Math. Systems Theory **15**, 95–125 (1982)

Mathematical Systems Theory

Three Hierarchies of Transducers

Joost Engelfriet

Twente University of Technology, Enschede, The Netherlands

Abstract. Composition of top-down tree transducers yields a proper hierarchy of transductions and of output languages. The same is true for ETOL systems (viewed as transducers) and for two-way generalized sequential machines.

1. Introduction

Soon after the introduction of the top-down tree transducer [35, 40], as a model of syntax-directed translation [4], one of the main problems turned out to concern their composition. In fact, the class of top-down tree transductions is not closed under composition [35, 40] unless certain restrictions or extensions are added to the main model. Thus the class of deterministic top-down tree transducers becomes closed under composition when the transducers are restricted to be total [35] or extended to have regular look-ahead [17]. A similar statement holds for linear (noncopying) top-down tree transducers. These phenomena are mainly due to the fact that a top-down tree transducer cannot handle nondeterminism followed by copying, cf. [16, 9].

Given the nonclosure of the class of tree transducers under composition, one may ask whether composition of tree transducers gives rise to a proper hierarchy. In other words the natural question arises whether $n+1$ transducers are more powerful than n, or whether perhaps there exists a constant N such that any sequence of tree transducers can be simulated by a sequence of at most N tree transducers. In this paper we prove that $n+1$ tree transducers are more powerful than n (as conjectured in [32, 9, 33]).

Related to the problems concerning composition is that of determining the closure properties of the class of output tree languages, or "surface sets", of tree transducers (given some class of input tree languages). Thus Rounds showed that the class of output tree languages of deterministic top-down tree transducers (with recognizable input tree languages) is closed under deterministic tree transductions

0025-5661/82/0015-0095$06.20

[35]. But Ogden and Rounds [32] proved that $T(\mathrm{REC}) \subsetneq T^2(\mathrm{REC})$, where T denotes the class of top-down tree transductions, T^2 the class of compositions of two transductions from T, and REC the class of recognizable tree languages [39] (i.e., roughly, the class of derivation tree languages of context-free grammars). This result means that the class $T(\mathrm{REC})$ of output languages of top-down tree transducers is not closed under top-down tree transductions, and thus it strengthens that of nonclosure of T under composition. It was strongly conjectured in [32] that composition of n top-down tree transducers gives rise to a proper hierarchy even with respect to output tree languages, i.e., for all $n, T^n(\mathrm{REC}) \subsetneq T^{n+1}(\mathrm{REC})$. In this paper we show that this is indeed the case. In fact we prove that a proper hierarchy is formed even by the corresponding classes of tree transformation languages $yT^n(\mathrm{REC})$: string languages obtained by taking the yield, denoted by y, of the output trees (i.e., the string of their leaf labels). Moreover, even for an arbitrary class K of input tree languages (satisfying a few closure properties) a proper hierarchy is obtained whenever the first class is properly contained in the second: if $yT(K) \subsetneq yT^2(K)$ then, for all n, $yT^n(K) \subsetneq yT^{n+1}(K)$. After [32] some results concerning this problem were established in [9, 19, 21, 33]. Baker [9, 46] provided partial solutions to the problem of properness of both the $T^n(K)$ hierarchy and the $yT^n(K)$ hierarchy, and reduced it to problems concerning closure properties of the classes involved. She showed moreover that the bottom-up tree transducers give rise to an infinite hierarchy if and only if the top-down transducers do (see also [16]). Perrault [33] established an intercalation theorem for $yT(\mathrm{REC})$ which he used to show that $yT(\mathrm{REC}) \subsetneq yT^2\mathrm{REC})$. In [21, 19] partial solutions to the problem of properness of the $yT^n(K)$ hierarchy were given (using the operations of copying and of regular substitution, respectively). The ideas developed in [19], in particular concerning bounded copying tree transducers, led to the present paper.

To prove properness of the $yT^n(K)$ hierarchy we establish some general properties of these classes (and some of their subclasses) which can also be used to obtain proper hierarchies for two related string transducers. The first of these transducers is the ETOL system [36] viewed as a transducer of its control string into the string it generates (cf. [5, 15, 19] for the relationship between ETOL systems and top-down tree transducers). We establish a proper hierarchy of classes of string languages obtained by iterating the process of controlling ETOL systems (starting with a given class of control languages, such as the regular languages). Since classes of controlled ETOL languages are full hyper-AFLs (i.e., full AFLs closed under iterated substitution), see [6], this provides us with a means to construct proper hierarchies of full hyper-AFLs containing a given class of languages.

The second of these transducers is the 2-way generalized sequential machine (2gsm) [3, 13, 26, 30, 34]. The relationship between top-down tree transducers and 2-way automata was established in [4, 19] and that between ETOL systems and 2-way automata in [34, 42, 19]. Results concerning composition of 2gsm's were obtained in [2, 30, 26, 10]. We show that if $2\mathrm{GSM}(K) \subsetneq 2\mathrm{GSM}^2(K)$ then, for all n, $2\mathrm{GSM}^n(K) \subsetneq 2\mathrm{GSM}^{n+1}(K)$, and in particular the classes $2\mathrm{GSM}^n(\mathrm{REG})$ form a proper hierarchy (these results were obtained independently by Greibach [27]). Moreover, the 2GSM hierarchy is a subhierarchy of both the ETOL hierarchy and the top-down tree transducer hierarchy. Examples which show the properness of these hierarchies can already be found in the 2GSM hierarchy. Properness of all three hierarchies is caused by the two facilities of nondeterminism and copying present in the transducers. Actually the examples with which we prove the hierarchies to be proper are constructed (from some initial language) by repeated application of two operations: one is a special regular substitution which inserts any number of occurrences of a new symbol anywhere in the strings of a language, and the other is the operation c_* of unbounded copying defined by $c_*(L)=\{(w\$)^n \mid n\geq 1, w\in L\}$ where \$ is a new symbol.

To establish these hierarchy results we will not use intercalation theorems (which soon become too complex), but "bridge theorems" (in the terminology of 12]) which construct, from each language not in some class by means of some operation, another language not in some larger class. Thus these theorems have the form "if $L\notin K_1$ then $f(L)\notin K_2$" or equivalently "if $f(L)\in K_2$ then $L\in K_1$", where $K_1\subseteq K_2$. In case for instance $K_2\subseteq f(K_2)\subseteq K_3$, we get that $L\in K_2-K_1$ implies $f(L)\in K_3-K_2$, i.e., properness of the inclusion $K_1\subseteq K_2$ implies properness of the inclusion $K_2\subseteq K_3$. In such a way properness of inclusions of a hierarchy can be shown in a step by step fashion; the problem is of course to find the appropriate classes and operations for which bridge theorems can be proved.

The power of the method of bridges is shown by the fact that just two bridge theorems are needed to prove all our hierarchy results. On the other hand, bridge theorems should always be accompanied by several results on closure properties, for two reasons. First, to show that $f(K_2)\subseteq K_3$ one usually proves closure of K_3 under the operation f. Second, to show "if $f(L)\in K_2$, then $L\in K_1$" one usually shows that $f(L)$, or an appropriate subset of it, is in K_1, and then one applies another operation (under which K_1 is closed) to retrieve L from $f(L)$, or its subset. Thus the method of bridges is intimately related to the theory of operations on languages [23].

Bridge theorems are usually easier to obtain than intercalation (or pumping) theorems. In fact, in an intercalation theorem for a class K_2 a property should be expressed which holds for *all* languages in K_2, as a result of general properties of the device defining the languages of K_2. Then it has to be shown that a language of a special form, say $f(L)$, does not satisfy the property. However, in a bridge theorem one considers *only* languages of a special form ($f(L)$) and one shows how the special form of the language $f(L)$ forces the device defining $f(L)$ to have certain restricted properties. This then implies that $f(L)$, or an appropriate subset of it, belongs to a smaller class K_1. Thus, rather than looking at general properties of languages forced by general properties of the device (as in an intercalation theorem), one looks (in a bridge theorem) at restricted properties of the device forced by restricted properties of languages.

Bridge theorems were used, among many others, by Greibach [24] to obtain proper hierarchies of full AFLs, by Skyum [38] to establish proper inclusions between several classes of languages generated by parallel rewriting systems (such as ETOL systems), and by Baker [46] to give a partial solution to the problem of properness of the yT^n(REC) hierarchy. The operations involved in their bridge theorems were substitution, copying (i.e., the operation c_2 defined by $c_2(L)=\{w\$w \mid w\in L\}$) and regular substitution, respectively. As mentioned above, we will use a special regular substitution and c_*: operations expressing nondeterminism and copying.

