\}$, and
– for every $(q,\nu) \in Q'$, $(\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k) \in \Gamma^{(k)}$, and $k \geq 0$, the rule

$$\langle (q,\nu), (\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)(x_1, \ldots, x_k)\rangle \to \zeta$$

is in R', where $\zeta = \mathrm{comb}_b(\mathrm{part}_\nu(y(\xi\Phi)))$, $\xi = \mathrm{rhs}_M(q,\sigma)$, and Φ is the substitution

$$[\![\langle q', x_i\rangle \leftarrow \mathrm{comb}_b(\langle (q',0), x_i\rangle \mathrm{per}_i(q')(1)\langle (q',1), x_i\rangle \mathrm{per}_i(q')(2) \cdots \\ \mathrm{per}_i(q')(m)\langle (q',m), x_i\rangle) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle^{(m)}, m \geq 0]\!].$$

Recall from the Preliminaries that $\mathrm{per}_i(q)(j)$ denotes the j-th symbol of $\mathrm{per}_i(q)$.

We now prove the correctness of M', i.e., that for every $s \in T_\Sigma$, $y(\tau_{M'}(\tau_N(s))) = y\tau_M(s)$. It follows from Claim 3 for $(q,\nu) = (q_0, 0)$.

Claim 3: For every $(q,\nu) \in Q'$ and $s \in T_\Sigma$, $yM'_{(q,\nu)}(\tau_N(s)) = \mathrm{part}_\nu(yM_q(s))$.

The proof of this claim is by induction on the structure of s. Let $s = \sigma(s_1, \ldots, s_k)$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. Then $yM'_{(q,\nu)}(\tau_N(s)) = yM'_{(q,\nu)}((\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k)(\tau_N(s_1), \ldots, \tau_N(s_k)))$. By Definition 3 and the fact that M' is a top-down tree transducer, i.e., all elements of $\langle Q, X_k\rangle$ are of rank zero, this equals $y(\zeta[\ldots])$, where $\zeta = \mathrm{rhs}_{M'}((q,\nu), (\sigma, \mathrm{per}_1, \ldots, \mathrm{per}_k))$ and $[\ldots] = [\langle (q',\nu'), x_i\rangle \leftarrow M'_{(q',\nu')}(\tau_N(s_i)) \mid \langle (q',\nu'), x_i\rangle \in \langle Q', X_k\rangle]$. By the definition of the rules of M', $\zeta = \mathrm{comb}_b(\mathrm{part}_\nu(y(\xi\Phi)))$, where $\xi = \mathrm{rhs}_M(q,\sigma)$ and Φ as in the definition of M'. By induction, $yM'_{(q',\nu')}(\tau_N(s_i)) = \mathrm{part}_{\nu'}(yM_{q'}(s_i))$ for every $\langle (q',\nu'), x_i\rangle \in \langle Q', X_k\rangle$. Thus, we can apply Lemma 1(a) and replace $M'_{(q',\nu')}(\tau_N(s_i))$ by $\mathrm{comb}_b(\mathrm{part}_{\nu'}(yM_{q'}(s_i)))$ in $[\ldots]$, to get $y(\mathrm{comb}_b(\mathrm{part}_\nu(y(\xi\Phi)))[\_])$ with $[\_] = [\langle (q',\nu'), x_i\rangle \leftarrow \mathrm{comb}_b(\mathrm{part}_{\nu'}(yM_{q'}(s_i))) \mid \langle (q',\nu'), x_i\rangle \in \langle Q', X_k\rangle]$). We can now apply yield and then move the (string) substitution that corresponds to $[\_]$ inside the application of part_ν and yield, because $\mathrm{part}_{\nu'}(yM_{q'}(s_i)) \in (\Delta^{(0)})^*$. We get $\mathrm{part}_\nu(y(\xi\Phi[\_]))$. By application of $[\_]$ we obtain that $\xi\Phi[\_] = \xi[\![\langle q', x_i\rangle \leftarrow t_{\langle q', x_i\rangle} \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle^{(m)}, m \geq 0]\!]$, where, by the definition of Φ, each tree $t_{\langle q', x_i\rangle}$ has yield $w_0 y_{j_1} w_1 \cdots y_{j_m} w_m$ with $w_{\nu'} = \mathrm{part}_{\nu'}(yM_{q'}(s_i))$ for $0 \leq \nu' \leq m$, and, $y_{j_{\nu'}} = \mathrm{per}_i(q')(\nu')$ for $\nu' \in [m]$. By Claim 1, $\mathrm{per}_i(q')(\nu')$ equals $\mathrm{res}_Y(yM_{q'}(s_i))(\nu')$. Hence, $yt_{\langle q', x_i\rangle}$ equals

$$\mathrm{part}_0(yM_{q'}(s_i))\mathrm{res}_Y(yM_{q'}(s_i))(1)\mathrm{part}_1(yM_{q'}(s_i)) \cdots \\ \mathrm{res}_Y(yM_{q'}(s_i))(m)\mathrm{part}_m(yM_{q'}(s_i)),$$

which equals $yM_{q'}(s_i)$. By Lemma 1(b) we can replace $t_{\langle q', x_i\rangle}$ by $M_{q'}(s_i)$ to get $\mathrm{part}_\nu(y(\xi[\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!])) = \mathrm{part}_\nu(yM_q(s))$. □

Let us take a look at an example of an application of the construction in the proof of Lemma 13.

Example 14. Let M be the MTT$_{\mathrm{sp}}$ of Example 10. We construct the finite state relabeling N and the top-down tree transducer M' following the construction in the proof of Lemma 13. Let $N = (Q_N, \Sigma, \Gamma, R_N)$ be the finite state relabeling with $Q_N = \{q_{12}, q_{21}\}$, $q_{12} = \{(q_0, \varepsilon), (q, y_1y_2)\}$, $q_{21} = \{(q_0, \varepsilon), (q, y_2y_1)\}$, and $\Gamma = \{(\sigma, q_{12}, q_{12})^{(2)}, (\sigma, q_{12}, q_{21})^{(2)}, (\sigma, q_{21}, q_{12})^{(2)}, (\sigma, q_{21}, q_{21})^{(2)}, a^{(0)}, b^{(0)}\}$, where a and b stand for (a) and (b), respectively. The set R_N consists of the rules

$$\begin{aligned} a &\to \langle q_{12}, a\rangle \\ b &\to \langle q_{21}, b\rangle \\ \sigma(\langle r, x_1\rangle, \langle r', x_2\rangle) &\to \langle r', (\sigma, r, r')(x_1, x_2)\rangle \quad \text{for all } r, r' \in Q_N. \end{aligned}$$

Consider again the input tree $t = \sigma(a, \sigma(b, \sigma(b, b)))$. Then $\tau_N(t)$ equals

$$(\sigma, q_{12}, q_{21})(a, (\sigma, q_{21}, q_{21})(b, (\sigma, q_{21}, q_{21})(b, b))).$$

We now construct the top-down tree transducer M'. Let $M' = (Q', \Gamma, \Delta', (q_0, 0), R')$ with $Q' = \{(q_0, 0)^{(0)}, (q, 0)^{(0)}, (q, 1)^{(0)}, (q, 2)^{(0)}\}$ and $\Delta' = \Sigma^{(0)} \cup \{c^{(2)}, e^{(0)}\}$ (where c is the symbol b from the proof of Lemma 13, used to make combs). For simplicity we write down the rules of M' as tree-to-string rules, i.e., we merely show the yield of the corresponding right-hand side. Let us consider in detail how to obtain the right-hand sides of the $((q, \nu), (\sigma, r, q_{21}))$-rules for $0 \leq \nu \leq 2$ and $r \in Q_N$. Since we are only interested in the yields, we have to consider the string $v = y(\mathrm{rhs}_M(q, \sigma)\Phi)$, where Φ is defined as in the proof of Lemma 13. This string equals

$$\underbrace{\langle (q,0), x_2\rangle \langle (q_0,0), x_1\rangle}_{\mathrm{part}_0(v)} y_2 \underbrace{\langle (q,1), x_2\rangle}_{\mathrm{part}_1(v)} y_1 \underbrace{\langle (q_0,0), x_1\rangle \langle (q,2), x_2\rangle}_{\mathrm{part}_2(v)}.$$

Hence, for every $r \in Q_N$ and $0 \leq \nu \leq 2$, $y\mathrm{rhs}_{M'}((q, \nu), (\sigma, r, q_{21})) = \mathrm{part}_\nu(v)$; similarly we get $y\mathrm{rhs}_{M'}((q, 0), (\sigma, r, q_{12})) = \langle (q, 0), x_2\rangle$,
$y\mathrm{rhs}_{M'}((q, 1), (\sigma, r, q_{12})) = \langle (q_0, 0), x_1\rangle \langle (q, 1), x_2\rangle \langle (q_0, 0), x_1\rangle$,
$y\mathrm{rhs}_{M'}((q, 2), (\sigma, r, q_{12})) = \langle (q, 2), x_2\rangle$.
The remaining rules are, for $0 \leq \nu \leq 2$ and $r, r' \in Q_N$,

$$\begin{aligned}
\langle (q_0, 0), (\sigma, r, r')(x_1, x_2)\rangle &\to \langle (q, 0), x_2\rangle \langle (q_0, 0), x_1\rangle \langle (q, 1), x_2\rangle \langle (q_0, 0), x_1\rangle \langle (q, 2), x_2\rangle \\
\langle (q_0, 0), a\rangle &\to a \\
\langle (q_0, 0), b\rangle &\to b \\
\langle (q, \nu), a\rangle &\to \varepsilon \\
\langle (q, \nu), b\rangle &\to \varepsilon
\end{aligned}$$

Consider the derivation by M' with input tree $t' = \tau_N(t)$ (shown above), where $t'/2 = \tau_N(\sigma(b, \sigma(b, b)))$ and $t'/22 = \tau_N(\sigma(b, b))$; again we merely show the corresponding yields.

$$\begin{aligned}
& \langle (q_0, 0), t'\rangle \\
\Rightarrow_{M'} & \langle (q, 0), t'/2\rangle \langle (q_0, 0), a\rangle \langle (q, 1), t'/2\rangle \langle (q_0, 0), a\rangle \langle (q, 2), t'/2\rangle \\
\Rightarrow^*_{M'} & \langle (q, 0), t'/22\rangle \langle (q_0, 0), b\rangle\, a\, \langle (q, 1), t'/22\rangle\, a\, \langle (q_0, 0), b\rangle \langle (q, 2), t'/22\rangle \\
\Rightarrow^*_{M'} & \langle (q, 0), b\rangle\, bba\, \langle (q, 1), b\rangle\, abb\, \langle (q, 2), b\rangle \\
\Rightarrow^*_{M'} & bbaabb.
\end{aligned}$$

Thus, indeed, $y\tau_{M'}(\tau_N(t)) = y\tau_M(t)$; see Example 10 for $\tau_M(t)$.

Let us also show how $yM'_{(q_0,0)}(t')$ can be obtained in terms of q'-translations for the states q' of M'. Since we only consider the corresponding yields, all of the following substitutions are on strings. First, $yM'_{(q_0,0)}(b) = b$ and $yM'_{(q,\nu)}(b) = \varepsilon$ for $0 \leq \nu \leq 2$. Thus,
$yM'_{(q,0)}(t'/22)$
$= y\mathrm{rhs}_{M'}((q, 0), (\sigma, q_{21}, q_{21}))[\langle (q, 0), x_2\rangle \leftarrow yM'_{(q,0)}(b), \langle (q_0, 0), x_1\rangle \leftarrow yM'_{(q_0,0)}(b)]$
$= \langle (q, 0), x_2\rangle \langle (q_0, 0), x_1\rangle[\langle (q, 0), x_2\rangle \leftarrow \varepsilon, \langle (q_0, 0), x_1\rangle \leftarrow b] = b$,

$yM'_{(q,1)}(t'/22) = y\mathrm{rhs}_{M'}((q, 1), (\sigma, q_{21}, q_{21}))[\langle (q, 1), x_2\rangle \leftarrow yM'_{(q,1)}(b)]$
$= yM'_{(q,1)}(b) = \varepsilon$, and

$yM'_{(q,2)}(t'/22)$
$= y\mathrm{rhs}_{M'}((q, 2), (\sigma, q_{21}, q_{21}))[\langle (q_0, 0), x_1\rangle \leftarrow yM'_{(q_0,0)}(b), \langle (q, 2), x_2\rangle \leftarrow yM'_{(q,2)}(b)]$
$= yM'_{(q_0,0)}(b) yM'_{(q,2)}(b) = b$.

Since, as shown in Example 10, $yM_q(\sigma(b, b)) = by_2y_1b$ and $\mathrm{part}_\nu(by_2y_1b)$ equals b,ε,b for $\nu = 0, 1, 2$, respectively, these results are in accordance with Claim 3 in the proof of Lemma 13. Next,

$yM'_{(q,0)}(t'/2)$
$= y\mathrm{rhs}_{M'}((q,0),(\sigma,q_{21},q_{21}))[\langle(q,0),x_2\rangle \leftarrow yM'_{(q,0)}(t'/22),\langle(q_0,0),x_1\rangle \leftarrow yM'_{(q_0,0)}(b)]$
$= yM'_{(q,0)}(t'/22)yM'_{(q_0,0)}(b) = bb$,

$yM'_{(q,1)}(t'/2) = y\mathrm{rhs}_{M'}((q,1),(\sigma,q_{21},q_{21})[\langle(q,1),x_2\rangle \leftarrow yM'_{(q,1)}(t'/22)]$
$= yM'_{(q,1)}(t'/22) = \varepsilon$, and

$yM'_{(q,2)}(t'/2)$
$= y\mathrm{rhs}_{M'}((q,2),(\sigma,q_{21},q_{21}))[\langle(q_0,0),x_1\rangle \leftarrow yM'_{(q_0,0)}(b),\langle(q,2),x_2\rangle \leftarrow yM'_{(q,2)}(t'/22)]$
$= yM'_{(q_0,0)}(b)yM'_{(q,2)}(t'/22) = bb$.

Again, these results are in accordance with the fact that $yM_q(\sigma(b,\sigma(b,b))) = bby_2y_1bb$. Finally, $yM'_{(q_0,0)}(t'/1) = y\mathrm{rhs}_{M'}((q_0,0),a) = a$ and $yM'_{(q_0,0)}(t')$ equals

$$yM'_{(q,0)}(t'/2)yM'_{(q_0,0)}(t'/1)yM'_{(q,1)}(t'/2)yM'_{(q_0,0)}(t'/1)yM'_{(q,2)}(t'/2) = bbaabb.$$

□

We are now ready to prove that $\mathrm{MTT_{sp}}$s and top-down tree transducers generate the same class of string languages if they take as input a class of tree languages that is closed under finite state relabelings. Note that this result can be seen as a generalization of Corollary 7.9 of [EM99], which says that finite copying MTTs generate the same class of string languages as finite copying top-down tree transducers, i.e., for a class $\mathcal{L}$ of tree languages that is closed under finite state relabelings, $yMTT_{\mathrm{fc}}(\mathcal{L}) = yT_{\mathrm{fc}}(\mathcal{L})$, where fc denotes that the corresponding transducers are finite copying.

Theorem 15. *Let $\mathcal{L}$ be a class of tree languages that is closed under finite state relabelings. Then $yMTT_{\mathrm{sp}}(\mathcal{L}) = yT(\mathcal{L})$.*

Proof. By Lemma 13, $yMTT_{\mathrm{sp}}(\mathcal{L}) \subseteq yT(\mathcal{L})$ and since every top-down tree transducer is an $\mathrm{MTT_{sp}}$, $yT(\mathcal{L}) \subseteq yMTT_{\mathrm{sp}}(\mathcal{L})$. □

By Lemma 11, we can apply Theorem 15 to $\mathcal{L}' = MTT(\mathcal{L})$, for an arbitrary class of tree languages $\mathcal{L}$. We get $yMTT_{\mathrm{sp}}(MTT(\mathcal{L})) = yT(MTT(\mathcal{L}))$ which, by Lemma 5, equals $MTT(\mathcal{L})$. Thus we obtain the following corollary which says that the class $MTT(\mathcal{L})$ is closed under translations in MTT_{sp}, with respect to yield languages.

Corollary 16. For a class $\mathcal{L}$ of tree languages, $yMTT_{\mathrm{sp}}(MTT(\mathcal{L})) = yMTT(\mathcal{L})$.

Since the class $REGT$ of regular tree languages is closed under finite state relabelings (cf. Lemma IV.6.5 of [GS84]), we get $yMTT_{\mathrm{sp}}(REGT) = yT(REGT)$ from Theorem 15. We want to make two more remarks about the class $MTT_{\mathrm{sp}}(REGT)$. First, about its yield languages: For top-down tree transducers it is known (Theorem 3.2.1 of [ERS80] and Theorem 4.3 of [Man98]) that $T(REGT)$ is equal to the class $OUT(T)$ of output tree languages of top-down tree transducers (i.e., taking the particular regular tree languages T_Σ as input). In fact, it is shown in [Man98] that for any class Ψ of tree translations which is closed under left composition with "semi-relabelings", which are particular linear top-down tree translations, $\Psi(REGT) = OUT(\Psi)$. Since it can be shown, as a special case of Lemma 4, that MTT_{sp} is closed under left composition with top-down tree translations we get that $yOUT(MTT_{\mathrm{sp}}) = yOUT(T)$, i.e., $\mathrm{MTT_{sp}}$s and top-down tree transducers generate the same class of output string languages. Second, about its path languages: If we consider $\mathrm{MTT_{sp}}$s with monadic output alphabet, then the class of path languages generated by them taking regular tree languages as input is also equal to $yT(REGT)$ (cf. the proof of Lemma 7.6 of [EM99]). Thus, the classes of path and yield languages of the class $MTT_{\mathrm{sp}}(REGT)$ are equal; this is a rare property of a class of tree languages.