This paper consists of four sections of which this is the first. Section 2 contains preliminary definitions and a few basic lemmas. In Section 3 we present three bridge theorems (Theorems 3.1, 3.6 and 3.9) from which properness of the tree transducer hierarchy follows (for arbitrary classes of input tree languages in Theorem 3.12, and in particular for REC in Theorem 3.14). Using another bridge result, we give concrete examples of languages not in yT^n(REC) for any n (Theorem 3.16). In Section 4 we establish the hierarchies of ETOL systems and 2-way gsm's (in Theorems 4.2, 4.3 and Theorem 4.7, respectively). We show the existence of proper hierarchies of full hyper-AFLs (Theorems 4.5, 4.10), and we show that the 2GSM hierarchy is a small subhierarchy of both the ETOL hierarchy and the top-down tree transducer hierarchy (Theorem 4.8).

2. Preliminaries

We assume the reader to be familiar with the basic facts of formal language theory [29, 23], in particular some tree language theory [41]. In this section we fix some notation, define some more or less well-known concepts and prove a few lemmas.

Whenever it does not give rise to confusion we identify a class K of languages with the class $\{L-\{\lambda\} | L\in K\}$, where λ denotes the empty string.

We denote by REG, CF and CS the classes of regular, context-free and context-sensitive languages, respectively.

A *hierarchy* is a family of classes K_n, $n\geq 1$, such that $K_n \subseteq K_{n+1}$ for all n. It is denoted by $\{K_n\}$; we also refer to $\{K_n\}$ as "the K_n hierarchy". The hierarchy $\{K_n\}$ is *proper* if $K_n \subsetneq K_{n+1}$ for all n.

In an inclusion diagram (such as those in Fig. 2 and 3) an ascending line from K_1 to K_2 means that $K_1 \subseteq K_2$. An inclusion diagram is *correct* if (i) an ascending line from K_1 to K_2 means proper inclusion ($K_1 \subsetneq K_2$), and (ii) if K_1 and K_2 are not connected by any path in the diagram, then they are incomparable with respect to inclusion.

2.1. *Trees and tree languages*

An alphabet Σ is *ranked* if $\Sigma = \cup\{\Sigma_n | n\geq 0\}$, where the Σ_n are (not necessarily disjoint) subsets of Σ such that only finitely many of them are nonempty. If $\sigma\in\Sigma_n$, then we say that σ has rank n. The set T_Σ of *trees* over Σ is the smallest set of strings over $\Sigma\cup\{(\ ,\ )\}$ such that (1) $\Sigma_0 \subseteq T_\Sigma$ and (2) if $\sigma\in\Sigma_n (n\geq 1)$ and $t_1,\ldots,t_n \in T_\Sigma$, then $\sigma(t_1\ldots t_n)\in T_\Sigma$.

If Y is a set of strings, then $T_\Sigma[Y]$ is the smallest set of strings such that (1) $\Sigma_0 \cup Y \subseteq T_\Sigma[Y]$ and (2) if $\sigma\in\Sigma_n(n\geq 1)$ and $t_1,\ldots,t_n\in T_\Sigma[Y]$, then $\sigma(t_1\ldots t_n)\in T_\Sigma[Y]$. Let $X=\{x_1,x_2,x_3,\ldots\}$ and, for $k\geq 0$, $X_k=\{x_1,\ldots,x_k\}$. For a tree $t\in T_\Sigma[X_k]$ and trees $t_1,\ldots,t_k$, the result of substituting t_i for x_i in t is denoted by $t[t_1,\ldots,t_k]$.

Note that we have defined trees to be a particular type of strings. We shall make extensive use of this fact in our notation, in particular concatenation of strings is denoted by juxtaposition as usual. We shall also use the usual more intuitive terminology with respect to trees. Thus a tree consists of *labeled nodes*, such that a node labeled $\sigma \in \Sigma_n$ has n sons (ordered from left to right). The *root* of a tree is the unique node without a father. A *subtree* of a tree consists of a node (the root of the subtree) together with all its descendants. As a notational example, if $t \in T_\Sigma$ has a subtree $s \in T_\Sigma$, then we write $t = usv$, where u and v are strings over $\Sigma \cup \{(\ ,\ )\}$ which together form the part of t outside s (u and v are not trees themselves). A *leaf* of a tree is labeled with a symbol of rank 0, and the yield of a tree is the string of labels of its leaves.

Formally we define the *yield* of a tree t, denoted by yield(t) or $y(t)$ or even yt, as follows. We introduce a special symbol e of rank 0 with yield λ (the empty string).

(1) For $\sigma \in \Sigma_0$, $\text{yield}(\sigma) = \sigma$ if $\sigma \neq e$,
$\text{yield}(e) = \lambda$.

(2) For $\sigma \in \Sigma_n (n \geq 1)$ and $t_1, \ldots, t_n \in T_\Sigma$,
$\text{yield}(\sigma(t_1 \ldots t_n)) = \text{yield}(t_1) \ldots \text{yield}(t_n)$.

A *tree language* over Σ is a subset of T_Σ. Note that every tree language is also a (string) language because every tree is a string. For a tree language L and a class K of tree languages, their yields yL and yK are defined to be a string language and a class of string languages, respectively, in the usual way: $yL = \{yt \mid t \in L\}$ and $yK = \{yL \mid L \in K\}$.

A *tree translation* from Σ to Δ (both ranked alphabets) is a subset of $T_\Sigma \times T_\Delta$. Let S be a class of tree translations. Then S^n denotes the class of all tree translations which are the (relational) composition of n elements of S and S^∞ denotes the union of all S^n. For a class K of tree languages, $S(K) = \{M(L) \mid M \in S, L \in K\}$ is the class of *output tree languages* of S (with *input tree languages* from K), and $yS(K) = \{y(M(L)) \mid M \in S, L \in K\}$ is the corresponding class of *tree transformation languages*. K is closed under the translations from S if $S(K) \subseteq K$. The same notation is used for string translations.

A ranked alphabet Σ is *monadic* if $\Sigma = \Sigma_0 = \Sigma_1$ and $\Sigma_n = \varnothing$ for $n \geq 2$. Trees, tree languages and classes of tree languages over such a Σ are also called monadic. We will identify an unranked alphabet Σ with the corresponding monadic ranked alphabet, the monadic tree $\sigma_1(\sigma_2(\ldots \sigma_{n-1}(\sigma_n) \ldots))$ with the string $\sigma_1 \sigma_2 \ldots \sigma_{n-1} \sigma_n$, and hence T_Σ with Σ^+ and tree languages over Σ with (λ-free) string languages over Σ. In this sense monadic trees are also called vertical strings. Note that, by this identification, the yield of a tree is a monadic tree.

2.2. *Top-down tree transducers and their derivations*

In this subsection we define the top-down tree transducer (see also [35, 40, 16, 9]). We discuss some important concepts concerning its derivations.

A *top-down tree transducer* M is a construct $(Q, \Sigma, \Delta, q_0, R)$ where Q is a finite set of states, Σ and Δ are ranked alphabets of input and output symbols respectively, $q_0 \in Q$ is the initial state and R is a finite set of rules of the form

$q(\sigma(x_1 \ldots x_k)) \to t$ with $q \in Q, \sigma \in \Sigma_k (k \geq 0)$ and $t \in T_\Delta[Q(X_k)]$, where $Q(X_k)$ is the finite language $\{q'(x_i) | q' \in Q, 1 \leq i \leq k\}$; recall the definition of $T_\Delta[Y]$ in Section 2.1. For $k=0$, we write $q(\sigma) \to t$, with $t \in T_\Delta$. M is *deterministic* if no two different rules have the same left-hand side. M is a *tree homomorphism* if it is deterministic, has only one state and is total (i.e., for each $\sigma \in \Sigma_k$ there is a rule with left-hand side $q(\sigma(x_1 \ldots x_k))$). M is a (top-down) *finite state relabeling* if all rules in R are of the form $q(\sigma(x_1 \ldots x_k)) \to \tau(q_1(x_1) \ldots q_k(x_k))$ with $\tau \in \Delta_k$. M is a (top-down) *finite tree automaton* if $\Sigma = \Delta$ and all rules in R have the above form with $\tau = \sigma$.

A *configuration* of M is an element of $T_\Delta[Q(T_\Sigma)]$, where $Q(T_\Sigma)$ is the language $\{q(t) | q \in Q, t \in T_\Sigma\}$. A derivation relation between configurations is defined as follows. If $q(\sigma(x_1 \ldots x_k)) \to t$ is a rule and $t_1, \ldots, t_k \in T_\Sigma$, then $uq\ ((t_1 \ldots t_k))v \Rightarrow ut'v$ where $t' = t[t_1, \ldots, t_k]$. A *derivation* $t_1 \Rightarrow t_2 \Rightarrow \ldots \Rightarrow t_n$ is denoted as usual by $t_1 \overset{*}{\Rightarrow} t_n$. *The top-down tree translation* realized by M, also denoted by M, is $M = \{\langle t_1, t_2 \rangle \in T_\Sigma \times T_\Delta | q_0(t_1) \overset{*}{\Rightarrow} t_2\}$.

Since we can view the elements of X to have rank 0, and those of Q to have rank 1, the elements of languages like $T_\Delta[Q(X_k)]$ and $T_\Delta[Q(T_\Sigma)]$ are also trees.

The class of top-down tree transducers and the corresponding class of top-down tree translations is denoted by T. The class of deterministic top-down tree transducers is denoted by DT, and the class of tree homomorphisms by HOM.

Note that, due to the use of the special symbol e with yield$(e) = \lambda$, the class $yT(K)$ is the same as in [19], where yT (i.e., the composition of T with yield) is defined directly by a class of tree-to-string transducers. The same holds for all subclasses of yT which we will consider. Under weak conditions on K (see Section 2.3) the symbol e can be dropped.

A tree language is *recognizable* if it is the domain (or range) of a finite tree automaton. The class of recognizable tree languages is denoted REC; thus REC $= \{M(T_\Sigma) | M$ is a finite tree automaton with input alphabet $\Sigma\}$. Note that yREC $= CF$ [39]. Note also that a (string) language is recognizable (when viewed as a monadic tree language) iff it is regular; thus we identify REG with the class of monadic tree languages in REC.

A top-down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$ may be viewed as a system of (nondeterministic) recursive procedures. Each state q of M is a recursive function procedure which has one parameter t_1 of type tree and returns a tree t_2 $(q(t_1) \overset{*}{\Rightarrow} t_2, t_1 \in T_\Sigma, t_2 \in T_\Delta)$: t_2 is a "q-translation" of t_1 (or *the* q-translation of t_1, if M is deterministic). The q-translation of a tree depends only on the label of its root and the translations of its subtrees. This dependence is expressed by the rules. A rule $q(\sigma(x_1 \ldots x_k)) \to t$ is part of the body of the procedure q and shows how the q-translation of a tree with root label σ is expressed (by t) in terms of certain translations of its subtrees (denoted $q'(x_i)$). Derivations as defined above mirror the usual operational semantics of recursive procedures: a configuration in a derivation $q_0(t_1) \overset{*}{\Rightarrow} t_2$ is a partially evaluated call $q_0(t_1)$ in which calls with subtrees of t_1 as actual parameters still occur. Note that in the right-hand side t of a rule $q(\sigma(x_1 \ldots x_k)) \to t$ a formal parameter x_i may occur more than once ("copying"), exactly once, or not at all ("deletion"). Thus a configuration may contain several calls with the same subtree as parameter.

We now define a few basic concepts associated with the number of translations made of subtrees of an input tree, cf. [19]. Consider an input tree t, a subtree

s of t, and a derivation $q_0(t) \overset{*}{\Rightarrow} t'$ of M on t with output tree t'. It should be clear from the above description of M as a system of recursive procedures that, during this derivation, s will appear as actual parameter to some number, say n, of calls. Consequently the derivation can be divided into a derivation on the part of t outside s and n derivations on s. The output tree t' contains n (disjoint) subtrees which are the translations of s.

More formally, let $u \in T_\Sigma[\{x_1\}]$ contain one occurrence of x_1 and let $t = u[s]$, i.e., u is the part of t outside s. Then there are derivations $q_0(u) \overset{*}{\Rightarrow} v$ (with $v \in T_\Delta[Q(x_1)]$) and $q_i(s) \overset{*}{\Rightarrow} s_i$ (with $s_i \in T_\Delta, 1 \leq i \leq n, n \geq 0$) such that the derivation $q_0(t) \overset{*}{\Rightarrow} t'$ can be reordered as

$$q_0(t) \overset{*}{\Rightarrow} v[s] = v_1 q_1(s) v_2 q_2(s) \ldots v_n q_n(s) v_{n+1} \overset{*}{\Rightarrow} v_1 s_1 v_2 s_2 \ldots v_n s_n v_{n+1} = t',$$

where $v = v_1 q_1(x_1) v_2 q_2(x_1) \ldots v_n q_n(x_1) v_{n+1}$ and v_i does not contain x_1. We note that these derivations are unique modulo reordering; we will not formalize this concept of reordering, it is similar to the one for derivations of context-free grammars; see the notion of computation tree in [45].

We say that, with respect to $q_0(t) \overset{*}{\Rightarrow} t', \langle q_1(s) \overset{*}{\Rightarrow} s_1, \ldots, q_n(s) \overset{*}{\Rightarrow} s_n \rangle$ is the *derivation-sequence* of s, $\langle q_1, \ldots, q_n \rangle$ is the *state-sequence* of s and $\langle r_1, \ldots, r_n \rangle$ is the *rule-sequence* of s, where r_i is the first rule applied in the derivation $q_i(s) \overset{*}{\Rightarrow} s_i$. We also say that these sequences are associated with the root of s. The following simple property is of importance: if we replace in $q_0(t) \overset{*}{\Rightarrow} t'$ the derivations $q_i(s) \overset{*}{\Rightarrow} s_i$ of the derivation-sequence of s by other derivations $q_i(s) \overset{*}{\Rightarrow} s_i'$ (with $s_i' \in T_\Delta$), then we obtain another valid derivation $q_0(t) \overset{*}{\Rightarrow} t'' = v_1 s_1' v_2 s_2' \ldots v_n s_n' v_{n+1}$ such that the rule-sequences of all nodes of u (the part of t outside s) remain unaltered.

In general, for a given top-down tree transducer, the length of the derivation-sequences in its derivations (i.e., the number of calls with a subtree as actual parameter, i.e., the number of copies made of an input subtree) is unbounded. We now define those transducers for which a bound on this number exists. They will play an essential intermediate role in the proof of the properness of the tree transducer hierarchy. For $k \geq 1$, a derivation $q_0(t) \overset{*}{\Rightarrow} t'$ of a top-down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$ is *k-copying* if for each subtree s of t the length of the derivation-sequence of s is at most k. M is k-copying if each of its derivations $q_0(t) \overset{*}{\Rightarrow} t'$ $(t \in T_\Sigma, t' \in T_\Delta)$ is k-copying. M is *finite copying* if it is k-copying for some $k \geq 1$. M is *linear* if it is 1-copying. The class of finite copying (k-copying) top-down tree translations is denoted by T_{fc} $(T_{fc(k)})$, and DT_{fc} $(DT_{fc(k)})$ in the deterministic case. We note that there is no difference between $T_{fc}(K)$ and $DT_{fc}(K)$ under weak conditions on K (see next subsection). Consequently, we will never consider $T_{fc}(K)$.

The state-sequences of a finite copying tree transducer can be computed by a finite tree automaton, cf. [19]. This is shown in the following lemma.

Lemma 2.1. *Let $M \in DT_{fc}$. There exists a deterministic finite tree automaton A which has the state-sequences of M as states, and has the following property: for every node d of an input tree, A arrives at d in state q, where q is the state-sequence of M at d. Formally, let $t = u_1 s u_2$ with $t, s \in T_\Sigma$, where Σ is the input alphabet of M*

(and A). Then $q_0(t) \overset{*}{\Rightarrow} v_1q_1(s)v_2q_2(s)\ldots v_nq_n(s)v_{n+1}$ in M if and only if $\langle q_0\rangle(t) \overset{*}{\Rightarrow} u_1z(s)u_2$ in A, where $z=\langle q_1, q_2, \ldots, q_n\rangle$.

Proof. Let $M=(Q, \Sigma, \Delta, q_0, R)$ be in $DT_{fc(k)}$. Note that, by the determinism of M, each subtree of an input tree has at most one state-sequence. Define the finite tree automaton $A=(Q_A, \Sigma, \Sigma, \langle q_0\rangle, R_A)$ where Q_A is the set of all sequences of states of M of length at most k, i.e., $Q_A=\cup\{Q^n|0\leq n\leq k\}$, and R_A is constructed as follows. Consider $\sigma\in\Sigma_m$ and $z=\langle q_1,\ldots,q_n\rangle\in Q^n$, $n\leq k$. Let $q_j(\sigma(x_1\ldots x_m))\rightarrow u_j$ be a rule in R for every j, $1\leq j\leq n$. For $1\leq i\leq m$, let $z_i=\langle p_1,\ldots,p_s\rangle\in Q^s$ if $s\leq k$ and $p_1(x_i), p_2(x_i),\ldots, p_s(x_i)$ occur in this order in the string $u_1u_2\ldots u_n$ (and there are no other occurrences of x_i in $u_1u_2\ldots u_n$). Then the rule $z(\sigma(x_1\ldots x_m))\rightarrow \sigma(z_1(x_1)\ldots z_m(x_m))$ is in R_A. In case not all u_j exist or $s>k$ for some i, there is no rule with left-hand side $z(\sigma(x_1\ldots x_m))$ in R_A.