5 Bridge Theorems

This section establishes the bridge theorems which are used in Sections 6 and 7 to prove that certain languages cannot be generated as output by compositions of MTTs. The basic idea is presented in Lemma 17 which gives a "bridge" from $yMTT_{\rm sp}(\mathcal{L})$ to $yMTT(\mathcal{L})$, that is, a statement of the form: if $L \notin yMTT_{\rm sp}(\mathcal{L})$ then $L' \notin yMTT(\mathcal{L})$. Using the closure properties of the previous section this will allow us to prove in Theorem 18 a bridge from $yMTT^n(\mathcal{L})$ to $yMTT^{n+1}(\mathcal{L})$, and in Theorem 20 a bridge from $yT(REGT)$ to $\bigcup_{n\geq 0} yMTT^n(REGT)$.

Let A and B be disjoint alphabets. Consider a string of the form

$$w_1a_1w_2a_2\cdots a_{l-1}w_la_lw_{l+1}$$

with $l \geq 0$, $a_1,\dots,a_l \in A$, $w_1,\dots,w_{l+1} \in B^*$, and all $w_2,\dots,w_l$ pairwise different. We call such a string a *δ-string for $a_1\cdots a_l$*. Now let $L \subseteq A^*$ and $L' \subseteq (A\cup B)^*$. If L' contains, for every $w \in L$, a δ-string for w, then L' is called *δ-complete for L*. The following theorem shows that if an MTT M generates L', then, due to the structure of δ-strings, M cannot make use of its copying facility as far as L is concerned. Recall from the Preliminaries that $\mathrm{res}_A(w_1a_1\cdots w_la_lw_{l+1}) = a_1\cdots a_l$.

Lemma 17. Let $\mathcal{L}$ be a class of tree languages which is closed under finite state relabelings and under intersection with regular tree languages. Let A, B be disjoint alphabets and let $L \subseteq A^*$ and $L' \subseteq (A\cup B)^*$ be languages such that

(1) L' is δ-complete for L and
(2) $\mathrm{res}_A(L') = L$.

If $L' \in yMTT(\mathcal{L})$ then $L \in yMTT_{\rm sp}(\mathcal{L})$.

Proof. Let $M = (Q,\Sigma,\Delta,q_0,R)$ be an MTT and $K \in \mathcal{L}$ such that $y\tau_M(K) = L'$. Obviously, we may assume that $\Delta^{(0)} = A\cup B\cup\{e\}$. Furthermore, by Lemma 9 and the closure of $\mathcal{L}$ under finite state relabelings, we may assume that M is nondeleting.

Clearly, it is sufficient to consider only δ-strings in order to generate the language L, because, by δ-completeness of L' for L, L' has a δ-string for every $w \in L$, and so $\mathrm{res}_A(\{v \in L' \mid v \text{ is a } \delta\text{-string}\}) = L$. We will construct a finite state relabeling N and an MTT$_{\rm sp}$ M' such that for every $s \in T_\Sigma$

(a) either $y\tau_{M'}(\tau_N(s)) = \mathrm{res}_A(y\tau_M(s))$ or $\tau_{M'}(\tau_N(s))$ contains a (new) dummy symbol, and
(b) if $y\tau_M(s)$ is a δ-string, then $\tau_{M'}(\tau_N(s))$ contains no dummy symbol.

We now show that this proves the lemma. Due to the closure properties of $\mathcal{L}$, the restriction of $\tau_N(K)$ to trees t such that $\tau_{M'}(t)$ contains no dummy symbol is in $\mathcal{L}$. This can be seen as follows. Since inverse macro tree translations preserve the regular tree languages (Theorem 7.4(1) of [EV85]), $R = \tau_{M'}^{-1}(T_{\Delta'-\{\rm dummy\}})$ is a regular tree language, where Δ' is the output alphabet of M'. Hence $K' = \tau_N(K)\cap R$ is in $\mathcal{L}$. Now, from (a) and (2) we get $y\tau_{M'}(K') \subseteq \mathrm{res}_A(L') = L$. By (b), $\{\tau_N(s) \mid s \in K, y\tau_M(s) \text{ is a } \delta\text{-string}\} \subseteq K'$ and thus, by (1) and (a), $L = \mathrm{res}_A(\{v \in L' \mid v \text{ is a } \delta\text{-string}\}) = \{\mathrm{res}_A(y\tau_M(s)) \mid s \in K, y\tau_M(s) \text{ is a } \delta\text{-string}\} \subseteq y\tau_{M'}(K')$. Thus, $L = y\tau_{M'}(K') \in yMTT_{\rm sp}(\mathcal{L})$.

Consider the right-hand side of a rule of M in which some parameter y_j occurs more than once. If, during the derivation of a tree which has as yield a δ-string, this rule was applied, then the tree which is substituted for y_j in this derivation contains at most one symbol in A. Because otherwise, due to copying, the resulting string would not be a δ-string. Hence, when deriving a δ-string, a rule which contains multiple occurrences of a parameter y_j is only applicable if the yield of the tree being substituted for y_j contains at most one symbol in A. Based on this fact we construct the MTT$_{\text{sp}}$ M'. The information whether the yield of the tree which will be substituted for a certain parameter contains none, one, or more than one occurrences of a symbol in A is determined by first relabeling the input tree. Then this information is kept in the states of M'. More precisely, we will define a finite state relabeling N which relabels $\sigma \in \Sigma^{(k)}$ in the tree $\sigma(s_1, \dots, s_k)$ by $(\sigma, (\phi_1, f_1), \dots, (\phi_k, f_k))$, where for every $i \in [k]$, $q \in Q^{(m)}$, and $m \geq 0$,

$$\phi_i(q) = \begin{cases} \varepsilon & \text{if } yM_q(s_i) \text{ contains no symbol in } A \\ a & \text{if } yM_q(s_i) = w_1 a w_2 \text{ with } a \in A \text{ and } w_1, w_2 \in (Y \cup B)^* \\ dd & \text{otherwise} \end{cases}$$

with d an arbitrary symbol in A, and for every $j \in [m]$,

$$(f_i(q))(j) = \begin{cases} \varepsilon & \text{if } yM_q(s_i) \text{ contains no occurrence of } y_j \\ y_j & \text{if } yM_q(s_i) \text{ contains exactly one occurrence of } y_j \\ y_j y_j & \text{otherwise.} \end{cases}$$

The case $(f_i(q))(j) = \varepsilon$ actually never occurs, because M is nondeleting and hence, by Lemma 8, y_j occurs in $yM_q(s_i)$; we only include it because it simplifies the correctness proof. Before defining N, let us define two auxiliary notions that define the above information for an arbitrary string (instead of $yM_q(s_i)$). For $w \in (A \cup B \cup Y)^*$, $\text{oc}_A(w)$ is defined as follows. If $w \in (Y \cup B)^*$, then $\text{oc}_A(w) = \varepsilon$; if $w = w_1 a w_2$ with $a \in A$ and $w_1, w_2 \in (Y \cup B)^*$, then $\text{oc}_A(w) = a$; and otherwise $\text{oc}_A(w) = dd$. Furthermore, for $j \geq 1$, $\text{oc}_j(w)$ is defined as follows. If w contains no occurrence of y_j, then $\text{oc}_j(w) = \varepsilon$; if w contains exactly one occurrence of y_j, then $\text{oc}_j(w) = y_j$; and otherwise $\text{oc}_j(w) = y_j y_j$.

Note that the existence of the relabeling N follows from the facts that for given (ϕ, f) and $q \in Q^{(m)}$ the set $\{t \in T_\Delta(Y_m) \mid \text{oc}_A(yt) = \phi(q), \text{oc}_j(yt) = (f(q))(j)$ for every $j \in [m]\}$ is regular and that inverse macro tree translations preserve the regular tree languages (Theorem 7.4(1) of [EV85]). Since part of the correctness proof of N is also needed in the correctness proof of the MTT M', we give the detailed construction of N together with a correctness proof. Note that the construction of N is similar to the constructions of the look-ahead automata A_1 and A_2 of the proofs of Lemmas 6.3 and 6.6 in [EM99], respectively; the automaton A_1 determines the precise number of occurrences of y_j in $M_q(s)$, where M is an MTT for which this number is bounded by some $B \in \mathbb{N}$, and the automaton A_2 determines whether or not y_j occurs in $M_q(s)$.

It should be clear from Definition 3 that, to define N, we have to know how oc_A and oc_j behave with respect to second-order substitution, i.e., how the oc_A and oc_j of the yield of a tree $t[\![\omega_i \leftarrow \xi_i \mid i \in [n]]\!]$ can be determined from the oc_A and oc_j's of the yields of the trees $\xi_1, \dots, \xi_n$. This is expressed in Claim 1.

Claim 1: Let Ω be a ranked alphabet such that $\Omega^{(0)} = \Delta^{(0)}$. Let $n, m \geq 1$, $\omega_1, \dots, \omega_n \in \Omega$, and $t, \xi_1, \dots, \xi_n \in T_\Omega(Y_m)$. Then for $\text{oc} \in \{\text{oc}_A, \text{oc}_1, \dots, \text{oc}_m\}$,

$$\text{oc}(y(t[\![\omega_i \leftarrow \xi_i \mid i \in [n]]\!])) = \text{oc}(y(t[\![\omega_i \leftarrow \xi_i' \mid i \in [n]]\!])),$$

where $\xi_i' = \text{comb}_b(\text{oc}_A(y\xi_i)\text{oc}_1(y\xi_i) \cdots \text{oc}_m(y\xi_i))$ for $i \in [n]$, and b is an arbitrary binary symbol.

This claim is proved by induction on the structure of t. Let $[\![\dots]\!]$ denote the substitution $[\![\omega_i \leftarrow \xi_i \mid i \in [n]]\!]$ and let $[\![\_]\!] = [\![\omega_i \leftarrow \xi_i' \mid i \in [n]]\!]$. If $t \in Y_m$, then $\text{oc}(y(t[\![\dots]\!])) = \text{oc}(yt) = \text{oc}(y(t[\![\_]\!]))$. Let $t_1, \dots, t_l \in T_\Omega(Y_m)$ and $l \geq 0$.

If $t = \delta(t_1, \dots, t_l)$ with $\delta \in \Omega^{(l)} - \{\omega_1, \dots, \omega_n\}$, then $\text{oc}(y(t[\![\dots]\!])) = \text{oc}(y(t_1[\![\dots]\!])\, y(t_2[\![\dots]\!]) \cdots y(t_l[\![\dots]\!]))$. Since $\text{oc}(uw) = \text{oc}(\text{oc}(u)\text{oc}(w))$ for $u, w \in (A \cup B \cup Y)^*$, we can apply oc to each $y(t_\nu[\![\ldots]\!])$ and get $\mathrm{oc}(\mathrm{oc}(y(t_1[\![\ldots]\!]))\cdots\mathrm{oc}(y(t_l[\![\ldots]\!])))$. By induction this equals $\mathrm{oc}(\mathrm{oc}(y(t_1[\![\_]\!]))\cdots\mathrm{oc}(y(t_l[\![\_]\!]))) = \mathrm{oc}(y(t_1[\![\_]\!])y(t_2[\![\_]\!])\cdots y(t_l[\![\_]\!])) = \mathrm{oc}(y(\delta(t_1,\ldots,t_l)[\![\_]\!]))$.

If $t=\omega_i(t_1,\ldots,t_l)$ with $i\in[n]$ and ω_i of rank l, then $\mathrm{oc}(y(t[\![\ldots]\!])) = \mathrm{oc}(y(\xi_i[y_\nu \leftarrow t_\nu[\![\ldots]\!] \mid \nu\in[l]])) = \mathrm{oc}(y(\xi_i)[y_\nu \leftarrow y(t_\nu[\![\ldots]\!]) \mid \nu\in[l]])$. Since $\mathrm{oc}(w)=\mathrm{oc}(w')$ if w' is a permutation of w, this equals $\mathrm{oc}((\mathrm{res}_{A\cup B}(y\xi_i)\mathrm{res}_{\{y_1\}}(y\xi_i)\cdots\mathrm{res}_{\{y_l\}}(y\xi_i))[y_\nu \leftarrow y(t_\nu[\![\ldots]\!]) \mid \nu\in[l]])$. Applying oc we get

$$\mathrm{oc}((\mathrm{oc}_A(y\xi_i)\mathrm{res}_{\{y_1\}}(y\xi_i)\cdots\mathrm{res}_{\{y_l\}}(y\xi_i))[\ldots])$$

with $[\ldots] = [y_\nu \leftarrow \mathrm{oc}(y(t_\nu[\![\ldots]\!])) \mid \nu\in[l]]$. This is true because for $\mathrm{oc}=\mathrm{oc}_A$, $\mathrm{oc}(\mathrm{oc}_A(y\xi_i)) = \mathrm{oc}(y\xi_i) = \mathrm{oc}(\mathrm{res}_{A\cup B}(y\xi_i))$ and for $\mathrm{oc}=\mathrm{oc}_j$, $\mathrm{oc}(\mathrm{oc}_A(y\xi_i)) = \varepsilon = \mathrm{oc}(\mathrm{res}_{A\cup B}(y\xi_i))$. Since, for $\mu\geq 0$, $\mathrm{oc}(y_\nu^\mu[\ldots]) = \mathrm{oc}(\mathrm{oc}_\nu(y_\nu^\mu)[\ldots])$, it follows that $\mathrm{oc}(\mathrm{res}_{\{y_\nu\}}(y\xi_i)[\ldots]) = \mathrm{oc}(\mathrm{oc}_\nu(y\xi_i)[\ldots])$. Hence we get

$$\mathrm{oc}((\mathrm{oc}_A(y\xi_i)\mathrm{oc}_1(y\xi_i)\cdots\mathrm{oc}_l(y\xi_i))[\ldots]) = \mathrm{oc}(y(\xi_i')[\ldots]).$$

By induction we can replace $\mathrm{oc}(y(t_\nu[\![\ldots]\!]))$ in $[\ldots]$ by $\mathrm{oc}(y(t_\nu[\![\_]\!]))$, and hence by $y(t_\nu[\![\ldots]\!])$. Thus we get $\mathrm{oc}(y(\xi_i')[y_\nu \leftarrow y(t_\nu[\![\_]\!]) \mid \nu\in[l]])$, which equals $\mathrm{oc}(y(t[\![\_]\!]))$. This concludes the proof of Claim 1.

We now construct the finite state relabeling N which adds the ϕ_i's and f_i's to the labels of the input tree. Let $N=(Q_N,\Sigma,\Gamma,R_N)$ such that

- Q_N consists of all pairs (ϕ,f), where $\phi : Q\to(\{\varepsilon,dd\}\cup A)$ and f is a function which associates with every $q\in Q^{(m)}$, $m\geq 0$, a mapping $f(q):[m]\to Y_m^*$ such that for every $j\in[m]$, $(f(q))(j)\in\{\varepsilon,y_j,y_jy_j\}$,
- $\Gamma = \{(\sigma,(\phi_1,f_1),\ldots,(\phi_k,f_k))^{(k)} \mid \sigma\in\Sigma^{(k)}, k\geq 0, (\phi_1,f_1),\ldots,(\phi_k,f_k)\in Q_N\}$, and
- R_N contains for every $(\phi_1,f_1),\ldots,(\phi_k,f_k)\in Q_N$ and $\sigma\in\Sigma^{(k)}$ with $k\geq 0$ the rule

$$\sigma(\langle(\phi_1,f_1),x_1\rangle,\ldots,\langle(\phi_k,f_k),x_k\rangle)\to \\ \langle(\phi,f),(\sigma,(\phi_1,f_1),\ldots,(\phi_k,f_k))(x_1,\ldots,x_k)\rangle,$$

where for every $q\in Q^{(m)}$ with $m\geq 0$, $\phi(q)=\mathrm{oc}_A(y\zeta)$, for every $j\in[m]$, $(f(q))(j)=\mathrm{oc}_j(y\zeta)$, and $\zeta=\mathrm{rhs}_M(q,\sigma)\Theta$. The second order substitution Θ equals (where b is an arbitrary binary symbol)

$$[\![\langle q',x_i\rangle \leftarrow \mathrm{comb}_b(\phi_i(q')(f_i(q'))(1)\cdots(f_i(q'))(l)) \mid \langle q',x_i\rangle\in\langle Q,X_k\rangle^{(l)}, l\geq 0]\!].$$

It should be clear from Claim 1 that N realizes the relabeling as described above. Formally this follows from Claim 2.

Claim 2: Let $s\in T_\Sigma$ and $(\phi,f)\in Q_N$. If $s\Rightarrow_N^*\langle(\phi,f),\tau_N(s)\rangle$ then, for every $q\in Q^{(m)}$ and $m\geq 0$,

(i) $\phi(q)=\mathrm{oc}_A(yM_q(s))$ and
(ii) for every $j\in[m]$, $(f(q))(j)=\mathrm{oc}_j(yM_q(s))$.

This claim is proved by induction on the structure of s. Let $s=\sigma(s_1,\ldots,s_k)$, $k\geq 0$, $\sigma\in\Sigma^{(k)}$, and $s_1,\ldots,s_k\in T_\Sigma$. For every $i\in[k]$ let $(\phi_i,f_i)\in Q_N$ such that $s_i\Rightarrow_N^*\langle(\phi_i,f_i),\tau_N(s_i)\rangle$. Then $s\Rightarrow_N^*\sigma(\langle(\phi_1,f_1),\tau_N(s_1)\rangle,\ldots,\langle(\phi_k,f_k),\tau_N(s_k)\rangle)\Rightarrow_N\langle(\phi,f),(\sigma,(\phi_1,f_1),\ldots,(\phi_k,f_k))(\tau_N(s_1),\ldots,\tau_N(s_k))\rangle$, where, for every $q\in Q^{(m)}$ and $m\geq 0$, $\phi(q)=\mathrm{oc}_A(y\zeta)$, for every $j\in[m]$, $(f(q))(j)=\mathrm{oc}_j(y\zeta)$, and ζ equals $\mathrm{rhs}_M(q,\sigma)\Theta$. To be able to apply Claim 1, we now take $t=\mathrm{rhs}_M(q,\sigma)$, $\{\omega_1,\ldots,\omega_n\}=\langle Q,X_k\rangle$, and for $\omega_\mu=\langle q',x_i\rangle$, $\xi_\mu=M_{q'}(s_i)$. By induction, $\phi_i(q')=$

$\mathrm{oc}_A(yM_{q'}(s_i))$ and $(f_i(q'))(j) = \mathrm{oc}_j(yM_{q'}(s_i))$ for $j \in [m]$. Thus, Θ equals the substitution $[\![\omega_i \leftarrow \mathrm{comb}_b(\mathrm{oc}_A(y\xi_i)\mathrm{oc}_1(y\xi_i)\cdots\mathrm{oc}_m(y\xi_i)) \mid i \in [n]]\!] = [\![\omega_i \leftarrow \xi_i' \mid i \in [n]]\!]$ of Claim 1. By application of Claim 1 we obtain that $\mathrm{oc}_A(y\zeta) = \mathrm{oc}_A(y(t[\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!]))$ which equals $\mathrm{oc}_A(yM_q(s))$. This proves Claim 2(i) and by replacing oc_A by oc_j it proves Claim 2(ii).

We now define the MTT M'. The idea is to keep a parameter of a state only if the yield of the tree that is substituted for it contains more than one occurrence of a symbol in A. This information is kept in the states of M' and is determined using the information provided by the relabeling N (and by the actual state of M'). If such a parameter is copied in a rule of M, then the right-hand side of the corresponding rule of M' contains a dummy symbol, because then $yM_{q_0}(s)$ is not a δ-string.

Let $M' = (Q', \Gamma, \Delta', q_0', R')$ be the MTT with

- $Q' = \{(q, \varphi) \mid q \in Q^{(m)}, m \geq 0, \varphi : [m] \to (\{\varepsilon, dd\} \cup A)\}$, where the rank of (q, φ) with $q \in Q^{(m)}$ is $|\{j \in [m] \mid \varphi(j) = dd\}|$,
- $\Delta' = (\Delta - B) \cup \{b^{(2)}, \mathrm{dummy}^{(2)}\}$, where b and dummy are symbols not in Δ,
- $q_0' = (q_0, \varnothing)$, and
- R' consisting of the following rules. For every $(q, \varphi) \in Q'^{(n)}$ and $(\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k)) \in \Gamma^{(k)}$ with $n, m, k \geq 0$ and $q \in Q^{(m)}$ let

$$\langle (q, \varphi), (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k))(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_n) \to \zeta$$

  be in R', where $\zeta = \mathrm{comb}_{\mathrm{dummy}}(y_1 \cdots y_n)$ if there is a $j \in [m]$ such that $\varphi(j) = dd$ and y_j occurs more than once in $\mathrm{rhs}_M(q, \sigma)$, and otherwise $\zeta = \mathrm{trans}(\mathrm{rhs}_M(q, \sigma))$, where for every $t \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$ the tree $\mathrm{trans}(t)$ is recursively defined as follows (depending on $\varphi, (\phi_1, f_1), \ldots, (\phi_k, f_k)$).
  For $t = y_j$ and $j \in [m]$, $\mathrm{trans}(y_j) = \mathrm{comb}_b(\varphi(j))$ if $\varphi(j) \neq dd$, and otherwise $\mathrm{trans}(y_j) = y_\nu$ with $\nu = |\{\mu \mid \mu < j \text{ and } \varphi(\mu) = dd\}| + 1$.
  For $t = \beta(t_1, \ldots, t_l)$, $\beta \in (\langle Q, X_k\rangle \cup \Delta)^{(l)}$, $l \geq 0$, and $t_1, \ldots, t_l \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$ we have:
  - If $\beta = \langle q', x_i\rangle$, then $\mathrm{trans}(t) = \langle (q', \varphi'), x_i\rangle(\mathrm{trans}(t_{j_1}), \ldots, \mathrm{trans}(t_{j_{l'}}))$, where $\{j_1, \ldots, j_{l'}\} = \varphi'^{-1}(dd)$ with $j_1 < \cdots < j_{l'}$ and for every $j \in [l]$, $\varphi'(j) = \mathrm{oc}_A(y(t_j\Theta)\Phi)$ with Θ as in the definition of N, viz.,

$$\Theta = [\![\langle q', x_i\rangle \leftarrow \mathrm{comb}_b(\phi_i(q')(f_i(q'))(1)\cdots(f_i(q'))(l)) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle^{(l)}, l \geq 0]\!],$$

    and

$$\Phi = [y_j \leftarrow \varphi(j) \mid j \in [m]].$$

  - If $\beta \in \Delta^{(l)}$ and $l \geq 1$, or $\beta \in A$, then $\mathrm{trans}(t) = \beta(\mathrm{trans}(t_1), \ldots, \mathrm{trans}(t_l))$.
  - If $\beta \in B \cup \{e\}$, then $\mathrm{trans}(t) = e$.

Let us first show that M' is sp, i.e., that each y_ν, $\nu \in [n]$, occurs exactly once in ζ. Let $\nu \in [n]$. If ζ is a dummy right-hand side then y_ν occurs exactly once in ζ. Otherwise, $\zeta = \mathrm{trans}(\mathrm{rhs}_M(q, \sigma))$ and every y_j with $\varphi(j) = dd$ occurs at most once in $\mathrm{rhs}_M(q, \sigma)$. Since $y_\nu = \mathrm{trans}(y_j)$ for some $j \in [m]$ with $\varphi(j) = dd$, this obviously implies that y_ν occurs at most once in ζ. It remains to show that y_ν occurs in ζ. This follows from the following claim for $t = \mathrm{rhs}_M(q, \sigma)$ and the fact that y_j occurs in $\mathrm{rhs}_M(q, \sigma)$ because M is nondeleting.

Claim 3: Let $t \in T_{\Delta \cup \langle Q, X_k\rangle}(Y_m)$. If y_j occurs in t, then y_ν occurs in $\mathrm{trans}(t)$.

The proof of this claim is by induction on the structure of t. The induction hypothesis is denoted by IH3. If $t = y_j$ then $\mathrm{trans}(t) = y_\nu$. Let $\eta \geq 1$ and

$t_1, \ldots, t_\eta \in T_{\Delta \cup \langle Q, X_k \rangle}(Y_m)$. If $t = \delta(t_1, \ldots, t_\eta)$ with $\delta \in \Delta^{(\eta)}$, then $\text{trans}(t) = \delta(\text{trans}(t_1), \ldots, \text{trans}(t_\eta))$. Since y_j occurs in t, it occurs in $t_{j'}$ for some $j' \in [\eta]$. By IH3, y_ν occurs in $\text{trans}(t_{j'})$ and therefore it occurs in $\text{trans}(t)$.

If $t = \langle r, x_\mu \rangle(t_1, \ldots, t_\eta)$ with $\langle r, x_\mu \rangle \in \langle Q, X_k \rangle^{(\eta)}$, then $\text{trans}(t) = \langle (r, \varphi'), x_\mu \rangle($ $\text{trans}(t_{j_1}), \ldots, \text{trans}(t_{j_{\eta'}})$, where $\{j_1, \ldots, j_{\eta'}\} = {\varphi'}^{-1}(dd)$ with $j_1 < \cdots < j_{\eta'}$ and for every $j' \in [\eta]$, $\varphi'(j') = \text{oc}_A(y(t_{j'}\Theta)\Phi)$. Let $j' \in [\eta]$ such that y_j occurs in $t_{j'}$. Then y_ν occurs in $\text{trans}(t_{j'})$ by IH3. Hence, we have to show that $\varphi'(j') = dd$, i.e., that $\text{oc}_A(y(t_{j'}\Theta)\Phi) = dd$. Each $(f_i(q'))(\bar{j})$ in the substitution Θ equals either $y_{\bar{j}}$ or $y_{\bar{j}}y_{\bar{j}}$ because, by Claim 2(ii), $(f_i(q'))(\bar{j}) = \text{oc}_{\bar{j}}(M_{q'}(s))$ for some $s \in T_\Sigma$, and, by Lemma 8 and the fact that M is nondeleting, $M_{q'}(s)$ contains $y_{\bar{j}}$. Thus, the substitution Θ is 'nondeleting', i.e., it replaces each $\langle q', x_i \rangle$, $q' \in Q^{(l)}$, by a tree that contains $y_1, \ldots, y_l$ and thus it behaves as the substitution $[\![\ldots]\!]$ in the claim of the proof of Lemma 8. Since y_j occurs in $t_{j'}$, this means that y_j also occurs in $t_{j'}\Theta$. Now Φ replaces y_j by $\varphi(j) = dd$ and thus $\text{oc}_A(y(t_{j'}\Theta)\Phi) = dd$. This concludes the proof of Claim 3.

We now formally prove properties (a) and (b) from the beginning of this proof. Let $(q, \varphi) = (q_0, \varnothing)$. Then Claim 4 proves (b), i.e., if $y\tau_M(s)$ is a δ-string, then $\tau_{M'}(\tau_N(s))$ contains no dummy symbol. Furthermore, Claim 5 proves (a), i.e., either $y\tau_{M'}(\tau_N(s)) = \text{res}_A(y\tau_M(s))$ or $\tau_{M'}(\tau_N(s))$ contains a dummy symbol.

Claim 4: Let $(q, \varphi) \in {Q'}^{(n)}$, $q \in Q^{(m)}$, $m, n \geq 0$, and $s \in T_\Sigma$. If $M'_{(q,\varphi)}(\tau_N(s))$ contains a dummy, then for all $u_1, \ldots, u_m \in T_\Delta$ with $\text{oc}_A(yu_j) = \varphi(j)$ for every $j \in [m]$, $y(M_q(s)[y_j \leftarrow u_j \mid j \in [m]])$ is not a δ-string.

The proof of this claim is by induction on the structure of s. The induction hypothesis is denoted by IH4. Let $s = \sigma(s_1, \ldots, s_k)$, $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma$. Let $(\phi_1, f_1), \ldots, (\phi_k, f_k) \in Q_N$ such that $\tau_N(s) = (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k))(\tau_N($ $s_1), \ldots, \tau_N(s_k))$. Then $M'_{(q,\varphi)}(\tau_N(s)) = \text{rhs}_{M'}((q, \varphi), (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k)))[\![\_]\!]$, where $[\![\_]\!]$ denotes the substitution $[\![\langle (q', \varphi'), x_i \rangle \leftarrow M'_{(q',\varphi')}(\tau_N(s_i)) \mid \langle (q', \varphi'), x_i \rangle \in \langle Q', X_k \rangle]\!]$. Since M' is sp, it is nondeleting and hence (similar to the claim in the proof of Lemma 8), $M'_{(q,\varphi)}(\tau_N(s))$ contains a dummy if and only if

(i) $\text{rhs}_{M'}((q, \varphi), (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k)))$ contains a dummy, or
(ii) there is an occurrence of $\langle (q', \varphi'), x_i \rangle \in \langle Q', X_k \rangle$ in the tree $\text{trans}(\text{rhs}_M(q, \sigma))$ such that $M'_{(q',\varphi')}(\tau_N(s_i))$ contains a dummy.

By the definition of the right-hand sides of M', (i) means that there is a $j \in [m]$ with $\varphi(j) = dd$ and y_j occurs more than once in $\text{rhs}_M(q, \sigma)$. Then, since M is nondeleting (cf. the claim in the proof of Lemma 8), $M_q(s) = \text{rhs}_M(q, \sigma)[\![\ldots]\!]$ has more than one occurrence of y_j, where $[\![\ldots]\!] = [\![\langle q', x_i \rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i \rangle \in \langle Q, X_k \rangle]\!]$. Thus, $y(M_q(s)[y_j \leftarrow u_j \mid j \in [m]])$ has more than one occurrence of the string yu_j. This means that it has more than one occurrence of some awa', with $a, a' \in A$ and $w \in (B \cup Y)^*$, because $\text{oc}_A(yu_j) = \varphi(j) = dd$. Hence, $y(M_q(s)[y_j \leftarrow u_j \mid j \in [m]])$ is not a δ-string.

(ii) By the definition of trans, $\text{rhs}_M(q, \sigma)$ must have a subtree $\langle q', x_i \rangle(t_1, \ldots, t_l)$ such that $\text{trans}(\langle q', x_i \rangle(t_1, \ldots, t_l))$ equals $\langle (q', \varphi'), x_i \rangle(\text{trans}(t_{j_1}), \ldots, \text{trans}(t_{j_{l'}}))$ for some $t_1, \ldots, t_l \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$, $l, l' \geq 0$ with $q' \in Q^{(l)}$, and $j_1, \ldots, j_{l'} \geq 1$. Since M is nondeleting, the tree $M_q(s)[y_j \leftarrow u_j \mid j \in [m]] = \text{rhs}_M(q, \sigma)[\![\ldots]\!][y_j \leftarrow u_j \mid j \in [m]]$ has a subtree $\xi = M_{q'}(s_i)[y_\nu \leftarrow u'_\nu \mid \nu \in [l]]$ with $u'_\nu = t_\nu[\![\ldots]\!][y_j \leftarrow u_j \mid j \in [m]]$ for $\nu \in [l]$. By the definition of trans, for $\nu \in [l]$, $\varphi'(\nu) = \text{oc}_A(y(t_\nu\Theta)\Phi)$ which equals $\text{oc}_A(yu'_\nu)$. This can be seen as follows: $\text{oc}_A(y(t_\nu\Theta)\Phi) = \text{oc}_A(y(t_\nu\Theta)[y_j \leftarrow \text{oc}_A(yu_j) \mid j \in [m]]) = \text{oc}_A(y(t_\nu\Theta)[y_j \leftarrow yu_j \mid j \in [m]]) = \text{oc}_A(y(t_\nu[y_j \leftarrow u_j \mid j \in [m]]\Theta))$. We can apply Claim 1 to this, because, by Claim 2, Θ equals the substitution $[\![\omega_i \leftarrow \xi'_i \mid i \in [n]]\!]$ in Claim 1 (with $t = t_\nu[y_j \leftarrow u_j \mid j \in [m]]$ and the ω's and ξ's chosen appropriately, as in the proof of Claim 1). We get $\text{oc}_A(y(t_\nu[y_j \leftarrow u_j \mid$

$j \in [m]]\llbracket\ldots\rrbracket)) = \mathrm{oc}_A(yu'_\nu)$. Now we can apply IH4 to (q', φ'), s_i, and $u'_1, \ldots, u'_l$ to obtain that $y\xi$ is not a δ-string. Then also $y(M_q(s)[y_j \leftarrow u_j \mid j \in [m]])$ is not a δ-string, because it has $y\xi$ as substring. This proves Claim 4.

For technical convenience we define a mapping d_B on $T_\Delta(Y)$ which realizes res_A on trees in T_Δ; for a tree $t \in T_\Delta(Y)$, $d_B(t) = t[b \leftarrow e \mid b \in B]$ and hence, for $t \in T_\Delta$, $yd_B(t) = \mathrm{res}_A(yt)$.

Claim 5: Let $(q, \varphi) \in Q'^{(n)}$, $q \in Q^{(m)}$, $m, n \geq 0$, and $s \in T_\Sigma$ such that $M'_{(q,\varphi)}(\tau_N(s))$ contains no dummy symbol. Then

$$yM'_{(q,\varphi)}(\tau_N(s)) = yd_B(M_q(s)[\varphi \neq dd][\varphi = dd]),$$

where $[\varphi \neq dd]$ denotes the substitution $[y_j \leftarrow \mathrm{comb}_b(\varphi(j)) \mid j \in [m], \varphi(j) \neq dd]$ and $[\varphi = dd]$ denotes the substitution $[y_j \leftarrow y_\nu \mid j \in [m], \varphi(j) = dd, \nu = |\{\mu \mid \mu < j \text{ and } \varphi(\mu) = dd\}| + 1]$.

This claim is proved by induction on the structure of s. The induction hypothesis is denoted by IH5. As in the proof of Claim 4, let $s = \sigma(s_1, \ldots, s_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in T_\Sigma$. Let $(\phi_1, f_1), \ldots, (\phi_k, f_k) \in Q_N$ such that $\tau_N(s) = (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k))(\tau_N(s_1), \ldots, \tau_N(s_k))$. Then $yM'_{(q,\varphi)}(\tau_N(s)) = y(\mathrm{rhs}_{M'}((q, \varphi), (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k)))\ \llbracket\langle(q', \varphi'), x_i\rangle \leftarrow M'_{(q',\varphi')}(\tau_N(s_i)) \mid \langle(q', \varphi'), x_i\rangle \in \langle Q', X_k\rangle\rrbracket)$. Since $M'_{(q,\varphi)}(\tau_N(s))$ contains no dummy symbol, neither (i) nor (ii) of the proof of Claim 4 holds, i.e., $\mathrm{rhs}_{M'}((q, \varphi), (\sigma, (\phi_1, f_1), \ldots, (\phi_k, f_k)))$ contains no dummy and hence equals $\mathrm{trans}(\mathrm{rhs}_M(q, \sigma))$, and, for every $\langle(q', \varphi'), x_i\rangle$ occurring in $\mathrm{rhs}_{M'}((q, \varphi), (\sigma, (\phi_1,\ f_1), \ldots, (\phi_k, f_k)))$, the tree $M'_{(q',\varphi')}(\tau_N(s_i))$ contains no dummy symbol. Therefore we can apply IH5 to $yM'_{(q',\varphi')}(\tau_N(s_i))$ and so, by Lemma 1(b), we can replace $M'_{(q',\varphi')}(\tau_N(s_i))$ in the second order substitution above by $d_B(M_{q'}(s_i)[\varphi' \neq dd][\varphi' = dd])$. We obtain that $yM'_{(q,\varphi)}(\tau_N(s)) = y(\mathrm{trans}(\mathrm{rhs}_M(q, \sigma))\llbracket\_\rrbracket)$, where $\llbracket\_\rrbracket$ denotes the substitution

$$\llbracket\langle(q', \varphi'), x_i\rangle \leftarrow d_B(M_{q'}(s_i)[\varphi' \neq dd][\varphi' = dd]) \mid \langle(q', \varphi'), x_i\rangle \in \langle Q', X_k\rangle\rrbracket.$$

By Claim 6 for $t = \mathrm{rhs}_M(q, \sigma)$ we get $yd_B(\mathrm{rhs}_M(q, \sigma)\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$, where $\llbracket\ldots\rrbracket = \llbracket\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle\rrbracket$. This equals $yd_B(M_q(s)[\varphi \neq dd][\varphi = dd])$ which ends the proof of Claim 5.