It should be clear that A has the required property: if z is the state-sequence of $\sigma(t_1\ldots t_m)$ then obviously z_i is the state-sequence of t_i. Note that, in particular, A and M have the same domain. □

We will need an easy technical result: a top-down tree transducer can be changed in such a way that it simultaneously simulates a finite tree automaton on its own output tree, cf. Lemma 2.10 of [17]. This is formalized in the following lemma.

Lemma 2.2. *Let $M=(Q, \Sigma, \Delta, q_0, R)$ be a top-down tree transducer and $A=(P, \Delta, \Delta, p_0, R_A)$ a finite tree automaton (working on M's output trees). Then there exists a top-down tree transducer $M'=(Q', \Sigma, \Delta, \langle q_0, p_0\rangle, R')$ with $Q'=Q\times P$, such that*

$$\langle q, p\rangle(t) \overset{*}{\Rightarrow} v_1\langle q_1, p_1\rangle(s_1)v_2\ldots v_n\langle q_n, p_n\rangle(s_n)v_{n+1} \text{ in } M'$$

if and only if

$$q(t) \overset{*}{\Rightarrow} v_1q_1(s_1)v_2\ldots v_nq_n(s_n)v_{n+1} \text{ in } M \text{ and}$$

$$p(v_1s_1'v_2\ldots v_ns_n'v_{n+1}) \overset{*}{\Rightarrow} v_1p_1(s_1')v_2\ldots v_np_n(s_n')v_{n+1} \text{ in } A$$

where $s_1,\ldots,s_n$ are subtrees of t and $s_1',\ldots,s_n'$ are arbitrary trees in T_Δ.

Proof. The rules of R' are constructed as follows. Let $q(\sigma(x_1\ldots x_k))\rightarrow t$ be a rule in R with $t\in T_\Delta[Q(X_k)]$. Let $p(t)\overset{*}{\Rightarrow}t_1$ be a derivation in A with $p\in P$ and $t_1\in T_\Delta[P(Q(X_k))]$, i.e. "$A$ is run on t as long as possible". Then the rule $\langle q, p\rangle(\sigma(x_1\ldots x_k))\rightarrow t_2$ is in R', where $t_2\in T_\Delta[Q'(X_k)]$ is the result of replacing each $p_1(q_1(x_i))$ by $\langle q_1, p_1\rangle(x_i)$ in t_1.

It should be clear that M' indeed simulates A on its own output trees (but note that in one step of M' several steps of A are simulated). In particular, M' produces only those output trees of M which are in the domain of A. □

2.3. *Operations and closure properties*

We assume the reader to be familar with the usual operations on languages, see [29, 23]. Recall that a *trio* is a class of languages closed under intersection with regular languages, inverse homomorphisms, and λ-free homomorphisms (which implies that it is closed under sequential machine mappings and λ-free regular substitution). A trio closed under union is called a semi-AFL. An AFL is a semi-AFL closed under concatenation and Kleene star. A hyper-AFL is an AFL closed under iterated substitution (see, e.g., [37, 6]). Whenever such a class is closed under arbitrary homomorphisms, the adjective "full" is added. Note that a full trio is also called a rational cone. Apart from the usual operations on languages we shall use the following *copy-operations* and a particular kind of regular substitution called *rub* (the *r*egular s*ub*stitution of *rub*bish).

For a language $L \subseteq \Sigma^*$ and a symbol $\$ \notin \Sigma$, we define

$$\begin{aligned} c_2(L) &= \{w\$w \mid w \in L\}, \\ c_*(L) &= \{(w\$)^n \mid w \in L, n \geq 1\} \text{ and} \\ c_{\exp}(L) &= \{(w\$)^{2^n} \mid w \in L, n \geq 0\}. \end{aligned}$$

The regular substitution rub is defined by $\text{rub}(a) = \$^*a\$^*$ for all $a \in \Sigma$. Note that rub is also an inverse homomorphism.

With respect to tree languages we shall use the operations of finite state relabeling (see Section 2.2) and of regular insertion (see [19]), to be defined next. The first generalizes the sequential machine (with accepting states) on strings. The second generalizes the notion of (λ-free) regular substitution.

Let Σ be a ranked alphabet and Δ an alphabet. A *regular insertion f* is given by a mapping f_k from Σ_k to languages over Δ, for each $k \geq 0$, such that, for each $\sigma \in \Sigma_k$, $f_k(\sigma) \in \text{REG}$, i.e., $f_k(\sigma)$ is a regular language over Δ.

The regular insertion f is a mapping from tree languages over Σ to tree languages over $\Sigma \cup \Delta$ (where the elements of Δ have rank 1) defined as follows; for $w \in \Delta^*$ and $t \in T_\Sigma$ we denote by $w(t)$ the tree t with the monadic tree w on top, i.e., $\lambda(t) = t$ and $aw(t) = a(w(t))$. If $w \in f_0(\sigma)$ then $w(\sigma) \in f(\sigma)$. If $s_i \in f(t_i)$ for $1 \leq i \leq k$ and $w \in f_k(\sigma)$, then $w(\sigma(s_1 \dots s_k)) \in f(\sigma(t_1 \dots t_k))$. If $L \subseteq T_\Sigma$ then $f(L) = \bigcup \{f(t) \mid t \in L\}$. Thus a regular insertion inserts (substitutes) strings from a regular language $f_k(\sigma)$ just above each node labeled σ.

The next concept is introduced for the sake of this paper only. A class of tree languages is a *tree trio* if it is closed under finite state relabelings and regular insertions. A monadic tree trio is not necessarily a trio, but every trio is a monadic tree trio.

If K is a tree trio, then $T_{fc(k)}(K) = DT_{fc(k)}(K)$ for every k; see Lemma 3.2.3 of [19]. If K is a tree trio, then we can do without the special symbol e (with yield λ) in all classes $yT(K)$, $yDT(K)$, $yDT_{fc}(K)$ and $y\text{HOM}(K)$, cf. Lemma 1.3 of [17].

Lemma 2.3. REC *is a tree trio.*

Proof. Closure of REC under finite state relabelings is easy to prove (REC is even closed under linear tree transducers [41]). To prove closure under regular insertion let A be a finite tree automaton recognizing the tree language $L \subseteq T_\Sigma$, and let f be a regular insertion with alphabet Δ. Since REC is closed under finite state relabeling we may assume Σ and Δ to be disjoint. It is now easy to see that $f(L)$ is the intersection of two recognizable tree languages. In fact, for any tree $s \in T_{\Sigma \cup \Delta}$, one finite tree automaton checks that the "Σ-part" of s belongs to L (simulating A and disregarding elements of Δ), and the other finite tree automaton checks that all "vertical Δ-strings" of s belong to the appropriate regular languages $f_k(\sigma)$ (simulating all the corresponding ordinary finite automata). Since REC is closed under intersection [41], this shows the lemma. □

The next lemmas state some relationships between operations on tree languages. Let K be a class of tree languages.

Lemma 2.4. *If K is a tree trio, then $T(K)$ is a tree trio.*

Proof. As far as finite state relabelings are concerned, $T(K)$ is even closed under linear top-down tree translations [17]. Closure under regular insertion can easily be proved as follows. Let $L \in K$, $M = (Q, \Sigma, \Delta, q_0, R)$ in T, and f a regular insertion on T_Δ. We have to show that $f(M(L)) \in T(K)$. Define a regular insertion g on T_Σ by

$$g_n(\sigma) = \bar{\sigma}(\cup \{\bar{\tau} f_k(\tau) \mid \tau \in \Delta_k, k \geq 0\})^* \quad \text{for each } \sigma \in \Sigma_n (n \geq 0).$$

It is straightforward to construct a new top-down tree transducer M' such that $M'(g(L)) = f(M(L))$. M' simulates M and moreover it selects a string of $f_k(\tau)$ from the input before it produces the output symbol τ. □

Lemma 2.5. *If K is a tree trio, then $yDT_{fc}(K)$ and $yT(K)$ are closed under regular substitution.*

Proof. Direct from Lemma 3.2.2. of [19], where it is shown that a generalized top-down tree transducer in which the rules are allowed to have regular languages as right-hand sides can be simulated by an ordinary top-down tree transducer. Thus, if $L = yM(L_1)$ with $L_1 \in K$ and $M \in T$, and the generalized top-down tree transducer M' is obtained from M by applying the regular substitution f to the right-hand sides of its rules, then $yM'(L_1) = f(L)$. □

Lemma 2.6. *If K is a tree trio, then $yDT_{fc}(K)$ is closed under c_2, $yDT(K)$ is closed under c_2, c_* and $c_{\exp'}$, and yHOM(K) is closed under c_2 and $c_{\exp}$.*

Proof. Note that, since K is a tree trio, we may assume that the root of the input tree of a transducer has a label different from all other labels in the tree. This allows us to insert a regular language above the root of the tree and insert nothing, i.e. $\{\lambda\}$, everywhere else.