Claim 6: Let $m \geq 0$. For $t \in T_{\Delta \cup \langle Q, X_k\rangle}(Y_m)$,

$$y(\mathrm{trans}(t)\llbracket\_\rrbracket) = yd_B(t\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd]).$$

Claim 6 is proved by induction on the structure of t. The induction hypothesis is denoted by IH6. If $t = y_j \in Y_m$, then $y(\mathrm{trans}(y_j)\llbracket\_\rrbracket) = y\mathrm{trans}(y_j)$. By the definition of trans this is equal to $y(y_j[\varphi \neq dd][\varphi = dd])$ and thus equals $yd_B(t\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$. If $t \in B \cup \{e\}$, then $y(\mathrm{trans}(t)\llbracket\_\rrbracket) = \varepsilon = yd_B(t) = yd_B(t\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$. Let $l \geq 0$ and $t_1, \ldots, t_l \in T_{\Delta \cup \langle Q, X_k\rangle}(Y_m)$.

If $t = \delta(t_1, \ldots, t_l)$ with $\delta \in \Delta^{(l)}$, then $y(\mathrm{trans}(t)\llbracket\_\rrbracket) = y(\delta(\mathrm{trans}(t_1)\llbracket\_\rrbracket, \ldots, \mathrm{trans}(t_l)\llbracket\_\rrbracket)) = y(\mathrm{trans}(t_1)\llbracket\_\rrbracket) \cdots y(\mathrm{trans}(t_l)\llbracket\_\rrbracket)$. By IH6 we get $yd_B(t_1\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd]) \cdots yd_B(t_l\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$ which equals $yd_B(t\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$.

If $t = \langle q', x_i\rangle(t_1, \ldots, t_l)$ with $\langle q', x_i\rangle \in \langle Q, X_k\rangle^{(l)}$, then $y(\mathrm{trans}(t)\llbracket\_\rrbracket)$ equals $y(\langle(q', \varphi'), x_i\rangle(\mathrm{trans}(t_{j_1}), \ldots, \mathrm{trans}(t_{j_{l'}}))\llbracket\_\rrbracket)$, where $\varphi'(\nu) = \mathrm{oc}_A(y(t_\nu\Theta)\Phi)$ for $\nu \in [l]$, and ${\varphi'}^{-1}(dd) = \{j_1, \ldots, j_{l'}\}$ with $j_1 < \cdots < j_{l'}$. By application of $\llbracket\_\rrbracket$ we get

$$y(d_B(M_{q'}(s_i)[\varphi' \neq dd][\varphi' = dd])[y_\nu \leftarrow \mathrm{trans}(t_{j_\nu})\llbracket\_\rrbracket \mid \nu \in [l']]).$$

By IH6, $y(\mathrm{trans}(t_{j_\nu})\llbracket\_\rrbracket)$ equals $yd_B(t_{j_\nu}\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$, which means, by Lemma 1(a), that $d_B(t_{j_\nu}\llbracket\ldots\rrbracket[\varphi \neq dd][\varphi = dd])$ can be put in the substitution for

y_ν. Now this substitution can be combined with $[\varphi' = dd]$. We get

$$yd_B(M_{q'}(s_i)[\varphi' \neq dd][y_j \leftarrow t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd] \mid j \in \{j_1, \ldots, j_{l'}\}]).$$

In the substitution $[\varphi' \neq dd]$, $\varphi'(j) = \mathrm{oc}_A(y(t_j\Theta)\Phi)$ which by Claims 2 and 1 equals $\mathrm{oc}_A(y(t_j[\![\ldots]\!])\Phi)$ as in the proof of Claim 4, where for $\nu \in [m]$ we let u_ν be an arbitrary tree in T_Δ such that $\mathrm{oc}_A(yu_\nu) = \varphi(\nu)$ and hence $\mathrm{oc}_A(y(t_j\Theta)\Phi) = \mathrm{oc}_A(y(t_j\Theta[y_\nu \leftarrow u_\nu \mid \nu \in [m]])) = \mathrm{oc}_A(y(t_j[\![\ldots]\!][y_\nu \leftarrow u_\nu \mid \nu \in [m]])) = \mathrm{oc}_A(y(t_j [\![\ldots]\!])\Phi)$. Now, since $\Phi = (y[\varphi \neq dd])[dd]$ with $[dd] = [y_\nu \leftarrow dd \mid \nu \in [m], \varphi(\nu) = dd]$, we get $\varphi'(j) = \mathrm{oc}_A(y(t_j[\![\ldots]\!][\varphi \neq dd])[dd])$. This is in $\{\varepsilon\} \cup A$ and hence no y_ν with $\varphi(\nu) = dd$ appears in $y(t_j[\![\ldots]\!][\varphi \neq dd])$. Therefore the substitution $[dd]$ can be replaced by $[\varphi = dd]$. For the same reason, oc_A can be replaced by the application of d_B and y to get $yd_B(t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd])$. This means that the substitution $[\varphi' \neq dd]$ can be replaced by $[y_j \leftarrow t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd] \mid j \in ([l] - \{j_1, \ldots, j_{l'}\})]$. Altogether we get

$$\begin{aligned}
&yd_B(M_{q'}(s_i)[y_j \leftarrow t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd] \mid j \in ([l] - \{j_1, \ldots, j_{l'}\})] \\
&\qquad\qquad [y_j \leftarrow t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd] \mid j \in \{j_1, \ldots, j_{l'}\}]) \\
&= yd_B(M_{q'}(s_i)[y_j \leftarrow t_j[\![\ldots]\!][\varphi \neq dd][\varphi = dd] \mid j \in [l]]) \\
&= yd_B(t[\![\ldots]\!][\varphi \neq dd][\varphi = dd]).
\end{aligned}$$

□

Based on Lemma 17 and the closure properties of Section 4 we can now state two bridge theorems for yield languages of compositions of MTTs. Note that in the applications of Theorem 18, the language L' will often be of the form $\varphi(L)$, where φ is an operation on languages.

Theorem 18. Let A, B be disjoint alphabets and let $L \subseteq A^*$ and $L' \subseteq (A \cup B)^*$ be languages such that L' is δ-complete for L and $\mathrm{res}_A(L') = L$.

(a) For every $n \geq 1$, if $L' \in yMTT^{n+1}(REGT)$, then $L \in yMTT^n(REGT)$.
(b) If $L' \in yMTT(REGT)$, then $L \in yT(REGT)$.

Proof. (a) We want to apply Lemma 17 to L, L', and $\mathcal{L} = MTT^n(REGT)$. In order to do so, $\mathcal{L}$ must be closed (i) under intersection with $REGT$ and (ii) under finite state relabelings. To show (i), let $\tau \in MTT^n$ and $R_1, R_2 \in REGT$. Then $\tau(R_1) \cap R_2 = \tau(R_1 \cap \tau^{-1}(R_2))$. Since $REGT$ is preserved by the inverse of MTT^n, by Theorem 7.4(1) of [EV85], $\tau^{-1}(R_2) \in REGT$. Hence $R_1 \cap \tau^{-1}(R_2) \in REGT$ and $\tau(R_1 \cap \tau^{-1}(R_2)) \in MTT^n(REGT) = \mathcal{L}$. Closure property (ii) follows from Lemma 11. The application of Lemma 17 to L, L', and $\mathcal{L} = MTT^n(REGT)$ gives: if $L' \in yMTT^{n+1}(REGT)$, then L is in $yMTT_{\mathrm{sp}}(MTT^n(REGT))$, which equals $yMTT^n(REGT)$ by Corollary 16 and the fact that $n \geq 1$.

(b) Since $REGT$ is closed under intersection and under finite state relabelings (cf., e.g., [GS84]), we can apply Lemma 17 to L, L', and $\mathcal{L} = REGT$. We obtain that L is in $yMTT_{\mathrm{sp}}(REGT)$ which equals $yT(REGT)$ by Theorem 15. □

In the second bridge theorem, $L' = \varphi(L)$ for a particular operation φ on languages. Let A, B be disjoint alphabets with B nonempty, and let $L \subseteq A^*$ be a language. The function rub_B ("*rub*bish") inserts any number of symbols in B anywhere in the strings of the language to which it is applied. Hence,

$$\begin{aligned}
\mathrm{rub}_B(L) = \{&w_1a_1w_2a_2\ldots a_{l-1}w_la_lw_{l+1} \mid \\
&\qquad a_1, \ldots, a_l \in A, a_1 \cdots a_l \in L, w_1, \ldots, w_{l+1} \in B^*\}.
\end{aligned}$$

Note that $\mathrm{rub}_B(L) = \mathrm{res}_A^{-1}(L)$. Obviously $\mathrm{rub}_B(L)$ is δ-complete for L and $\mathrm{res}_A($ $\mathrm{rub}_B(L)) = L$. This means that Theorem 18 can be applied. For $B = \{b_1, \ldots, b_n\}$,

$\mathrm{rub}_B(L) = \mathrm{rub}_{\{b_n\}}(\mathrm{rub}_{\{b_{n-1}\}}(\dots\ \mathrm{rub}_{\{b_1\}}(L)\dots))$. Thus, by the n-fold application of Theorem 18 we get that if $\mathrm{rub}_{\{b_1,\dots,b_n\}}(L) \in yMTT^n(REGT)$ then $L \in yT(REGT)$.

We now show that actually two symbols $0, 1$ suffice in order to get through the whole hierarchy $yMTT^n(REGT)$ (for *any* n). The reason for this is that every symbol in an arbitrary set B can be represented by a string in $\{0,1\}^*$ in such a way that $\{0,1\}^*$ represents B^*. The translation of strings in $\{0,1\}^*$ to strings in B can be realized by an MTT M in such a way that for a tree s with $ys \in \{0,1\}^*$, $y\tau_M(s)$ is the string in B^* that corresponds to ys.

Lemma 19. Let $\mathcal{L}$ be a class of tree languages and let B be a nonempty alphabet. For a language L, if $\mathrm{rub}_{\{0,1\}}(L) \in y\mathcal{L}$, then $\mathrm{rub}_B(L) \in yMTT(\mathcal{L})$.

Proof. Let $K \in \mathcal{L}$ with $yK = \mathrm{rub}_{\{0,1\}}(L)$ and let Σ be a ranked alphabet such that $K \subseteq T_\Sigma$. By Lemma 7 there is an MTT M_Σ with output alphabet $\Gamma = \{0^{(1)}, 1^{(1)}, e^{(0)}\} \cup \{a^{(1)} \mid a \in \Sigma^{(0)}, a \neq e\}$ which translates every tree s in T_Σ into the monadic tree $\mathrm{sm}(ys) \in T_\Gamma$.

We use a Huffman code to represent each $b \in B$ by a string over $\{0,1\}$. More precisely, if $B = \{b_1, \dots, b_n\}$, then for $i \in [n]$, the string $0^{i-1}1$ represents b_i. Additionally, 0^k1 also represents b_n, for $k \geq n$. A string in $\{0,1\}^*$ can now be uniquely decoded into symbols of B, disregarding the zeros at the end. Hence, every string in $\{0,1\}^*$ represents a string in B^* and vice versa.

Let us define the top-down tree transducer M_n which translates every monadic tree $\mathrm{sm}(w_1a_1 \cdots w_la_lw_{l+1})$ with $w_1, \dots, w_l, w_{l+1} \in \{0,1\}^*$ and $a_1, \dots, a_l \in \Sigma^{(0)}$ into a tree with yield $w'_1a_1 \cdots w'_la_lw'_{l+1}$, where each $w'_i \in B^*$ is the decoded version of w_i. Let $M_n = ([n], \Gamma, \Sigma^{(0)} \cup \{\sigma^{(2)}, b_1^{(0)}, \dots, b_n^{(0)}, e^{(0)}\}, 1, R)$, where R consists of the following rules.

$$\begin{array}{lll}
\langle i, 1(x_1)\rangle & \to \sigma(b_i, \langle 1, x_1\rangle) & \text{for } i \in [n] \\
\langle i, 0(x_1)\rangle & \to \langle i+1, x_1\rangle & \text{for } i \in [n-1] \\
\langle n, 0(x_1)\rangle & \to \langle n, x_1\rangle & \\
\langle i, a(x_1)\rangle & \to \sigma(a, \langle 1, x_1\rangle) & \text{for } i \in [n] \text{ and } a \in (\Sigma^{(0)} - \{e\}) \\
\langle i, e\rangle & \to e & \text{for } i \in [n]
\end{array}$$

It should be clear that M_n realizes the translation as described, and hence $y\tau_{M_n}(\tau_{M_\Sigma}(K)) = \mathrm{rub}_B(L)$. By Lemma 5, $\tau_{M_\Sigma} \circ \tau_{M_n} \in MTT$. Thus $y\tau_{M_n}(\tau_{M_\Sigma}(K)) \in yMTT(\mathcal{L})$. □

Theorem 20. If $\mathrm{rub}_{\{0,1\}}(L) \in \bigcup_{n\geq 0} yMTT^n(REGT)$, then $L \in yT(REGT)$.

Proof. Let $n \geq 1$ and let $B = \{b_1, \dots, b_{n+1}\}$ be a set of distinct symbols which do not appear in L. By Lemma 19, if $\mathrm{rub}_{\{0,1\}}(L) \in yMTT^n(REGT)$, then $\mathrm{rub}_B(L) \in yMTT^{n+1}(REGT)$. Now we apply Theorem 18(a) to L_n and L_{n+1}, where $L_0 = L$ and for $m \geq 1$, $L_m = \mathrm{rub}_{\{b_m\}}(L_{m-1})$. We obtain that $L_{n+1} = \mathrm{rub}_B(L) \in yMTT^{n+1}(REGT)$ implies $L_n \in yMTT^n(REGT)$ and thus, by induction, that $L_1 = \mathrm{rub}_{\{b_1\}}(L) \in yMTT(REGT)$. By application of Theorem 18(b) to L and L_1 we obtain that $L \in yT(REGT)$. □

6 The yMTT-hierarchy and the EDT0L-hierarchy

In this section the bridge theorems of Section 5 are applied to prove that composition of MTTs yields a proper hierarchy of output string languages, i.e., the hierarchy $yMTT^n(REGT)$ (for short, the yMTT-hierarchy) is proper (at each level). In fact, we prove that witnesses for the properness of this hierarchy can already be found in the EDT0L-hierarchy. This will imply that also the EDT0L-hierarchy is proper. Note that from Theorem 9.10 of [Dam82] it follows that the hierarchy $MTT^n(REGT)$ of tree languages generated by compositions of MTTs is proper. Moreover, it easily follows from the proof of that theorem that the yMTT-hierarchy is infinite (because there are monadic tree languages arbitrarily high in the hierarchy $MTT^n(REGT)$).

Then we show that there are nondeterministic languages, generated by quite simple devices, which are not in the two hierarchies discussed: There is a language generated by a two-way generalized sequential machine which is not in the yMTT-hierarchy, and there is a context-free language not in the EDT0L-hierarchy.

We now move to the proof of properness of the yMTT-hierarchy. The witnesses for this properness can be generated by (controlled) EDT0L systems, which are viewed here as string transducers. Essentially, an EDT0L system is a top-down tree transducer M with monadic input alphabet (cf. [ERS80]). However, instead of a tree translation it realizes a string translation as follows: first, the input string w is turned into a monadic tree s (i.e., $s = \mathrm{sm}(w)$); then it is translated into the string $y\tau_M(s)$. The *EDT0L translation realized by* M, denoted by τ_M^{EDT0L}, is defined as $\mathrm{sm} \circ \tau_M \circ y$. Hence, the class *EDT0L* of EDT0L translations is $\mathrm{sm} \circ T \circ y$. The *EDT0L-hierarchy* consists of all $EDT0L^n(REG)$, obtained by iterating *EDT0L* on the class *REG* of regular languages. It starts with the class $EDT0L(REG)$ of EDT0L languages (because the regular control can be internalized, cf. e.g., Theorem 3.2.1 of [ERS80]).

Let us first show that the EDT0L-hierarchy is contained in the yMTT-hierarchy.

Theorem 21. For every $n \geq 1$, $EDT0L^{n+1}(REG) \subseteq yMTT^n(REGT)$.