Let $M \in DT$ and $L \in K$. We now consider three cases: c_2, c_*, and $c_{\exp}$.

For c_2, let L_1 be obtained from L by inserting a symbol $\sharp$ above the root of each tree of L, i.e., $L_1 = \sharp(L)$. Since K is closed under regular insertion, $L_1 \in K$. Construct from M a new transducer M' which, for an input tree $\sharp(t)$, first makes two copies of t (by the rule $q_0(\sharp(x_1)) \to \sharp(q_0(x_1)\$q_0(x_1))$, where q_0 is the initial state of M, and $\sharp$ has ranks 1 and 3), and then treats both copies of t in the same way as M. Clearly $yM'(L_1) = yM'(\sharp(L)) = c_2(yM(L))$. Furthermore $M' \in DT$, and if M is finite copying then so is M' (but the bound is doubled). If M is a tree homomorphism, then so is M'.

For c_*, insert the regular language $\sharp^*¢$ above the root of each tree of L and let $L_1 = \sharp^*¢(L)$ be the resulting language (again $L_1 \in K$). Extend M to M' by the rules

$$q_0'(\sharp(x_1)) \to \sharp(q_1(x_1)\$q_0'(x_1)),$$
$$q_0'(¢(x_1)) \to ¢(q_0(x_1)\$),$$
$$q_1(\sharp(x_1)) \to q_1(x_1) \text{ and } q_1(¢(x_1)) \to q_0(x_1),$$

where q_0' is the new and q_0 the old initial state, q_1 is a new state, and $¢$ has ranks 1 and 2. Clearly $M' \in DT$ and $yM'(L_1) = yM'(\sharp^*¢(L)) = c_*(yM(L))$.

For $c_{\exp}$, insert $¢\sharp^*$ at the roots; add the rules $q_0(¢(x_1)) \to ¢(q_0(x_1)\$)$ and $q_0(\sharp(x_1)) \to \sharp(q_0(x_1)\$q_0(x_1))$. The resulting M' is in DT, and if $M \in \mathrm{HOM}$ then $M' \in \mathrm{HOM}$. Clearly $yM'(¢\sharp^*(L)) = c_{\exp}(yM(L))$. This proves the lemma. □

3. The Tree Transducer Hierarchy

In this section we establish some general bridge theorems for classes of tree transformation languages which together lead to the properness of the hierarchy $\{yT^n(\mathrm{REC})\}$. Due to their general character, we even obtain the uniform result that, for an arbitrary tree trio K, if $yT(K) \subsetneq yT^2(K)$, then the hierarchy $\{yT^n(K)\}$ is proper. In other words if, for some input tree trio, two tree transducers are more powerful than one, then $n+1$ are more powerful than n. The (sharper) main result of this section is stated in Theorem 3.12. The same bridge theorems will also be used in the next section to obtain two other hierarchies of transducers. At the end of this section we provide some concrete examples of languages not in $yT^n(\mathrm{REC})$ for any n.

We start with the following bridge theorem from [19] which intuitively says that a deterministic transducer which fully uses its copying facility (i.e. is not finite copying) cannot simultaneously handle nondeterminism. The second statement of the theorem is shown also in Theorem 16 of [46]. In Theorem 15 of [46] it is shown for tree languages (i.e. without "y" and with a tree variant of rub), based on the techniques of [32]. The first statement of the theorem was discovered independently by Latteux [31] for the case $K = \mathrm{REG}$.

Theorem 3.1 [19]. *Let K be a tree trio and L a (string) language.*
If $\mathrm{rub}(L) \in yDT(K)$, then $L \in yDT_{fc}(K)$.
If $\mathrm{rub}(L) \in y\mathrm{HOM}(K)$, then $L \in yDT_{fc(1)}(K)$.

Proof. The proof is given in Theorem 3.2.14 of [19]. It is based on the fact that the form of the language $\text{rub}(L)$ forces the involved tree transducer to have many derivations which are finite copying as far as the symbols from L are concerned, i.e., unbounded copying is used only to produce the \$ symbols.

Note that $y\text{HOM}(K) \subseteq yDT_{(1)}(K)$ in the notation of [19]. The classes are even equal. □

We note that, to obtain a language in $yT(\text{REC}) - yDT(\text{REC})$, Perrault [33] used $f(L)$, where f is a finite substitution and L an exponential language (i.e. its strings are of exponential length). Exponentiality forced L outside $yDT_{fc}(\text{REC})$ and the finite substitution forced L outside $yDT(\text{REC})$. The same method was used in [11] for the monadic case.

Since $yDT_{fc}(K)$, $yT(K)$, and $yDT_{fc}(T(K))$ are all closed under regular substitution by Lemmas 2.4 and 2.5, we obtain from Theorem 3.1 the following three corollaries. The second corollary is one of the steps to be used to show that the inclusions in the tree transducer hierarchy are proper.

Corollary 3.2. *If K is a tree trio, then $yDT_{fc}(K)$ is the largest subclass of $yDT(K)$ closed under regular substitution.*

Proof. Let K_0 be a subclass of $yDT(K)$ closed under regular substitution, and let $L \in K_0$. Then also $\text{rub}(L) \in K_0$ and hence $\text{rub}(L) \in yDT(K)$. By Theorem 3.1, $L \in yDT_{fc}(K)$. Hence $K_0 \subseteq yDT_{fc}(K)$. □

Corollary 3.3. *Let K be a tree trio and L a (string) language. If $L \in yDT(K) - yDT_{fc}(K)$, then* $\text{rub}(L) \in yT(K) - yDT(K)$. □

Corollary 3.4. *Let K be a tree trio, let $K_1 = T(K)$, and let L be a (string) language. If $L \in yDT_{fc}(K_1) - yK_1$, then* $\text{rub}(L) \in yDT_{fc}(K_1) - y\text{HOM}(K_1)$.

Proof. Use the second part of Theorem 3.1 and note that if K is a tree trio, then $T(K)$ is closed under linear tree transducers [17], i.e., $DT_{fc(1)}(K_1) = K_1$. □

Note that Theorem 3.1 and Corollary 3.3 illustrate the discussion on bridge theorems in the introduction, with $K_1 = yDT_{fc}(K)$, $K_2 = yDT(K)$, $K_3 = yT(K)$, and $f = \text{rub}$.

The next bridge theorem (taken from [21], see also [38]) will show that an essentially nondeterministic transducer cannot do copying of strings; more precisely, if $c_2(L) \in yT(K)$ then $L \in yDT(K)$. Although this theorem is not needed to obtain properness of the tree transducer hierarchy $\{yT^n(K)\}$, it is useful to establish a finer hierarchy and, moreover, it illustrates a general method to be used in other cases. Before presenting this theorem we need some more terminology, together with a lemma. The theorem is concerned with showing that, under certain circumstances, a nondeterministic transducer M can be replaced by a deterministic transducer. This is true in particular if M has sufficiently many "uniform" derivations (this notion was introduced in [33]). A derivation of a transducer is uniform if at each node the transducer applies the same rule when arriving in the same state at that node. Formally we define this as follows.

Let $M=(Q,\Sigma,\Delta,q_0,R)$ be a top-down tree transducer. Let $q_0(t)\overset{*}{\Rightarrow}t'$ be a derivation of $M(t\in T_\Sigma, t'\in T_\Delta)$, let s be a subtree of t, and let $\langle q_1,\ldots,q_n\rangle$ and $\langle r_1,\ldots,r_n\rangle$ be the state-sequence and rule-sequence of s respectively. The derivation $q_0(t)\overset{*}{\Rightarrow}t'$ is *uniform at* the root of s if, for all $1\le i, j\le n, q_i=q_j$ implies $r_i=r_j$. Note that, since the state-sequence is determined by the rule-sequence, uniformity is a property of the rule-sequence alone. The derivation $q_0(t)\overset{*}{\Rightarrow}t'$ is *uniform* if it is uniform at all nodes of t. For a tree language $L\subseteq T_\Sigma$, we denote by $M_{un}(L)$ the tree language $\{t'\in T_\Delta \mid$ there is a uniform derivation $q_0(t)\overset{*}{\Rightarrow}t'$ of M for some $t\in L\}$.

Lemma 3.5. *Let K be a tree trio, $L\in K$ and M a top-down tree transducer. Then $M_{un}(L)\in DT(K)$.*

Proof. We use a finite state relabeling to "guess" the rule applied by M at each node of the input tree. Let $M=(Q,\Sigma,\Delta,q_0,R)$. Let A be a (nondeterministic) finite state relabeling which adds to each label $\sigma\in\Sigma_k$ of a node of an input tree a mapping $f\colon Q\to R$, such that the left-hand side of $f(q)$ is $q(\sigma(x_1\ldots x_k))$ for all $q\in Q$. Construct M' such that when arriving in state q at a node labeled $\langle\sigma,f\rangle$, it applies rule $f(q)$, more precisely: M' applies the rules obtained from $f(q)$ by replacing its left-hand side $q(\sigma(x_1\ldots x_k))$ by $q(\langle\sigma,f\rangle(x_1\ldots x_k))$. Clearly M' is deterministic and simulates exactly all uniform derivations of M (due to the fact that it applies the same rule when arriving in the same state at some input node). Hence $M'(A(L))=M_{un}(L)$ which proves the lemma. □

We now continue with the second bridge theorem. We note here that copying was also used in [32] and [33] to obtain languages outside $T(\mathrm{REC})$ and $yT(\mathrm{REC})$, respectively.

Theorem 3.6 [21]. *Let K be a tree trio and L a (string) language. If $c_2(L)\in yT(K)$, then $L\in yDT(K)$.*

Proof. Let $c_2(L)=yM(L_1)$ with $L_1\in K$ and $M=(Q,\Sigma,\Delta,q_0,R)$ a top-down tree transducer. We will show that the form of the language $c_2(L)$ forces a large degree of uniformness on M; in fact we will show that $c_2(L)=yM_{un}(L_1)$; from this it follows by Lemma 3.5 that $c_2(L)\in yDT(K)$, and hence $L\in yDT(K)$ because $yDT(K)$ is closed under deterministic gsm mappings [17]. Consider a derivation $q_0(t)\overset{*}{\Rightarrow}t'$ of M such that $\mathrm{yield}(t')=w\$w$ with $w\in L$, and let s be a subtree of t with derivation-sequence $\langle q_1(s)\overset{*}{\Rightarrow}s_1,\ldots,q_n(s)\overset{*}{\Rightarrow}s_n\rangle$. Suppose that $q_i=q_j$ with $i<j$. Then $w\$w=xuyvz$ with $u=\mathrm{yield}(s_i)$ and $v=\mathrm{yield}(s_j)$. Since $q_i=q_j$, also $xuyuz$ and $xvyvz$ are in $yM(L_1)=c_2(L)$ by replacing $q_j(s)\overset{*}{\Rightarrow}s_j$ by $q_i(s)\overset{*}{\Rightarrow}s_i$ in the given derivation and vice versa. Due to the form of the string $w\$w$ this implies that $u=v$, i.e. $\mathrm{yield}(s_i)=\mathrm{yield}(s_j)$. Hence we can change, say $q_j(s)\overset{*}{\Rightarrow}s_j$ into $q_i(s)\overset{*}{\Rightarrow}s_i$ without changing the yield of the output tree. By repeating this process we can change the derivation into one which is uniform at the root of s without changing the yield of the output tree. We now apply this procedure to all nodes of the input tree t in a top-down fashion, i.e. first to the

root of t, then to its sons (in some arbitrary order), then to their sons, etc. In this way we change the derivation (without changing the yield of its output tree) into a uniform derivation.

Note that if the procedure is applied to some node, then the rule-sequences of all nodes which are not descendants of this node remain unaltered, cf. the "simple property" of rule-sequences in Section 2.2. Hence $w\$w \in yM_{un}(L_1)$. And so $c_2(L) = yM_{un}(L_1)$ and the theorem is proved. □

Since $yDT(K)$, $y\text{HOM}(T(K))$ and $yDT_{fc}(T(K))$ are all closed under c_2 (Lemmas 2.4 and 2.6), we obtain from Theorem 3.6 the following corollaries of which the second will be used to show that some of the inclusions in the tree transducer hierarchy are proper.

Corollary 3.7. *If K is a tree trio, then $yDT(K)$ is the largest subclass of $yT(K)$ closed under copying (c_2).*

Corollary 3.8. *Let K be a tree trio and L a language. If $L \in yT(K) - yDT(K)$, then $c_2(L) \in yDT_{fc}(T(K)) - yT(K)$ and $c_2(L) \in y\text{HOM}(T(K)) - yT(K)$.*

Corollaries 3.3 and 3.8 do not yet provide us with enough bridges to step from one bridge to the next. A bridge is missing from $yDT_{fc}(K) - yK$ to $yDT(K) - yDT_{fc}(K)$. The existence of such a bridge remains open. However, we can establish such a bridge if $K = T(K_1)$ for some tree trio K_1 and this suffices for our purposes. The involved operation is unbounded copying ($c_{\exp}$ or c_*). Thus the next theorem expresses the intuitively obvious fact that a finite copying transducer cannot do infinite copying (of an essentially nondeterministic language). It is the key result of this paper.

Theorem 3.9. *Let K be a tree trio and L a (string) language. Let $L_0 \subseteq c_*(L)$ have the property that for each $w \in L$ there are infinitely many n such that $(w\$)^n \in L_0$. If $L_0 \in yDT_{fc}(T(K))$, then $L \in yDT(K)$.*

Proof. The idea of the proof is similar to that of Theorem 3.6. Let $L_0 = yN(M(L_1))$ with $L_1 \in K$, $M \in T$ and $N \in DT_{fc(k)}$ for some $k \geq 1$. Roughly we will show that the form of the language L_0 forces M to have sufficiently many uniform derivations to obtain a subset L_0' of L_0 which still contains some $(w\$)^n$ for each $w \in L$ (in other words, the nondeterminism of M is used only to iterate $w\$$, not to obtain w itself). Consequently M can be replaced by a deterministic transducer (Lemma 3.5), $L_0' \in yDT_{fc}(DT(K))$, and hence $L_0' \in yDT(K)$ because $DT(K)$ is closed under DT, see [17]; since $yDT(K)$ is closed under deterministic gsm mappings [17], it then follows that $L \in yDT(K)$.

For technical reasons, but also to increase the transparency of the demonstration, we first change M in such a way that in its finite control it keeps track of the (bounded) state-sequences of N on the output tree of M. By Lemma 2.1 the state-sequences of N can be computed by a finite tree automaton A, and by Lemma 2.2 M can be changed in such a way that it additionally simulates A on its output trees. Let $M = (Q, \Sigma, \Delta, q_0, R_1)$ and $N = (Q_2, \Delta, \Omega, q_0^N, R_2)$, and let Z denote the (finite) set of all possible state-sequences of N, i.e. $Z = \cup \{Q_2^n \mid 0 \leq n \leq k\}$.

We obtain a top-down tree transducer $M' = (Q_1 \times Z, \Sigma; \Delta, q_0', R_1')$ with $q_0' = \langle q_0, \langle q_0^N \rangle \rangle$, such that if $q_0(t) \overset{*}{\Rightarrow} v_1 q(s) v_2 \overset{*}{\Rightarrow} v_1 s' v_2 = t'$ in M, then $q_0'(t) \overset{*}{\Rightarrow} v_1 \langle q, z \rangle (s) v_2 \overset{*}{\Rightarrow} v_1 s' v_2 = t'$ in M', where z is the state-sequence of s' with respect to N (note that because N is deterministic the state-sequences of N are unique). Note that M' produces only output trees which are in the domain of N.

Thus $L_0 = yN(M'(L_1))$. We now want to show that $yN(M'_{un}(L_1))$ contains some $(w\$)^n$ for each $w \in L$, which proves the theorem by Lemma 3.5 and the above arguments (set $L_0' = yN(M'_{un}(L_1))$).

For arbitrary $w \in L$, consider $(w\$)^n$ in L_0 such that $n \geq 2k+2$ (the reason for this number $2k+2$ will become clear later). Let $q_0'(t_1) \overset{*}{\Rightarrow} t_2$ and $q_0^N(t_2) \overset{*}{\Rightarrow} t_3$ be two derivations in M' and N respectively, such that $t_1 \in L_1$ and $\text{yield}(t_3) = (w\$)^n$, see Figure 1.