Proof. By definition, $EDT0L^{n+1} = (\mathrm{sm} \circ T \circ y)^{n+1}$ which equals

$$\mathrm{sm} \circ (T \circ y \circ \mathrm{sm})^n \circ T \circ y.$$

By Lemma 7, $y \circ \mathrm{sm} \in MTT$. Thus, the above is included in $\mathrm{sm} \circ (T \circ MTT)^n \circ T \circ y$. By Lemma 4 this is included in $\mathrm{sm} \circ MTT^n \circ T \circ y$ which, by Lemma 5, is included in $\mathrm{sm} \circ MTT^n \circ y$. Applying this to *REG* gives $yMTT^n(\mathrm{sm}(REG)) \subseteq yMTT^n(REGT)$. □

Based on Theorem 18 we will prove that there is a language which cannot be generated as output by the composition of n MTTs, but which can be generated by the composition of $n+2$ EDT0L translations. This time the language L' in Theorem 18 will be of the form $\mathrm{count}_b(L)$. When applied to a string w, count_b inserts $b^{|w|-i}$ after the i-th symbol of the string w, for $1 \leq i < |w|$. Formally, let A be an alphabet and let $B = \{b\}$ with $b \notin A$. Define the operation $\mathrm{count}_b : A^* \to (A \cup B)^*$ as follows:

$$\mathrm{count}_b(a_1 a_2 \cdots a_l) = \prod_{i=1}^{l} a_i b^{l-i} = a_1 b^{l-1} a_2 b^{l-2} \cdots a_{l-1} b a_l.$$

Clearly, $\mathrm{count}_b(w)$ is a δ-string for w. So, for a language L, $\mathrm{count}_b(L) = \{\mathrm{count}_b(w) \mid w \in L\}$ is δ-complete for L. Since, moreover, $\mathrm{res}_A(\mathrm{count}_b(L)) = L$, we can apply Theorem 18 to L and $\mathrm{count}_b(L)$. For distinct symbols $b_1, \ldots, b_n$ we abbreviate $\mathrm{count}_{b_1} \circ \mathrm{count}_{b_2} \circ \cdots \circ \mathrm{count}_{b_n}$ by $\mathrm{count}_{b_1,\ldots,b_n}$.

To start the application of Theorem 18 we need a language L that cannot be generated by a top-down tree transducer. As shown in Theorem 3.16 of [Eng82] such a language is

$$L_{\mathrm{ec}} = \{(a^m c)^{2^m} \mid m \geq 1\},$$

where ec stands for 'exponential copying'. In fact, it is shown in that theorem that $L_{\mathrm{ec}} \notin yT(REGT)$ and even that $L_{\mathrm{ec}} \notin \bigcup_{n\geq 0} yN\text{-}T^n(REGT)$.

Theorem 22. For every $n \geq 1$, $EDT0L^{n+2}(REG) - yMTT^n(REGT) \neq \varnothing$.

Proof. Let $b_1, \ldots, b_n$ be distinct symbols not in $\{a, c\}$. We will show that the language $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}})$ is in $EDT0L^{n+2}(REG) - yMTT^n(REGT)$. That is, we will show that (1) $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}}) \in EDT0L^{n+2}(REG)$ and (2) $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}}) \notin yMTT^n(REGT)$.

(1) First, we show that $L_{\mathrm{ec}} \in EDT0L^2(REG)$ by defining two top-down tree transducers M_1 and M_2 and a regular language L such that $\tau_{M_2}^{\mathrm{EDT0L}}(\tau_{M_1}^{\mathrm{EDT0L}}(L)) = L_{\mathrm{ec}}$. Let $M_1 = (\{q^{(0)}\}, \{a^{(1)}, e^{(0)}\}, \Delta, q, R_1)$ with $\Delta = \{\sigma^{(2)}, a^{(0)}, c^{(0)}, e^{(0)}\}$ and R_1 consisting of the following rules.

$$\begin{array}{ll} \langle q, a(x_1)\rangle & \to \sigma(c, \sigma(\langle q, x_1\rangle, a)) \\ \langle q, e\rangle & \to e \end{array}$$

Then for $i \geq 0$, $\tau_{M_1}^{\mathrm{EDT0L}}(a^i) = c^i a^i$. Let $M_2 = (\{q^{(0)}\}, \Sigma, \Delta, q, R_2)$ with $\Sigma = \{a^{(1)}, c^{(1)}, e^{(0)}\}$, Δ as above, and R_2 consisting of the following rules.

$$\begin{array}{ll} \langle q, c(x_1)\rangle & \to \sigma(\langle q, x_1\rangle, \langle q, x_1\rangle) \\ \langle q, a(x_1)\rangle & \to \sigma(a, \langle q, x_1\rangle) \\ \langle q, e\rangle & \to c \end{array}$$

Then $\tau_{M_2}^{\mathrm{EDT0L}}(c^i a^j) = (a^j c)^{2^i}$ and for the regular language $L = \{a^m \mid m \geq 1\}$, $\tau_{M_2}^{\mathrm{EDT0L}}(\tau_{M_1}^{\mathrm{EDT0L}}(L)) = \tau_{M_2}^{\mathrm{EDT0L}}(\{c^m a^m \mid m \geq 1\}) = \{(a^m c)^{2^m} \mid m \geq 1\} = L_{\mathrm{ec}}$.

Now that we know that $L_{\mathrm{ec}} \in EDT0L^2(REG)$, we show that there is an EDT0L translation $\tau_{M_b}^{\mathrm{EDT0L}}$ which realizes count_b. Define $M_b = (\{q_0^{(0)}, q^{(0)}\}, \Sigma, \Delta, q_0, R)$, $\Sigma = \{a^{(1)} \mid a \in A\} \cup \{e^{(0)}\}$, $\Delta = \{a^{(0)} \mid a \in A\} \cup \{\sigma^{(2)}, e^{(0)}, b^{(0)}\}$, A is an arbitrary alphabet not containing b, and R consists of the following rules.

$$\begin{array}{lll} \langle q_0, a(x_1)\rangle & \to \sigma(a, \sigma(\langle q, x_1\rangle, \langle q_0, x_1\rangle)) & \text{for every } a \in A \\ \langle q_0, e\rangle & \to e & \\ \langle q, a(x_1)\rangle & \to \sigma(b, \langle q, x_1\rangle) & \text{for every } a \in A \\ \langle q, e\rangle & \to e & \end{array}$$

Clearly, for every $w \in A^*$, $\tau_{M_b}^{\mathrm{EDT0L}}(w) = \mathrm{count}_b(w)$. Hence, $\mathrm{count}_b \in EDT0L$ and so $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}}) \in EDT0L^{n+2}(REG)$.

(2) Application of Theorem 18(a) gives: if $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}}) \in yMTT^n(REGT)$, then $\mathrm{count}_{b_1,\ldots,b_{n-1}}(L_{\mathrm{ec}}) \in yMTT^{n-1}(REGT)$. Hence, by induction, $\mathrm{count}_{b_1}(L_{\mathrm{ec}}) \in yMTT(REGT)$ and, by Theorem 18(b), $L_{\mathrm{ec}} \in yT(REGT)$. But, as mentioned before this theorem, $L_{\mathrm{ec}} \notin yT(REGT)$ and thus $\mathrm{count}_{b_1,\ldots,b_n}(L_{\mathrm{ec}}) \notin yMTT^n(REGT)$. □

From Theorems 21 and 22 we obtain the properness of the yMTT-hierarchy.

Theorem 23. For every $n \geq 1$, $yMTT^n(REGT) \subsetneq yMTT^{n+1}(REGT)$.

As shown in the proof of Theorem 22, $L_{\mathrm{ec}} \in EDT0L^2(REG) - yT(REGT)$ and thus $L_{\mathrm{ec}} \in EDT0L^2(REG) - EDT0L(REG)$, because $EDT0L(REG) = yT(\mathrm{sm}(REG)) \subseteq yT(REGT)$. Hence, by Theorems 21 and 22, the EDT0L-hierarchy is proper. This was mentioned as an open problem after Theorem 4.3 of [Eng82].

Theorem 24. For every $n \geq 1$, $EDT0L^n(REG) \subsetneq EDT0L^{n+1}(REG)$.

Nondeterministic Languages not in the yMTT- and EDT0L-hierarchies

Here we show that particular "nondeterministic" languages are not in the yMTT- and EDT0L-hierarchies. First, a language generated by a nondeterministic two-way generalized sequential machine (2GSM) is considered and it is proved that this language is not in the yMTT-hierarchy. Second, a context-free language is considered and proved not to be in the EDT0L-hierarchy.

A 2GSM is a nondeterministic finite-state device that takes as input a string (surrounded by end markers) on which it can move back and forth, possibly changing its state and generating output. Let *2GSM* denote the class of string-to-string translations realized by 2GSMs.

Theorem 25. $2GSM(REG) - \bigcup_{n\geq 0} yMTT^n(REGT) \neq \varnothing$.

Proof. Let $0, 1, b$, and a be distinct symbols and let $L = \mathrm{rub}_{\{0,1\}}(\mathrm{rub}_{\{b\}}(L_{\mathrm{np}}))$ with $L_{\mathrm{np}} = \{a^n \mid n \text{ is not a prime}\}$. Then $L \in 2GSM(REG) - \bigcup_{n\geq 0} yMTT^n(REGT)$, i.e., (1) $L \in 2GSM(REG)$ and (2) $L \notin \bigcup_{n\geq 0} yMTT^n(REGT)$.

(1) It is straightforward to show that there is a 2GSM M and a regular language R such that M's translation applied to R gives L. The language R consists of all strings a^p, $p \geq 2$. Now M traverses q times, with $q \geq 2$, the input string a^p, outputting an a at each move. Moreover, at every step M can nondeterministically choose not to move and to output a symbol in $\{0, 1, b\}$. Hence, M generates all strings in $\mathrm{rub}_{\{0,1,b\}}(\{a^{p\cdot q} \mid p, q \geq 2\}) = \mathrm{rub}_{\{0,1\}}(\mathrm{rub}_{\{b\}}(L_{\mathrm{np}})) = L$.

(2) By Theorem 20, $\mathrm{rub}_{\{0,1\}}(\mathrm{rub}_{\{b\}}(L_{\mathrm{np}})) \in \bigcup_{n\geq 0} yMTT^n(REGT)$ implies that $\mathrm{rub}_{\{b\}}(L_{\mathrm{np}}) \in yT(REGT)$. By Theorem 3.2.14 of [ERS80] (which is another bridge theorem, closely related to Lemma 17), $\mathrm{rub}_{\{b\}}(L_{\mathrm{np}}) \in yT(REGT)$ implies that $L_{\mathrm{np}} \in yT_{\mathrm{fc}}(REGT)$, where T_{fc} denotes the class of translations realized by top-down tree transducers that are finite copying. It is known that the language L_{np} is not in $yT_{\mathrm{fc}}(REGT)$, because it is not regular and hence its Parikh-set is not semi-linear (cf. Corollary 3.2.7 of [ERS80]; cf. also the proof of Theorem 4.8 of [Eng82]). Thus $L \notin \bigcup_{n\geq 0} yMTT^n(REGT)$. □

Since the class $2GSM(REG)$ is included in the class of ET0L languages (this follows, e.g., from the characterization of ET0L languages by checking-stack pushdown automata [vL76], which can easily simulate 2GSMs; see also [ERS80]), Theorem 25 implies that $ET0L(REG) - \bigcup_{n\geq 0} yMTT^n(REGT) \neq \varnothing$, i.e., there is an ET0L language that is not in the yMTT-hierarchy. Denote by $N\text{-}T$ the class of translations realized by nondeterministic top-down tree transducers. Then, analogous to the deterministic case, $ET0L = \mathrm{sm} \circ N\text{-}T \circ y$ and thus $ET0L(REG) \subseteq yN\text{-}T(REGT)$. Hence, $yN\text{-}T(REGT) - \bigcup_{n\geq 0} yMTT^n(REGT) \neq \varnothing$ by Theorem 25.

Finally, we show that there is a context-free language (i.e., a language in $yREGT$) which is not in the EDT0L-hierarchy. This strengthens the well-known result that there are context-free languages which cannot be generated by EDT0L systems, i.e., which are not in $EDT0L(REG)$ (cf., e.g., Corollary 3.2.18(i) of [ERS80]).

Let $REGT_{\mathrm{mon}}$ denote the restriction of $REGT$ to monadic trees. We prove that there is a language in the class CF of context-free languages, which is not in the hierarchy $yMTT^n(REGT_{\mathrm{mon}})$. Since this hierarchy includes the EDT0L-hierarchy by the proof of Theorem 21 (because $\mathrm{sm}(REGT) \subseteq REGT_{\mathrm{mon}}$), the above mentioned result follows as a corollary.

Theorem 26. $CF - \bigcup_{n\geq 0} yMTT^n(REGT_{\mathrm{mon}}) \neq \varnothing$.

Proof. Let $L \in CF - EDT0L(REG)$. Obviously, $\mathcal{L} = REGT_{\mathrm{mon}}$ satisfies the closure properties of Lemma 17 (because $REGT$ does). This implies that Theorems 18 and 20 can also be stated with $REGT$ replaced by $REGT_{\mathrm{mon}}$. Then, by Theorem 20, if $\mathrm{rub}_{\{0,1\}}(L) \in \bigcup_{n\geq 0} yMTT^n(REGT_{\mathrm{mon}})$, then $L \in yT(REGT_{\mathrm{mon}})$. Clearly, this means that $L \in yT(\mathrm{sm}(REG)) = EDT0L(REG)$, because a top-down tree transducer with monadic input trees, i.e., trees of the form $a_1(\cdots a_{n-1}(a_n)\cdots)$, can easily be changed into one with input trees of the form $\mathrm{sm}(a_1 \cdots a_n)$ that generates the same output: the input symbols of rank zero are changed to have rank one, the right-hand sides of all rules are taken over, and for the input symbol e an arbitrary rule is added (which will not be used). Since $L \notin EDT0L(REG)$, this means that $\mathrm{rub}_{\{0,1\}}(L)$ is not in $\bigcup_{n\geq 0} yMTT^n(REGT_{\mathrm{mon}})$. Clearly, $\mathrm{rub}_{\{0,1\}}(L) \in CF$, because the context-free languages are closed under substitution (see, e.g., Theorem 6.2 of [HU79]). $\square$

Corollary 27. $CF - \bigcup_{n\geq 0} EDT0L^n(REG) \neq \varnothing$.

7 The IO-hierarchy

In this section we investigate the relationship between the IO-hierarchy and both the yMTT-hierarchy and the EDT0L-hierarchy. By Theorem 7.5 of [ES78], the *IO-hierarchy* can be defined in terms of tree translations as follows:

$$\text{for } n \geq 1,\ IO(n) = yYIELD^n(REGT),$$

where *YIELD* is the class of YIELD mappings defined below. The hierarchy starts with the class $IO(1)$ of languages generated by the IO macro grammars of [Fis68]. Since $YIELD \subseteq MTT$ by Theorem 4.6 of [EV85], $IO(n) \subseteq yMTT^n(REGT)$, i.e., the IO-hierarchy is inside the yMTT-hierarchy. In fact, the yMTT-hierarchy differs from the IO-hierarchy only by a single application of a top-down tree transducer, because $yMTT^n(REGT) = yYIELD^n(T(REGT))$ by Corollary 4.13 of [EV85]. It is shown in [Dam82] that the IO-hierarchy is infinite, and that the IO-hierarchy of tree languages $YIELD^n(REGT)$ is proper.

A YIELD mapping Y_f is a mapping from T_Σ to $T_\Delta(Y)$ defined by a mapping f from $\Sigma^{(0)}$ to $T_\Delta(Y)$, for ranked alphabets Σ and Δ. It realizes the semantics of first-order tree substitution in the following way.

(i) for $\alpha \in \Sigma^{(0)}$, $Y_f(\alpha) = f(\alpha)$ and
(ii) for $\sigma \in \Sigma^{k+1}$, $s_0, s_1, \ldots, s_k \in T_\Sigma$, and $k \geq 0$,
$Y_f(\sigma(s_0, s_1, \ldots, s_k)) = Y_f(s_0)[y_i \leftarrow Y_f(s_i) \mid i \in [k]]$.

Example 28. Consider the tree language L_{cf} consisting of monadic trees of the form $c^m(a^m(e))$, $m \geq 1$. We want to show that L_{cf} is in $YIELD(REGT)$, i.e., that there is a regular tree language K and a mapping f such that $Y_f(K) = L_{\mathrm{cf}}$. The regular tree language K consists of binary trees with yields of the form $\gamma^m\alpha^m$ and is generated by the regular tree grammar with productions $S \to \sigma(A, e)$, $A \to \sigma(\gamma, \sigma(A, \alpha))$, and $A \to \sigma(\gamma, \alpha)$. Now the YIELD mapping Y_f simply has to generate yK, as monadic trees. Let $f(\alpha) = a(y_1)$, $f(\gamma) = c(y_1)$, and $f(e) = e$. Consider, e.g., the tree $s = \sigma(\sigma(\gamma, \alpha), e) \in K$. Then $Y_f(s) = Y_f(\sigma(\gamma, \alpha))[y_1 \leftarrow f(e)] = f(\gamma)[y_1 \leftarrow f(\alpha)][y_1 \leftarrow e] = c(y_1)[y_1 \leftarrow a(y_1)][y_1 \leftarrow e] = c(a(e))$. It should be clear that $Y_f(K) = L_{\mathrm{cf}}$. $\square$

7.1 Comparison with the yMTT-hierarchy

Now we compare the IO-hierarchy with the yMTT-hierarchy and prove (in Theorem 32) that $IO(n+1) - yMTT^n(REGT) \neq \varnothing$. Let us first show that YIELD mappings are closed under composition with tree homomorphisms (= second-order tree substitutions).

Lemma 29. Let Y_f be a YIELD mapping from T_Σ to $T_\Delta(Y)$ and M a tree homomorphism with input alphabet Δ. There is a YIELD mapping Y_g such that for every tree $s \in T_\Sigma$, if $Y_f(s)$ contains no parameters, then $Y_g(s) = \tau_M(Y_f(s))$.

Proof. Let f be a mapping from $\Sigma^{(0)}$ to $T_\Delta(Y)$, and $M = (\{q^{(0)}\}, \Delta, \Gamma, q, R)$. The idea is to define $g(\alpha)$ for $\alpha \in \Sigma^{(0)}$ by running M on $f(\alpha)$, leaving parameters unchanged. That is, if $\widehat{M}$ is the extension of M to input trees in $T_\Delta(Y_m)$ (for some m large enough) by rules $\langle q, y_j\rangle \to y_j$, then define the new mapping g by $g(\alpha) = \tau_{\widehat{M}}(f(\alpha))$. If $Y_f(s) \in T_\Delta$, then $\tau_M(Y_f(s)) = \tau_{\widehat{M}}(Y_f(s))$, which equals $Y_g(s)$ by the following claim.

<u>*Claim:*</u> For every $s \in T_\Sigma$, $Y_g(s) = \tau_{\widehat{M}}(Y_f(s))$.

The proof of this claim is by induction on the structure of s. Let $[\![\widehat{M}]\!]$ be the second-order substitution $[\![\sigma \leftarrow \zeta_\sigma \mid \sigma \in \Sigma]\!]$ with $\zeta_\sigma = \mathrm{rhs}_M(q,\sigma)[\langle q, x_i\rangle \leftarrow y_i \mid i \in [k]]$ for every $\sigma \in \Sigma$. Then, clearly, $t[\![\widehat{M}]\!] = \tau_{\widehat{M}}(t)$ for every $t \in T_\Delta(Y)$. If $s = \alpha \in \Sigma^{(0)}$, then $Y_g(s) = g(\alpha) = \tau_{\widehat{M}}(f(\alpha)) = \tau_{\widehat{M}}(Y_f(s))$. Let $s = \sigma(s_0, \ldots, s_k)$, $k \geq 1$, $\sigma \in \Sigma^{(k+1)}$, and $s_0, \ldots, s_k \in T_\Sigma$. Then $\tau_{\widehat{M}}(Y_f(s)) = Y_f(s)[\![\widehat{M}]\!] = Y_f(s_0)[y_i \leftarrow Y_f(s_i) \mid i \in [k]][\![\widehat{M}]\!]$. This equals $Y_f(s_0)[\![\widehat{M}]\!][y_i \leftarrow Y_f(s_i)[\![\widehat{M}]\!] \mid i \in [k]] = \tau_{\widehat{M}}(Y_f(s_0))[y_i \leftarrow \tau_{\widehat{M}}(Y_f(s_i)) \mid i \in [k]]$. By induction this is equal to $Y_g(s_0)[y_i \leftarrow Y_g(s_i) \mid i \in [k]] = Y_g(s)$. □

Example 30. Let M be the top-down tree transducer M_2 defined in the proof of Theorem 22 and let f be the mapping of Example 28. Since M is a tree homomorphism, we can apply the construction of the proof of Lemma 29. Define $g(\alpha) = \tau_{\widehat{M}}(f(\alpha)) = \tau_{\widehat{M}}(a(y_1)) = \sigma(a, y_1)$, $g(\gamma) = \tau_{\widehat{M}}(c(y_1)) = \sigma(y_1, y_1)$, and $g(e) = \tau_{\widehat{M}}(e) = c$.

Clearly, $Y_g(s) = \tau_M(Y_f(s))$ for every s. This means that for the regular tree language K of Example 28, $yY_g(K) = y\tau_M(Y_f(K)) = y\tau_M(L_{\mathrm{cf}}) = \{(a^m c)^{2^m} \mid m \geq 1\}$ which is the language L_{ec} defined before Theorem 22. Hence, L_{ec} is in $IO(1)$, i.e., it is a (well-known) example of an IO macro language. □

Now that we know that $L_{\mathrm{ec}} \in IO(1)$, we want to find an operation φ that can be realized by a YIELD mapping and which is defined in such a way that Theorem 18 can be applied to $L' = \varphi(L)$ for a language L. Unlike the operations rub and count of before, the operation we use now is a tree translation, i.e., $L' = y\varphi(K)$, where $yK = L$.

Let $\Sigma = \{\sigma^{(2)}, \mathrm{root}^{(1)}\} \cup \Sigma^{(0)}$ be a ranked alphabet and let l, r be symbols not in Σ. Recall from Section 2.1 that each node ρ of a tree s is denoted by a string in $\mathbb{N}^*$, and that the label of s at ρ is denoted by $s[\rho]$. Consider a tree translation τ from T_Σ to T_Δ with $\Delta = \Sigma \cup \{l^{(0)}, r^{(0)}, e^{(0)}\}$. Then τ is an *(l,r)-leaf insertion for* Σ, if, for every $s' = \mathrm{root}(s)$ and $s \in T_{\Sigma - \{\mathrm{root}\}}$,

(i) $\tau(s') = \mathrm{root}(t)$ for some $t \in T_{\Delta - \{\mathrm{root}\}}$ and
(ii) $y\tau(s') = \rho'_1 s[\rho_1]\rho'_2 s[\rho_2] \cdots \rho'_m s[\rho_m]$, where $\rho'_i = \rho_i[1 \leftarrow l, 2 \leftarrow r]$ and $\rho_1, \ldots, \rho_m \in \{1,2\}^*$ are all leaves of s in pre-order that are not labeled by e.

As an example, let $\Sigma^{(0)} = \{a, b, e\}$ and consider the tree $s = \sigma(a, \sigma(\sigma(e, b), a))$. Figure 3 shows $s' = \mathrm{root}(s)$ and the tree $\tau(s')$ for an (l,r)-leaf insertion τ (obviously, $y\tau(s') = larlrbrra$ is a δ-string for $ys' = aba$).

Let τ be an (l,r)-leaf insertion for Σ and let $A = \Sigma^{(0)} - \{e\}$ and $B = \{l, r\}$. It should be clear that, for a "rooted" tree language $K \subseteq \mathrm{root}(T_{\Sigma - \{\mathrm{root}\}})$, the language $L' = y\tau(K)$ is δ-complete for $L = yK$. Moreover, $\mathrm{res}_A(L') = L$ because $\mathrm{res}_A(y\tau(s')) = s[\rho_1]s[\rho_2] \cdots s[\rho_m] = ys$. This means that Theorem 18 can be applied to L and L'. Rather than defining an (l,r)-leaf insertion in *YIELD*, it suffices, due to Lemma 29, to show that there is an (l,r)-leaf insertion τ in $\tau_M \circ$ *YIELD* for some tree homomorphism M. This is true because τ will always be applied to a

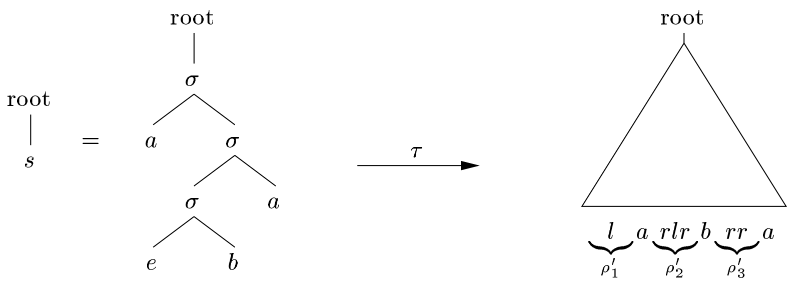

Fig. 3. The trees s' and $\tau(s')$ for an (l,r)-leaf insertion τ

tree language K in $\mathit{YIELD}(\mathcal{L})$ for some class $\mathcal{L}$ of tree languages, i.e., to a K of the form $Y_f(K')$ for some YIELD mapping Y_f and tree language $K' \in \mathcal{L}$. Hence, by Lemma 29, $\tau(K) \in \mathit{YIELD}(\tau_M(Y_f(K'))) \subseteq \mathit{YIELD}^2(\mathcal{L})$.

Lemma 31. Let $\Sigma = \{\sigma^{(2)}, \mathrm{root}^{(1)}\} \cup \Sigma^{(0)}$ be a ranked alphabet and let l, r be symbols not in Σ. There is a tree homomorphism M and a YIELD mapping Y_f such that $\tau_M \circ Y_f$ is an (l,r)-leaf insertion for Σ.

Proof. Define $M = (\{q^{(0)}\}, \Sigma, \Gamma, q, R)$ with $\Gamma = \{\delta^{(3)}, l^{(0)}, r^{(0)}, c^{(0)}, d^{(0)}, e^{(0)}\} \cup \Sigma$ and R consisting of the following rules.

$$\begin{array}{lll} \langle q, \mathrm{root}(x_1)\rangle & \to \sigma(d, \sigma(\langle q, x_1\rangle, e)) & \\ \langle q, \sigma(x_1, x_2)\rangle & \to \delta(c, \sigma(\langle q, x_1\rangle, l), \sigma(\langle q, x_2\rangle, r)) & \\ \langle q, a\rangle & \to a & \text{for every } a \in \Sigma^{(0)} \end{array}$$

The mapping f is defined as $f(d) = \mathrm{root}(y_1)$, $f(c) = \sigma(y_1, y_2)$, $f(e) = e$, and, for every $a \in \Sigma^{(0)} \cup \{l, r\}$ with $a \neq e$, $f(a) = \sigma(y_1, a)$.

Let us now prove that $\tau_M \circ Y_f$ is an (l,r)-leaf insertion. For $s \in T_{\Sigma - \{\mathrm{root}\}}$, $Y_f(\tau_M(\mathrm{root}(s))) = Y_f(\sigma(d, \sigma(\tau_M(s), e)))$ by the definition of M. This equals

$$\begin{aligned} f(d)[y_1 \leftarrow Y_f(\sigma(\tau_M(s), e))] &= \mathrm{root}(y_1)[y_1 \leftarrow Y_f(\tau_M(s))[y_1 \leftarrow Y_f(e)]] \\ &= \mathrm{root}(Y_f(\tau_M(s))[y_1 \leftarrow e]). \end{aligned}$$

By the rules of M, $\tau_M(s)$ does not contain occurrences of the symbol d, and thus $Y_f(\tau_M(s))[y_1 \leftarrow e] \in T_{\Delta - \{\mathrm{root}\}}$ with $\Delta = \Sigma \cup \{l^{(0)}, r^{(0)}, e^{(0)}\}$. This proves part (i) of the definition of (l,r)-leaf insertion.

The yield of $\mathrm{root}(Y_f(\tau_M(s))[y_1 \leftarrow e])$ is equal to $yY_f(\tau_M(s))[y_1 \leftarrow \varepsilon]$ which equals $\rho'_1 s[\rho_1]\rho'_2 s[\rho_2] \cdots \rho'_m s[\rho_m]$ by the following claim (with the ρ'_i as in the claim). This proves part (ii) of the definition of (l,r)-leaf insertion.

Claim: For every $s \in T_{\Sigma - \{\mathrm{root}\}}$, $yY_f(\tau_M(s)) = y_1\rho'_1 a_1 y_1 \rho'_2 a_2 \cdots y_1 \rho'_m a_m$, where $m \geq 0$, $ys = a_1 \cdots a_m$, $a_i \in \Sigma^{(0)} - \{e\}$, $\rho'_i = \rho_i[1 \leftarrow l, 2 \leftarrow r]$ for $i \in [m]$, and $\rho_1, \ldots, \rho_m$ are all leaves of s in pre-order that are not labeled e.

The claim is proved by induction on the structure of s. If $s = a \in \Sigma^{(0)} - \{e\}$, then $yY_f(\tau_M(s)) = yY_f(a) = y\sigma(y_1, a) = y_1\rho'_1 a_1$, where $ys = a_1 = a$, $\rho_1 = \varepsilon$, and $\rho'_1 = \rho_1[1 \leftarrow l, 2 \leftarrow r] = \varepsilon$. If $s = e$, then $yY_f(\tau_M(s)) = yY_f(e) = ye = \varepsilon$ (which proves the statement for $m = 0$). If $s = \sigma(s_1, s_2)$ with $s_1, s_2 \in T_{\Sigma - \{\mathrm{root}\}}$, then

$$\begin{aligned} Y_f(\tau_M(s)) &= Y_f(\delta(c, \sigma(\tau_M(s_1), l), \sigma(\tau_M(s_2), r)) \\ &= Y_f(c)[y_1 \leftarrow Y_f(\sigma(\tau_M(s_1), l)), y_2 \leftarrow Y_f(\sigma(\tau_M(s_2), r))] \\ &= \sigma(y_1, y_2)[y_1 \leftarrow Y_f(\tau_M(s_1))[y_1 \leftarrow \sigma(y_1, l)], \\ &\qquad\qquad y_2 \leftarrow Y_f(\tau_M(s_2))[y_1 \leftarrow \sigma(y_1, r)]]. \end{aligned}$$

The yield of this tree is $yY_f(\tau_M(s_1))[y_1 \leftarrow y_1 l]yY_f(\tau_M(s_2))[y_1 \leftarrow y_1 r]$. By induction, $yY_f(\tau_M(s_1)) = y_1 p'_1 b_1 \cdots y_1 p'_i b_i$ and $yY_f(\tau_M(s_2)) = y_1 q'_1 c_1 \cdots y_1 q'_j c_j$ with $ys_1 =$

$b_1 \cdots b_i$, $i \geq 0$, $ys_2 = c_1 \cdots c_i$, $j \geq 0$, $p'_\nu = p_\nu[1 \leftarrow l, 2 \leftarrow r]$ for $\nu \in [i]$, $q'_\mu = q_\mu[1 \leftarrow l, 2 \leftarrow r]$ for $\mu \in [j]$, and $p_1, \ldots, p_i$ and $q_1, \ldots, q_j$ are all leaves in pre-order not labeled by e of s_1 and s_2, respectively. Thus, $yY_f(\tau_M(s)) = y_1 l p'_1 b_1 \cdots y_1 l p'_i b_i y_1 r q'_1 c_1 \cdots y_1 r q'_j c_j$ which equals $y_1 \rho'_1 a_1 \cdots y_1 \rho'_m a_m$, where: $m = i + j$, for $\nu \in [i]$, $\rho'_\nu = (1p_\nu)[1 \leftarrow l, 2 \leftarrow r]$ and $a_\nu = b_\nu$, and, for $\mu \in [j]$, $\rho'_{i+\mu} = (2q_\mu)[1 \leftarrow 1, 2 \leftarrow r]$ and $a_{i+\mu} = c_\mu$. This proves the claim, because $a_1 \cdots a_m = b_1 \cdots b_i c_1 \cdots c_j = ys_1 ys_2 = ys$ and $1p_1, \ldots, 1p_i, 2q_1 \ldots, 2q_j$ are all leaves of s in pre-order that are not labeled by e. □

We now prove that witnesses for the properness of the yMTT-hierarchy can already be found in the IO-hierarchy.

Theorem 32. For every $n \geq 1$, $IO(n+1) - yMTT^n(REGT) \neq \varnothing$.

Proof. Let $n \geq 1$. We first define the language L_n in $IO(n+1) - yMTT^n(REGT)$. Let Σ be the ranked alphabet $\{\sigma^{(2)}, \mathrm{root}^{(1)}\} \cup \Sigma^{(0)}$ with $\Sigma^{(0)} = \{a, c, e\}$ and let K be the regular tree language defined in Example 28. Define the regular tree language K_{ec} as $\sigma(d, K) = \{\sigma(d, s) \mid s \in K\}$ and let g be the mapping as defined in Example 30, extended by $g(d) = \mathrm{root}(y_1)$. Then $Y_g(K_{\mathrm{ec}}) = \mathrm{root}(Y_g(K)) \subseteq T_\Sigma$, $yY_g(K_{\mathrm{ec}}) = L_{\mathrm{ec}}$, and every tree in $Y_g(K_{\mathrm{ec}})$ is "rooted", i.e., of the form $\mathrm{root}(s)$, $s \in T_{\Sigma - \{\mathrm{root}\}}$. Let $l_1, \ldots, l_n, r_1, \ldots, r_n$ be distinct symbols of rank zero, not in Σ. By Lemma 31 there is, for every $i \in [n]$, a tree homomorphism M_i and a YIELD mapping Y_{f_i} such that $\tau_i = \tau_{M_i} \circ Y_{f_i}$ is an (l_i, r_i)-leaf insertion for $\Sigma \cup \{l_1, \ldots, l_{i-1}, r_1, \ldots, r_{i-1}\}$. For $n \geq 0$ define

$$K_n = (Y_g \circ \tau_{M_1} \circ Y_{f_1} \circ \tau_{M_2} \circ Y_{f_2} \circ \cdots \circ \tau_{M_n} \circ Y_{f_n})(K_{\mathrm{ec}}).$$

Then $L_n = yK_n \in IO(n+1) - yMTT^n(REGT)$, i.e., (1) $L_n \in IO(n+1)$ and (2) $L_n \notin yMTT^n(REGT)$.

(1) Since $Y_g(K_{\mathrm{ec}}) \subseteq T_\Sigma$, $Y_g(K_{\mathrm{ec}})$ contains no parameters. Thus, by Lemma 29, there is a YIELD mapping $Y_{g'}$ such that $Y_{g'}(K_{\mathrm{ec}}) = \tau_{M_1}(Y_g(K_{\mathrm{ec}}))$. Let $i \in [n-1]$. Since $\tau_{M_i} \circ Y_{f_i}$ is a leaf insertion and every tree in $Y_g(K_{\mathrm{ec}})$ is rooted, $Y_{f_i}(s)$ has no parameters for $s \in \tau_{M_i}(Y_{f_{i-1}}(\tau_{M_{i-1}}(\ldots \tau_{M_1}(Y_g(K_{\mathrm{ec}}))\ldots)))$; by Lemma 29 there is a YIELD mapping f'_i such that $Y_{f'_i}(s) = \tau_{M_{i+1}}(Y_{f_i}(s))$. Altogether, there are YIELD mappings $f'_1, \ldots, f'_{n-1}$ such that

$$K_n = (Y_{g'} \circ Y_{f'_1} \circ Y_{f'_2} \circ \cdots \circ Y_{f'_{n-1}} \circ Y_{f_n})(K_{\mathrm{ec}})$$

which is in $(\mathit{YIELD} \circ \mathit{YIELD}^{n-1} \circ \mathit{YIELD})(REGT)$ and thus $L_n = yK_n \in IO(n+1)$.