We want to change the first derivation into a uniform one in such a way that the resulting (changed) t_3 has a yield which still "contains the same w". Let s be an arbitrary subtree of t_1 and let us try to make the first derivation uniform at the root of s. Let s_1 and s_2 be two different $\langle q, z \rangle$-translations of s, where $\langle q, z \rangle$ is some state of M'. Thus derivations $\langle q, z \rangle (s) \overset{*}{\Rightarrow} s_1$ and $\langle q, z \rangle (s) \overset{*}{\Rightarrow} s_2$ occur in the derivation-sequence of s (with respect to $q_0'(t_1) \overset{*}{\Rightarrow} t_2$). As one step toward uniformness at the root of s we show how to replace one of these derivations by the other. If $z = \langle p_1, \ldots, p_m \rangle$ with $m \leq k$, then s_1 and s_2 both have state-sequence $\langle p_1, \ldots, p_m \rangle$ in the derivation $q_0^N(t_2) \overset{*}{\Rightarrow} t_3$. Thus t_3 has (disjoint) subtrees $t(s_i, p_h)$ such that $p_h(s_i) \overset{*}{\underset{N}{\Rightarrow}} t(s_i, p_h)$ for $1 \leq i \leq 2$ and $1 \leq h \leq m$: these are the translations

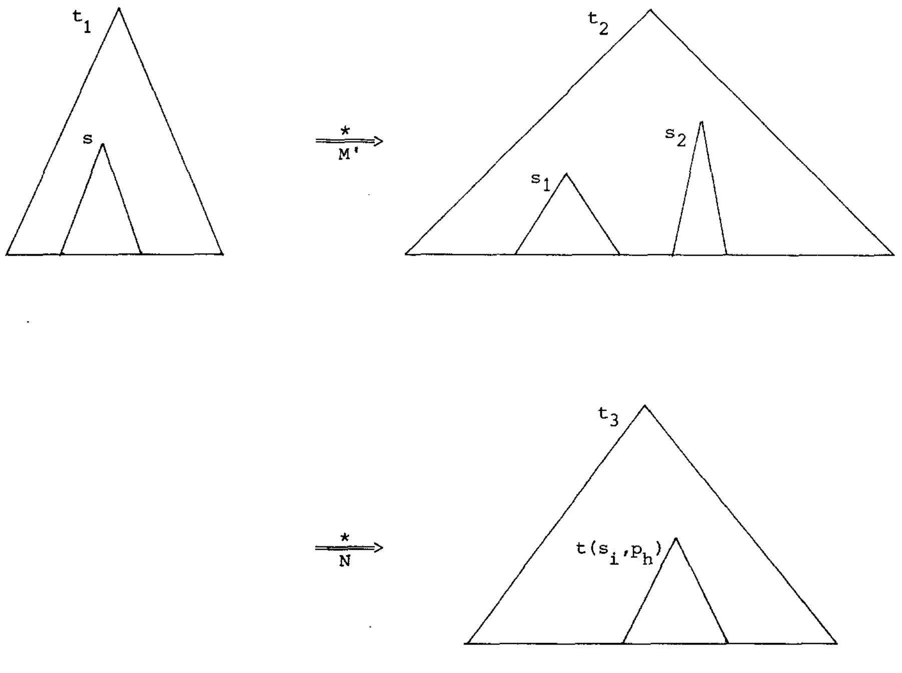
t_1
s
$\overset{*}{\underset{M'}{\Longrightarrow}}$
t_2
s_1
s_2
$\overset{*}{\underset{N}{\Longrightarrow}}$
t_3
$t(s_i, p_h)$

Fig. 1. A two-stage derivation.

of the trees s_i by N. Note that a change of a subderivation $\langle q, z\rangle(s) \overset{*}{\Rightarrow} s_i$ into $\langle q, z\rangle(s) \overset{*}{\Rightarrow} s_j$ in the first derivation results in a corresponding change in N's derivation, and one in t_3, consisting of the change of $t(s_i, p_h)$ into $t(s_j, p_h)$ for each h, $1 \leq h \leq m$. Note also that the strings $yt(s_i, p_h)$ are disjoint substrings of $yt_3 = (w\$)^n$.

For $1 \leq i \leq 2$, let num(i) denote the number of occurrences of \$ in the string $yt(s_i, p_1).yt(s_i, p_2)\ldots yt(s_i, p_m)$. Assume without loss of generality that num(2) $\leq$ num(1). We now replace $\langle q, z\rangle(s) \overset{*}{\Rightarrow} s_2$ by $\langle q, z\rangle(s) \overset{*}{\Rightarrow} s_1$ in the first derivation. Let this replacement change t_3 into t_3'. To show that yield(t_3') still "contains the same w" we consider the following two cases.

Case 1. There exists h such that $yt(s_1, p_h)$ contains at least two occurrences of \$. Then it contains a substring $\$w\$$. Since this occurrence of $\$w\$$ is not affected by changing s_2 into s_1, yield(t_3')$=(w\$)^{n_1}$ for some n_1. Moreover, since num(2)$\leq$ num(1), $n_1 \geq n$ and so $n_1 \geq 2k+2$.

Case 2. For all h, $yt(s_1, p_h)$ contains at most one occurrence of \$. Then num(1)$\leq m \leq k$ and hence also num(2)$\leq m \leq k$. Consequently the strings $yt(s_2, p_h)$ overlap with at most $2m$ occurrences of $\$w\$$ in yield(t_3). Therefore, since $n \geq 2k+2 \geq 2m+2$, there is an occurrence of $\$w\$$ which lies outside these $yt(s_2, p_h)$. Since this "outer" $\$w\$$ is not affected by changing $yt(s_2, p_h)$ into $yt(s_1, p_h)$, yield(t_3')$=(w\$)^{n_1}$ for some n_1. Moreover, as in Case 1, $n_1 \geq n \geq 2k+2$.

Note that in Case 2 essential use is made of the finite copying of N, i.e., of the assumption that $n \geq 2k+2$. We have been careful to see to it that the replacement of s_2 by s_1 results in some n_1 such that again $n_1 \geq 2k+2$. This ensures that the replacement process (and Cases 1 and 2) can be repeated to obtain a derivation which is uniform at the root of s. (Note that if s_3 is another $\langle q, z\rangle$-translation of s and num(3)$\geq$num(1), then both occurrences of s_1, obtained after replacing s_2 by s_1, have to be replaced by s_3; if num(3)$\leq$num(1), then s_3 is replaced by s_1). This procedure can now also be applied to all nodes of the tree t_1 in the usual top-down fashion. This leads to a uniform derivation $q_0'(t_1) \overset{*}{\Rightarrow} t_2'$ in M' and a derivation $q_0^N(t_2') \overset{*}{\Rightarrow} t_3'$ in N such that yield(t_3')$=(w\$)^{n_1}$ for some $n_1 \geq n$. Hence $(w\$)^{n_1} \in yN(M_{un}'(L_1))$. Since w was arbitrary this proves that $yN(M_{un}'(L_1))$ contains some $(w\$)^{n_1}$ for each $w \in L$ (even infinitely many), and the theorem is proved. □

Note that Theorem 3.9 holds in particular for $L_0 = c_{\exp}(L)$ and $L_0 = c_*(L)$. From Lemma 2.4 we again obtain two corollaries of which the second is the important one.

Corollary 3.10. *If K is a tree trio, then $yDT(K)$ is the largest subclass of $yDT_{fc}(T(K))$ closed under unbounded copying ($c_{\exp}, c_*$).*

Corollary 3.11. *Let K be a tree trio and L a language. If $L \in yT(K) - yDT(K)$, then $c_{\exp}(L) \in y\mathrm{HOM}(T(K)) - yDT_{fc}(T(K))$ and $c_*(L) \in yDT(T(K)) - yDT_{fc}(T(K))$.*

Corollaries 3.3, 3.8 and 3.11 lead to the main result of this section. For the notion of a correct diagram see Section 2.

Theorem 3.12. *Let K be a tree trio. If $yDT_{fc}(K) \subsetneq yT(K)$, then for all $n \geq 1$ the following diagram is correct, where K_n denotes $T^n(K)$, see Figure 2.*

More generally, if $yDT_{fc}(K) \subsetneq yT^m(K)$ for some $m \geq 1$, then this diagram is correct for $n \geq m$.

Proof. By Lemma 2.4 K_n is a tree trio for all n. Let L_n be a language in $yK_n - yDT(K_{n-1})$. Note that for $n=1$ the existence of L_n follows from the hypothesis of the theorem together with Corollary 3.3.

Then $c_{\text{exp}}(L_n) \in y\text{HOM}(K_n) - yDT_{fc}(K_n)$	by Corollary 3.11
$\text{rub}(c_2(L_n)) \in yDT_{fc}(K_n) - y\text{HOM}(K_n)$	by Corollaries 3.8 and 3.4
$\text{rub}(c_{\text{exp}}(L_n)) \in yT(K_n) - yDT(K_n)$	by Corollary 3.3.

This proves the first part of the theorem. Note also that

$c_2(L_n) \in (y\text{HOM}(K_n) \cap yDT_{fc}(K_n)) - yK_N$	by Corollary 3.8
$c_*(L_n) \in yDT(K_n) - yDT_{fc}(K_n)$	by Corollary 3.11
$\text{rub}(c_*(L_n)) \in yT(K_n) - yDT(K_n)$	by Corollary 3.3.