(2) As discussed before Lemma 31, Theorem 18 can be applied to $L = L_{n-1} = yK_{n-1}$ and $L' = L_n = y\tau_n(K_{n-1})$, for the rooted tree language K_{n-1} and the (l_n, r_n)-leaf insertion $\tau_n = \tau_{M_n} \circ Y_{f_n}$. By Theorem 18(a), if $L_n \in yMTT^n(REGT)$ then $L_{n-1} \in yMTT^{n-1}(REGT)$, and by induction, $L_1 = y\tau_1(Y_g(K_{\mathrm{ec}})) \in yMTT(REGT)$; by Theorem 18(b) this means that $L_0 = yY_g(K_{\mathrm{ec}}) = L_{\mathrm{ec}} \in yT(REGT)$ which contradicts the fact that $L_{\mathrm{ec}} \notin yT(REGT)$ as stated before Theorem 22. □

From Theorem 32 and the fact that $IO(n) \subseteq yMTT^n(REGT)$, as discussed at the beginning of this section, we obtain the properness of the IO-hierarchy.

Theorem 33. For every $n \geq 1$, $IO(n) \subsetneq IO(n+1)$.

7.2 Comparison with the EDT0L-hierarchy

Let us now turn to the comparison of the IO-hierarchy and the EDT0L-hierarchy. For ET0L systems, it was proved in [Vog88] that the ET0L-hierarchy is included in the OI-hierarchy $OI(n)$, generated by the n-level OI macro grammars (see, e.g., [ES78,Dam82]). We prove that a similar result holds for EDT0L systems: the EDT0L-hierarchy is included in the IO-hierarchy. Note that the IO-hierarchy and the OI-hierarchy are both generated by n-level grammars, but in a different mode of derivation (inside-out and outside-in, respectively). The hierarchies seem, however, to be incomparable; in one direction this follows from (the discussion following) Theorem 25: there is an ET0L language not in the IO-hierarchy.

The proof of the inclusion of the EDT0L-hierarchy in the IO-hierarchy is based on the following lemma which shows how to simulate a top-down tree transducer with monadic input trees by a YIELD mapping (applied to a regular tree language). This is basically the technique used in [Dow74] to prove that $\mathit{EDT0L}(\mathit{REG}) \subseteq \mathit{IO}(1)$, cf. Theorem 6.3 of [ERS80].

A YIELD mapping evaluates a tree in a bottom-up fashion. This means that, in order to simulate a top-down tree transducer with monadic input $\mathrm{sm}(u)$, the string u has to be reversed first.

Lemma 34. $T(\mathrm{sm}(\mathit{REG})) \subseteq \mathit{YIELD}(\mathit{REGT})$.

Proof. Let $L \subseteq A^*$ be a regular language for some alphabet A. Let $M = (\{q_1, \ldots, q_m\}, \Sigma, \Delta, q_1, R)$ be a top-down tree transducer with $m \geq 1$ and let $\Sigma = \{a^{(1)} \mid a \in A\} \cup \{e^{(0)}\}$. We now construct a linear tree homomorphism N (where 'linear' means that, for every input symbol σ of rank $k \geq 0$ and for every $i \in [k]$, $\langle q, x_i \rangle$ occurs at most once in the right-hand side of the (q,σ)-rule) and a YIELD mapping Y_f such that $Y_f(\tau_N(\mathrm{sm}(L'))) = \tau_M(\mathrm{sm}(L))$ for the regular language $L' = \{\#u^r \mid u \in L\}$, with $\# \notin A$. This proves the lemma, because $\mathrm{sm}(L') \subseteq \mathit{REGT}$ and linear tree homomorphisms preserve the regular tree languages (cf., e.g., Theorem 6.10 of [GS84]). The idea of the construction is to simulate M by associating with every state q_j of M a parameter y_j, containing the q_j-translation of M; the tree homomorphism N generates, for input a, constant symbols (q_j, a) for every $j \in [m]$ (and for input $\#$, symbols (q_j, e)), and f maps (q_j, a) to the right-hand side of the (q_j, a)-rule of M (with states replaced by the corresponding parameters).

Define $N = (\{q\}, \Sigma \cup \{\#^{(1)}\}, \Gamma, q, R_N)$ with $\Gamma = \{(q_j, a)^{(0)} \mid j \in [m], a \in \Sigma\} \cup \{\gamma^{(m+1)}, e^{(0)}\}$ and R_N consisting of the following rules.

$$\begin{array}{lll}
\langle q, \#(x_1) \rangle & \to \gamma(\langle q, x_1 \rangle, (q_1, e), \ldots, (q_m, e)) & \\
\langle q, a(x_1) \rangle & \to \gamma(\langle q, x_1 \rangle, (q_1, a), \ldots, (q_m, a)) & \text{for every } a \in A \\
\langle q, e \rangle & \to e &
\end{array}$$

The mapping f is defined as $f(e) = y_1$ and, for every $j \in [m]$ and $a \in \Sigma$, $f((q_j, a)) = \mathrm{rhs}_M(q_j, a)[\langle q_i, x_1 \rangle \leftarrow y_i \mid i \in [m]]$.

Let us now prove the correctness of the construction, i.e., that $Y_f(\tau_N(\mathrm{sm}(L'))) = \tau_M(\mathrm{sm}(L))$. By the definition of L' we have to show that, for every $u \in L$, $Y_f(\tau_N(\mathrm{sm}(\#u^r))) = \tau_M(\mathrm{sm}(u))$. By the definition of N, $Y_f(\tau_N(\mathrm{sm}(\#u^r))) = Y_f(\gamma(\tau_N(\mathrm{sm}(u^r)), (q_1, e), \ldots, (q_m, e))) = Y_f(\tau_N(\mathrm{sm}(u^r)))[y_j \leftarrow \mathrm{rhs}_M(q_j, e) \mid j \in [m]]$. Since $\mathrm{rhs}_M(q_j, e) = M_{q_j}(\mathrm{sm}(\varepsilon))$ this equals $\tau_M(\mathrm{sm}(u))$ by the following claim (for $v = \varepsilon$).

Claim: For every $u, v \in A^*$, $Y_f(\tau_N(\mathrm{sm}(u^r)))[\ldots] = \tau_M(\mathrm{sm}(uv))$, where $[\ldots] = [y_i \leftarrow M_{q_i}(\mathrm{sm}(v)) \mid i \in [m]]$.

The proof of this claim is by induction on the structure of u. If $u = \varepsilon$, then $Y_f(\tau_N(\mathrm{sm}(u^r)))[\ldots] = Y_f(e)[\ldots] = y_1[\ldots] = M_{q_1}(\mathrm{sm}(v)) = \tau_M(\mathrm{sm}(uv))$.

If $u' = ua$ with $u \in A^*$ and $a \in A$, then we get

$$\begin{array}{l}
\quad Y_f(\tau_N(\mathrm{sm}(u'^r)))[\ldots] \\
= Y_f(\tau_N(a(\mathrm{sm}(u^r))))[\ldots] \\
= Y_f(\gamma(\tau_N(\mathrm{sm}(u^r)), (q_1, a), \ldots, (q_m, a)))[\ldots] \\
= Y_f(\tau_N(\mathrm{sm}(u^r)))[y_j \leftarrow f((q_j, a)) \mid j \in [m]][\ldots].
\end{array}$$

The application of $[\dots]$ to $f((q_j,a))$ gives $\mathrm{rhs}_M(q_j,a)[\langle q_i,x_1\rangle \leftarrow M_{q_i}(\mathrm{sm}(v)) \mid i \in [m]]$ which equals $M_{q_j}(\mathrm{sm}(av))$ by Definition 3 and the fact that M is a top-down tree transducer. Thus we get $Y_f(\tau_N(\mathrm{sm}(u^r)))[y_j \leftarrow M_{q_j}(\mathrm{sm}(av)) \mid j \in [m]]$. By induction this equals $\tau_M(\mathrm{sm}(uav)) = \tau_M(\mathrm{sm}(u'v))$. □

Before we prove that the EDT0L-hierarchy is included in the IO-hierarchy, let us consider an example of the construction given in the proof of Lemma 34.

Example 35. Let M_b be the top-down tree transducer defined in the proof of Theorem 22, which translates a tree $\mathrm{sm}(w)$ into a tree with yield $\mathrm{count}_b(w)$, where $w \in A^*$ for an alphabet A. For $A = \{c,d\}$, M_b has the following rules.

$$\begin{aligned}
\langle q_0, c(x_1)\rangle &\to \sigma(c, \sigma(\langle q, x_1\rangle, \langle q_0, x_1\rangle))\\
\langle q_0, d(x_1)\rangle &\to \sigma(d, \sigma(\langle q, x_1\rangle, \langle q_0, x_1\rangle))\\
\langle q_0, e\rangle &\to e\\
\langle q, c(x_1)\rangle &\to \sigma(b, \langle q, x_1\rangle)\\
\langle q, d(x_1)\rangle &\to \sigma(b, \langle q, x_1\rangle)\\
\langle q, e\rangle &\to e
\end{aligned}$$

We now apply the construction of the proof of Lemma 34 to define the linear tree homomorphism N and the YIELD mapping Y_f such that $Y_f(\tau_N(\mathrm{sm}(\#u^r)) = \tau_{M_b}(\mathrm{sm}(u))$ for every $u \in A^*$. Then $N = (\{p\}, \Sigma \cup \{\#^{(1)}\}, \Gamma, p, R_N)$ with $\Sigma = \{c^{(1)}, d^{(1)}, e^{(0)}\}$, $\Gamma = \{(q_0,c)^{(0)}, (q_0,d)^{(0)}, (q_0,e)^{(0)}, (q,c)^{(0)}, (q,d)^{(0)}, (q,e)^{(0)}\} \cup \{\gamma^{(3)}, e^{(0)}\}$, and R_N consists of the following rules.

$$\begin{aligned}
\langle p, \#(x_1)\rangle &\to \gamma(\langle p, x_1\rangle, (q_0,e), (q,e))\\
\langle p, c(x_1)\rangle &\to \gamma(\langle p, x_1\rangle, (q_0,c), (q,c))\\
\langle p, d(x_1)\rangle &\to \gamma(\langle p, x_1\rangle, (q_0,d), (q,d))\\
\langle p, e\rangle &\to e
\end{aligned}$$

The mapping f is defined as $f((q_0,c)) = \sigma(c, \sigma(y_2, y_1))$, $f((q_0,d)) = \sigma(d, \sigma(y_2, y_1))$, $f((q_0,e)) = f((q,e)) = e$, $f((q,c)) = f((q,d)) = \sigma(b, y_2)$, and $f(e) = y_1$.

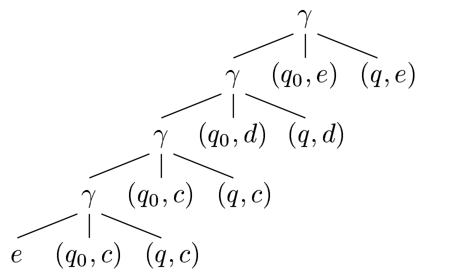

Fig. 4. The tree $t = \tau_N(\#(d(c(c(e)))))$

Now, consider the string $u = ccd$. Then

$$\tau_{M_b}(\mathrm{sm}(u)) = \sigma(c, \sigma(\sigma(b, \sigma(b,e)), \sigma(c, \sigma(\sigma(b,e), \sigma(d, \sigma(e,e)))))).$$

The application of τ_N to the tree $\mathrm{sm}(\#u^r) = \#(d(c(c(e))))$ gives the tree t shown in Figure 4. Let us now compute $Y_f(t)$ in a bottom-up fashion:

$$\begin{aligned}
Y_f(t/111) &= Y_f(e)[y_1 \leftarrow Y_f((q_0,c)), y_2 \leftarrow Y_f((q,c))]\\
&= y_1[y_1 \leftarrow \sigma(c, \sigma(y_2, y_1)), y_2 \leftarrow \sigma(b, y_2)]\\
&= \sigma(c, \sigma(y_2, y_1)),
\end{aligned}$$

$$
\begin{aligned}
Y_f(t/11) &= Y_f(t/111)[y_1 \leftarrow Y_f((q_0,c)), y_2 \leftarrow Y_f((q,c))]\\
&= \sigma(c,\sigma(y_2,y_1))[y_1 \leftarrow \sigma(c,\sigma(y_2,y_1)), y_2 \leftarrow \sigma(b,y_2)]\\
&= \sigma(c,\sigma(\sigma(b,y_2),\sigma(c,\sigma(y_2,y_1)))),\\
Y_f(t/1) &= Y_f(t/11)[y_1 \leftarrow Y_f((q_0,d)), y_2 \leftarrow Y_f((q,d))]\\
&= \sigma(c,\sigma(\sigma(b,y_2),\sigma(c,\sigma(y_2,y_1))))[y_1 \leftarrow \sigma(d,\sigma(y_2,y_1)), y_2 \leftarrow \sigma(b,y_2)]\\
&= \sigma(c,\sigma(\sigma(b,\sigma(b,y_2)),\sigma(c,\sigma(\sigma(b,y_2),\sigma(d,\sigma(y_2,y_1)))))),
\end{aligned}
$$

and finally $Y_f(t) = Y_f(t/1)[y_1 \leftarrow Y_f((q_0,e)), y_2 \leftarrow Y_f((q,e))] = Y_f(t/1)[y_1 \leftarrow e, y_2 \leftarrow e]$ which is precisely the tree $\tau_{M_b}(\mathrm{sm}(u))$ displayed above. □

We now prove the inclusion of the EDT0L-hierarchy in the IO-hierarchy.

Theorem 36. For every $n \geq 1$, $\mathit{EDT0L}^n(\mathit{REG}) \subseteq \mathit{IO}(n)$.

Proof. As shown in the proof of Theorem 21, $\mathit{EDT0L}^n \subseteq \mathrm{sm} \circ \mathit{MTT}^{n-1} \circ y$. Since $\mathrm{MTT}^n = T \circ \mathit{YIELD}^n$ by Corollary 4.13 of [EV85], this equals $\mathrm{sm} \circ T \circ \mathit{YIELD}^{n-1} \circ y$. Applying this to REG gives $y\,\mathit{YIELD}^{n-1}(T(\mathrm{sm}(\mathit{REG})))$. By Lemma 34 this is included in $y\,\mathit{YIELD}^{n-1}(\mathit{YIELD}(\mathit{REGT})) = y\,\mathit{YIELD}^n(\mathit{REGT}) = \mathit{IO}(n)$. □

Note that Theorem 22 implies that $\mathit{EDT0L}^{n+2}(\mathit{REG}) - \mathit{IO}(n) \neq \varnothing$. It would be interesting to know whether this result could be improved to $\mathit{EDT0L}^{n+1}(\mathit{REG}) - \mathit{IO}(n) \neq \varnothing$; but this requires stronger methods of proving that a language is not in the IO-hierarchy, which we do not have (cf. the discussion of open problems at the end of Section 8). Note further that Theorem 36 gives a somewhat tighter link between the EDT0L- and the IO-hierarchy, than the one between the ET0L- and the OI-hierarchy in [Vog88] which states that $\mathit{ET0L}^n(\mathit{REG})$ is included in $\mathit{OI}(2n-1)$.

8 Conclusions and Open Problems

We have proved the properness of the yMTT-, EDT0L-, and IO-hierarchies in Theorems 23, 24, and 33, respectively. In this section we want to discuss the relationships between the different hierarchies of string languages that were considered in this paper. By "the hierarchy $X(n)$" we mean that, for every $n \geq 1$, $X(n)$ is a class of languages and $X(n) \subseteq X(n+1)$. The hierarchy $X(n)$ is proper if $X(n) \subsetneq X(n+1)$ for every $n \geq 1$. Denote by $X(*)$ the union $\bigcup_{n\geq 1} X(n)$. For two hierarchies $X(n)$ and $Y(n)$ we want to know, whether the following holds.

– Is the hierarchy $X(n)$ included in the hierarchy $Y(n)$, i.e., is $X(*) \subseteq Y(*)$? And if so, is the inclusion proper, i.e., is $X(*) \subsetneq Y(*)$?
– Is $X(n)$ a *subhierarchy* of $Y(n)$? By this we mean that there is an $m \in \mathbb{N}$ such that for every $n \geq 1$: $X(m+n) \subseteq Y(n)$ and $X(m+n+1) - Y(n) \neq \varnothing$.
– Is $X(n)$ *small* in $Y(n)$? This means that $Y(1) - X(*) \neq \varnothing$.

If $X(n)$ is a subhierarchy of $Y(n)$, then $X(m+n)$ and $Y(n)$ are proper hierarchies, and $X(*) \subseteq Y(*)$. If $X(*) \subseteq Y(*)$ and $X(n)$ is small in $Y(n)$, then $X(*) \subsetneq Y(*)$. If $X(n)$ is a small subhierarchy of $Y(n)$, then the infinite inclusion diagram in Fig. 5 is a Hasse diagram.

We have shown (in Theorems 22 and 32) that the EDT0L- and IO-hierarchies are subhierarchies of the yMTT-hierarchy. Note that for Y being the yMTT-hierarchy, $m = 1$ if X is the EDT0L-hierarchy, and $m = 0$ if X is the IO-hierarchy.

Let us briefly consider another type of tree transducer and show that the output string languages of its compositions gives rise to a subhierarchy of the yMTT-hierarchy: the attributed tree transducer (ATT) [Fül81,FV98], which is a formal model for attribute grammars. It is well known that $\mathit{YIELD} \subseteq \mathit{ATT} \subseteq \mathit{MTT}$ (cf. Corollary 6.24 and Lemma 6.1 of [FV98]), where ATT denotes the class of all translations realized by ATTs. Thus, $\mathit{IO}(n) \subseteq y\mathit{ATT}^n(\mathit{REGT}) \subseteq y\mathit{MTT}^n(\mathit{REGT})$. By Theorem 32 we obtain the following corollary.

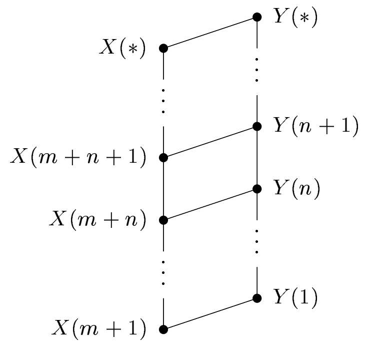

Fig. 5. The Hasse diagram for: "$X(n)$ is a small subhierarchy of $Y(n)$"

Theorem 37. For $n \geq 1$,

(a) $yATT^{n+1}(REGT) - yMTT^{n}(REGT) \neq \varnothing$.
(b) $yATT^{n}(REGT) \subsetneq yATT^{n+1}(REGT)$.

Thus, the hierarchy $yATT^{n}(REGT)$ is proper, and it is a subhierarchy of the yMTT-hierarchy. Note that it is open whether $yATT^{n}(REGT) \subsetneq yMTT^{n}(REGT)$. Note further that the yATT-hierarchy is not small in the yMTT-hierarchy, because, in fact, $yMTT^{n}(REGT) \subseteq yATT^{n+1}(REGT)$, and so $yATT^{*}(REGT)$ equals $yMTT^{*}(REGT)$, see, e.g., Section 6 of [FV98].

Now, we prove that the EDT0L-hierarchy is a small subhierarchy of the ET0L-hierarchy $ET0L^{n}(REG)$, where $ET0L$ denotes the class of all nondeterministic EDT0L translations (cf. the discussion after Theorem 25).

Theorem 38. The EDT0L-hierarchy is a small subhierarchy of the ET0L-hierarchy.

Proof. By Corollary 27, $CF - EDT0L^{*}(REG) \neq \varnothing$. Since $CF \subseteq ET0L(REG)$, this shows that the EDT0L-hierarchy is small in the ET0L-hierarchy. Alternatively, this follows from Theorem 25. It remains to show that $EDT0L^{n+1}(REG) - ET0L^{n}(REG) \neq \varnothing$. For a language L define the copy operations c_2 and c_* as $c_2(L) = \{w\$w \mid w \in L\}$ and $c_*(L) = \{(w\$)^n \mid w \in L, n \geq 1\}$ for a symbol \$ not in L. Let $L_2 = L_{\mathrm{ec}}$ and, for $n \geq 2$, let $L_{n+1} = c_2(\mathrm{count}_b(c_*(L_n)))$ for a symbol b not in $c_*(L_n)$.

(1) $L_n \in EDT0L^{n}(REG)$. As shown in the proof of Theorem 22, $L_{\mathrm{ec}} \in EDT0L^{2}(REG)$ and $\mathrm{count}_b \in EDT0L$. Hence $L_n \in EDT0L^{n}(REG)$, because it is easy to see that $EDT0L^{n}(REG)$ is closed under c_2 and c_*: Let $L = \tau_n(\cdots\tau_1(R)\cdots) \in EDT0L^{n}(REG)$ with $R \in REG$ and $\tau_i \in EDT0L$ for $i \in [n]$. To obtain $c_2(L)$ and $c_*(L)$, change R into the regular languages aR and a^*R, respectively, where a is a symbol not in R and not used in any of the τ_i. Now τ_1 is changed into τ_1' in such a way that $\tau_1'(aR)$ equals $a\tau_1(R)$, and $\tau_1'(a^*R)$ equals $a^*\tau_1(R)$. Similarly, for $i \in [n-1]$, τ_i is changed into τ_i' which translates $a\tau_{i-1}(\cdots\tau_1(R)\cdots)$ into $a\tau_i(\tau_{i-1}(\cdots\tau_1(R)\cdots))$ and $a^*\tau_{i-1}(\cdots\tau_1(R)\cdots)$ into $a^*\tau_i(\tau_{i-1}(\cdots\tau_1(R)\cdots))$. Finally, the translation τ_n is changed into τ_n' which translates $a\tau_{n-1}(\cdots\tau_1(R)\cdots)$ into $\tau_n(\cdots\tau_1(R)\cdots)\$\tau_n(\cdots\tau_1(R)\cdots) = c_2(L)$, and, similarly, $a^*\tau_{n-1}(\cdots\tau_1(R)\cdots)$ into $c_*(L)$.

(2) $L_n \notin ET0L^{n-1}(REG)$. For $n = 2$ this follows from Theorem 3.16 of [Eng82]: $L_{\mathrm{ec}} \notin yN\text{-}T(REGT)$. For $n > 2$ the result is obtained, by induction, as follows. It is straightforward to show that Theorem 3.1 of [Eng82], which is the bridge theorem (Theorem 3.2.14) of [ERS80], can also be stated for the operation count_b in place of the operation rub (in fact, it holds in general for languages L and L' that satisfy the assumptions of Lemma 17). Then the proof of Theorem 4.2 of [Eng82] (with rub changed into count_b) shows that if $L \notin EDT0L(ET0L^{n-2}(REG))$ then

– $\mathrm{count}_b(c_*(L)) \notin \mathit{EDT0L}(\mathit{ET0L}^{n-1}(\mathit{REG}))$ and
– $c_2(\mathrm{count}_b(c_*(L))) \notin \mathit{ET0L}^n(\mathit{REG})$.

Since $\mathit{ET0L}^{n-1}(\mathit{REG})) \supseteq \mathit{EDT0L}(\mathit{ET0L}^{n-2}(\mathit{REG}))$, this shows that for $n \geq 2$, if $L_n \notin \mathit{ET0L}^{n-1}(\mathit{REG})$, then $L_{n+1} \notin \mathit{ET0L}^n(\mathit{REG})$. $\square$

The proof of Theorem 38 shows that the properness of the ET0L-hierarchy is not caused by the alternation of copying and nondeterminism (as stated in [Eng82]), but rather by the alternation of copying and insertion.

Let us now summarize the relationships between the different hierarchies of string languages that have been considered, together with the nondeterministic version of the yMTT-hierarchy, and the nondeterministic top-down tree transducer hierarchy of [Eng82]. Let $\mathit{N\text{-}MTT}$ denote the class of translations realized by nondeterministic macro tree transducers and let, as before, $\mathit{N\text{-}T}$ denote the class of translations realized by nondeterministic top-down tree transducers. Note that the derivations of nondeterministic MTTs can be restricted to be OI (outside-in), see Corollary 3.13 of [EV85]. Furthermore, the composition closure $\mathit{N\text{-}MTT}^*$ can also be obtained by the restriction to IO-derivations, i.e., this class equals $\mathit{N\text{-}MTT}^*_{\mathrm{IO}}$, where $\mathit{N\text{-}MTT}_{\mathrm{IO}}$ denotes the class of translations realized by nondeterministic MTTs restricted to IO-derivations, see Theorem 7.3 of [EV85]. By the same theorem, $\mathit{N\text{-}MTT}^* = (\mathit{N\text{-}T} \cup \mathit{YIELD})^*$, and so $\mathit{yN\text{-}MTT}^*(\mathit{REGT})$ is the class of languages considered in [DE98], cf. the Introduction.

$$\begin{tikzcd}
 & & \mathit{yN\text{-}MTT}^*(\mathit{REGT}) & & \\
\begin{array}{c}\mathit{yMTT}^*(\mathit{REGT}) \\ = \mathit{yATT}^*(\mathit{REGT})\end{array} \arrow[urr] & & \mathit{OI}(*) \arrow[u, dash] & & \\
\mathit{IO}(*) \arrow[u, dash, "\mathrm{sub}"] & & \mathit{ET0L}^*(\mathit{REG}) \arrow[u, dash] & & \mathit{yN\text{-}T}^*(\mathit{REGT}) \arrow[uull] \\
 & \mathit{EDT0L}^*(\mathit{REG}) \arrow[ul] \arrow[ur, "\mathrm{sub}"'] & & \mathit{2GSM}^*(\mathit{REG}) \arrow[ul, "\mathrm{sub}"] \arrow[ur, "\mathrm{sub}"'] &
\end{tikzcd}$$

Fig. 6. Inclusions of hierarchies of string languages.

Figure 6 shows an inclusion diagram, where an ascending line from $X(*)$ to $Y(*)$ indicates that $X(*) \subseteq Y(*)$, and the label 'sub' indicates that $X(n)$ is a subhierarchy of $Y(n)$; an arrow from $X(*)$ to $Y(*)$ indicates a proper inclusion (and even that $X(n)$ is small in $Y(n)$). Note that the four hierarchies in the left part of the figure, i.e., $\mathit{EDT0L}^n(\mathit{REGT})$, $\mathit{IO}(n)$, $\mathit{yATT}^n(\mathit{REGT})$, and $\mathit{yMTT}^n(\mathit{REGT})$ are generated by total deterministic devices and the other five hierarchies involve partial nondeterministic devices.

Besides the hierarchy $\mathit{yN\text{-}MTT}^n(\mathit{REGT})$, all hierarchies in the figure are (now) known to be proper: For the 2GSM-, ET0L-, and yN-T-hierarchies this is known from [Eng82] (properness of the 2GSM-hierarchy was obtained independently in [Gre78]), for the EDT0L-hierarchy by Theorem 24, for the IO-hierarchy by Theorem 33, for the yMTT- and yATT-hierarchy by Theorems 23 and 37(b), respectively, and for the OI-hierarchy from Theorem 7.4 of [Eng91]. Infinity of the IO- and OI-hierarchies was proved in [Dam82]. Note that deterministic two-way generalized sequential machines and (deterministic) top-down tree transducers are closed under composition and therefore do not give rise to proper hierarchies.

Let us now discuss the inclusions in Figure 6. The inclusions of $EDT0L^*(REG)$ in $ET0L^*(REG)$, $yN\text{-}T^*(REGT)$ in $yN\text{-}MTT^*(REGT)$, and $yMTT^*(REGT)$ in $yN\text{-}MTT^*(REGT)$ hold by definition. The inclusion of the EDT0L-hierarchy in the IO-hierarchy follows from Theorem 36. The inclusions of $2GSM^*(REG)$ in $ET0L^*(REG)$ and in $yN\text{-}T^*(REGT)$ follow from Corollary 4.6 and Theorem 5.5 of [ERS80], see Lemma 4.6 of [Eng82]. The inclusion of $IO(*)$ in the yATT- and yMTT-hierarchy was discussed at the beginning of this section. The class $ET0L^*(REG)$ is in $OI(*)$ by Theorem 14 of [Vog88] (cf. also [Eng91]) and the inclusion of $OI(*)$ in $yN\text{-}MTT^*(REGT)$ follows from Theorem 8.1 of [EV88] (as discussed at the end of that paper).

Next, consider the subhierarchy and smallness relations in Figure 6. The fact that the 2GSM-hierarchy is a small subhierarchy of both $ET0L^n(REG)$ and $yN\text{-}T^n(REGT)$ holds by Theorem 4.8 of [Eng82] (indeed, the smallness follows from the fact that $CF - 2GSM^*(REG) \neq \varnothing$, which was proved in [Gre78]). By Theorem 38, the EDT0L-hierarchy is a small subhierarchy of the ET0L-hierarchy. From Theorem 25 and the fact that $2GSM(REG) \subseteq ET0L(REG) \subseteq yN\text{-}T(REGT) \subseteq yN\text{-}MTT(REGT)$, it follows that $yMTT^n(REGT)$ is small in $yN\text{-}MTT^n(REGT)$. The smallness of $yN\text{-}T^n(REGT)$ in $yN\text{-}MTT^n(REGT)$ follows from the fact that $L_{\text{ec}} \notin yN\text{-}T^*(REGT)$ (as mentioned before Theorem 22) and the fact (shown in the proof of Theorem 22) that L_{ec} is in $EDT0L^2(REG)$ which is included in $yN\text{-}MTT(REGT)$ by Theorem 21. The EDT0L-hierarchy is small in the IO-hierarchy, because, by Corollary 27, there is a context-free language not in $EDT0L^*(REG)$, and $IO(1)$ includes the context-free languages (cf. Theorem 7.9 of [ES78]). By Theorem 32, $IO(n)$ is a subhierarchy of the yMTT-hierarchy. Note that it is not indicated in Figure 6 that the EDT0L-hierarchy and the yATT-hierarchy are subhierarchies of the yMTT-hierarchy.

We conclude by mentioning some open problems related to the diagram in Figure 6. First of all, are there more subhierarchy relationships between the hierarchies shown in the figure? In particular, is the yMTT-hierarchy a subhierarchy of its nondeterministic version $yN\text{-}MTT^n(REGT)$? With respect to inclusion consider the following open problems.

- $IO(*) \subsetneq yMTT^*(REGT)$?
- $ET0L^*(REG) \subsetneq OI(*)$?
- $IO(*) \nsubseteq OI(*)$?
- $yN\text{-}T^*(REGT) \nsubseteq OI(*)$?

Our conjecture is that all these statements hold. Together with the facts that $2GSM^*(REG) - yMTT^*(REGT) \neq \varnothing$ by Theorem 25, and that $L_{\text{ec}} \in EDT0L^*(REG) - yN\text{-}T^*(REGT)$ as discussed above, this would prove that Figure 6 is a Hasse diagram. The problem with proving the conjectures listed above is that we do not have methods to show that languages are not in the OI- and ET0L-hierarchies, and need stronger methods to show that languages are not in the IO-hierarchy.

References

[BCN81] L. Boasson, B. Courcelle, and M. Nivat. The rational index: a complexity measure for languages. *SIAM Journal on Computing*, 10(2):284–296, 1981.

[CF82] B. Courcelle and P. Franchi-Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191 and 235–257, 1982.

[Cou83] B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.

[Dam82] W. Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20:95–207, 1982.

[DE98] F. Drewes and J. Engelfriet. Decidability of finiteness of ranges of tree transductions. *Inform. and Comput.*, 145:1–50, 1998.

[Dow74] P. J. Downey. Formal languages and recursion schemes. Technical Report TR-16-74, Harvard University, 1974.

[EM99] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inform. and Comput.*, 154:34–91, 1999.

[Eng77] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Systems Theory*, 10:289–303, 1977.

[Eng78] J. Engelfriet. On tree transducers for partial functions. *Informat. Processing Let.*, 7:170–172, 1978.

[Eng80] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory; perspectives and open problems*. New York, Academic Press, 1980.

[Eng82] J. Engelfriet. Three hierarchies of transducers. *Math. Systems Theory*, 15:95–125, 1982.

[Eng91] J. Engelfriet. Iterated stack automata and complexity classes. *Inform. and Comput.*, 95(1):21–75, 1991.

[ERS80] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *J. of Comp. Syst. Sci.*, 20:150–202, 1980.

[ES78] J. Engelfriet and E.M. Schmidt. IO and OI, Part II. *J. of Comp. Syst. Sci.*, 16:67–99, 1978.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. of Comp. Syst. Sci.*, 31:71–146, 1985.

[EV88] J. Engelfriet and H. Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26:131–192, 1988.

[EV94] J. Engelfriet and H. Vogler. The translation power of top-down tree-to-graph transducers. *J. of Comp. Syst. Sci.*, 49:258–305, 1994.

[Fis68] M.J. Fischer. *Grammars with macro-like productions*. PhD thesis, Harvard University, Massachusetts, 1968.

[Fül81] Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–279, 1981.

[FV98] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics – Formal Models based on Tree Transducers*. EATCS Monographs on Theoretical Computer Science (W. Brauer, G. Rozenberg, A. Salomaa, eds.). Springer-Verlag, 1998.

[Gre78] S. A. Greibach. Hierarchy theorems for two-way finite state transducers. *Acta Informatica*, 11:89–101, 1978.

[Gre81] S. A. Greibach. Formal languages: origins and directions. *Ann. of the Hist. of Comput.*, 3(1):14–41, 1981.

[GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[GS97] F. Gécseg and M. Steinby. Tree automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3*, chapter 1. Springer-Verlag, 1997.

[HU79] J. W. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979.

[Man98] S. Maneth. The generating power of total deterministic tree transducers. *Inform. and Comput.*, 147:111–144, 1998.

[Man99] S. Maneth. String languages generated by total deterministic macro tree transducers. In W. Thomas, editor, *Proc. FOSSACS'99*, volume 1578 of *LNCS*, pages 258–272. Springer-Verlag, 1999.

[Rou70] W.C. Rounds. Mappings and grammars on trees. *Math. Systems Theory*, 4:257–287, 1970.

[Roz73] G. Rozenberg. Extension of tabled 0L-systems and languages. *Internat. J. Comp. Inform. Sci.*, 2:311–336, 1973.

[vL76] J. van Leeuwen. Variations of a new machine model. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science, Houston, Texas*, pages 228–235. IEEE Computer Society Press, 1976.

[Vog88] H. Vogler. The OI-hierarchy is closed under control. *Inform. and Comput.*, 78:187–204, 1988.