For the second part of the theorem we observe that if $L \in yDT(K_{m-1}) - yK_{m-1}$, then $\text{rub}(c_*(L)) \in yT(K_{m-1}) - yDT(K_{m-1})$ by Lemmas 2.5 and 2.6 and Theorems

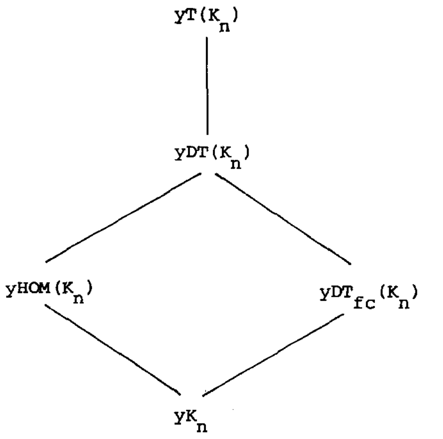

Fig. 2. The tree-transducer hierarchy.

3.1 and 3.9. Hence a language $L_m \in yK_m - yDT(K_{m-1})$ has to exist, and the argument is the same as above. □

The languages which prove the hierarchy to be proper are over an increasingly large alphabet (because with each application of an operation rub, c_2 or $c_{\exp}$ a new symbol \$ has to be taken). However, since $yDT(K)$ is closed under deterministic gsm mappings [17], it is easy to see that examples of such languages exist over a fixed alphabet (of, say, 4 symbols).

Remark. We have done our best to add $y\text{HOM}(T^n(K))$ to the diagram of Figure 2, because of the following ([46], see also [16]). If, in Theorem 3.12, K is also closed under linear tree transducers (equivalently, under linear tree homomorphisms), then, for all $n \geq 0$, $y\text{HOM}(T^n(K)) = yB^{n+1}(K)$ where B is the class of bottom-up tree transducers. This is in particular true for $K = \text{REC}$. The top-down/bottom-up hierarchy obtained from Figure 2 by disregarding $yDT(K_n)$ and $yDT_{fc}(K_n)$ was considered in [46]. We also note here that, as remarked in [46], under the conditions of Theorem 3.12 $y\text{HOM}(T^n(K))$ is not closed under inverse homomorphisms (and hence is not a trio) because $c_{\exp}(L_n) \in y\text{HOM}(T^n(K))$ but $\text{rub}(c_{\exp}(L_n)) \notin y\text{HOM}(T^n(K))$, cf. the proof of Theorem 3.12.

A weaker but intuitively attractive consequence of the above theorem is that if 2 transducers are more powerful than 1, then $n+1$ transducers are more powerful than n.

Corollary 3.13. *Let K be a tree trio. If $yT(K) \subsetneq yT^2(K)$, then $yT^n(K) \subsetneq yT^{n+1}(K)$ for all $n \geq 1$.*

We now turn to the tree transducer hierarchy which starts with the recognizable tree languages.

Theorem 3.14. *The tree transducer hierarchy $\{yT^n(\text{REC})\}$ is proper. More precisely, for $K_n = T^n(\text{REC})$ and $n \geq 1$ the diagram of Figure 2 is correct.*

Proof. Since by Lemma 2.3 REC is a tree trio, Theorem 3.12 is applicable to $K = \text{REC}$. It was shown in Corollary 3.2.7 of [19] that all languages in $yDT_{fc}(\text{REC})$ have semi-linear Parikh-sets. Hence languages such as $\{a^{2^n} \mid n \geq 0\}$ and $\{a^n \mid n$ is not prime$\}$ are in $yDT(\text{REC})$ but not in $yDT_{fc}(\text{REC})$. Note that $\{a^{2^n} \mid n \geq 0\} = h(c_{\exp}(\{a\}))$ and $\{a^n \mid n$ is not prime$\} = h(c_+(aaa^*))$, where h is the homomorphism erasing \$ and $c_+(L) = \{(w\$)^n \mid w \in L, n \geq 2\}$. □

Note that properness of the tree transducer hierarchy has implications with respect to closure properties of its classes; see Corollaries 3.2, 3.7 and 3.10. In particular, by Corollary 3.2, $yDT(T^n(\text{REC}))$ is not closed under regular substitution and hence is not a trio.

It follows trivially from Theorem 3.14 that the hierarchy $\{T^n(\text{REC})\}$ of classes of tree languages and the hierarchy $\{T^n\}$ of classes of tree translations are also proper (more precisely, the analogues of the diagram of Fig. 2 are correct). See [46] for some consequences with respect to closure properties of the $T^n(REC)$

hierarchy. We note that although the hierarchy $\{T^n\}$ is proper, all counterexamples have to be relations which are not partial functions. It can be shown that the class $T^n \cap \{f \mid f \text{ is a partial function}\}$ is contained in T^2; in fact this class is equal to the class of tree translations which can be realized by deterministic top-down tree transducers with regular look-ahead [18].

The situation with respect to the hierarchy $\{T^n(K)\}$ for an arbitrary tree trio K is not immediately clear from our results so far. It is however not difficult, using the same methods as for the $yT^n(K)$ hierarchy and the obvious analogues of the operations rub, c_2, c_{exp} and c_* for trees, to obtain similar results for $\{T^n(K)\}$. The detailed proofs, which are easier than in the y-case, are left to the reader.

Theorem 3.15. *Let K be a tree trio. If $DT_{fc}(K) \subsetneq T(K)$, then the hierarchy $\{T^n(K)\}$ is proper (more precisely, the diagram of Figure 2, with "y" deleted, is correct).*

Proof. The analogue of Theorem 3.1 can be shown using for rub the regular insertion which inserts $\bar{c}\$^*$ above each symbol c of rank 0. The analogue of the second statement of Theorem 3.1 was shown in this way in [46].

The analogue of Theorem 3.6 can be proved using the tree operation $c_2(L)=\{\$(tt) \mid t \in L\}$, where \$ has rank 2, and the analogue of Theorem 3.9 for c_{exp} and c_* can be shown using $c_{\text{exp}}(L)=\{c_2^n(t) \mid n \geq 0, t \in L\}$ and $c_*(L)=\{f_t^n(t) \mid n \geq 0, t \in L\}$, where, for every tree s, $f_t(s)=\$(ts)$; cf. the proof of Lemma 2.6.

We note here that, in this "tree notation", the tree language used in [32] to show that $T(\text{REC}) \subsetneq T^2(\text{REC})$ can be written as $c_2(\text{rub}(c_*(b^*\$)))$, where $b^*\$$ is a monadic tree language. □

In fact, we do not know any tree trio K to which Theorem 3.15 is applicable (i.e., $DT_{fc}(K) \subsetneq T(K)$), but Theorem 3.12 is not (i.e., $yDT_{fc}(K)=yT(K)$).

If $DT(K)=T(K)$ then the whole hierarchy $\{T^n(K)\}$ collapses to $DT(K)$: $T^2(K)=T(DT(K)) \subseteq T(K)$, cf. [17]. Hence the hierarchy $\{T^n(K)\}$ is proper if and only if nondeterminism does pay, i.e., $DT(K) \subsetneq T(K)$. It is not clear whether a similar result holds for $\{yT^n(K)\}$.

It was shown in [45] that $yT^\infty(\text{REC})$ is properly contained in the class of context-sensitive languages. We close this section by providing some concrete examples of context-sensitive languages outside the hierarchy $yT^\infty(\text{REC})$ (recall that $yT^\infty(\text{REC})$ is the union of all classes $yT^n(\text{REC})$ of the tree transducer hierarchy).

Theorem 3.16. *Let f be a partial injective function of positive integers with an infinite domain. Then $\{(a^n\$)^{f(n)} \mid n \geq 1, f(n) \text{ defined}\} \notin yT^\infty(\text{REC})$.*

Proof. Let $L_0=\{(a^n\$)^{f(n)} \mid n \geq 1, f(n) \text{ is defined}\}$. We shall first prove the bridge result that for a tree trio K

$$(*) \quad \text{if } L_0 \in yDT(T(K)), \text{ then } L_0 \in yDT(K),$$

and then show that $L_0 \notin yDT(\mathrm{REC})$. Together this clearly proves the theorem: if $L_0 \in T^n(\mathrm{REC})$, then, trivially, $L_0 \in yDT(T^n(\mathrm{REC}))$ and so, by repeated application of $(*)$, $L_0 \in yDT(\mathrm{REC})$.

The proof of $(*)$ uses the same technique as that of Theorem 3.9. Let $L_0 = yN(M(L_1))$ with $L_1 \in K$, $M \in T$ and $N \in DT$. We will construct M' such that $L_0 = yN(M'_{un}(L_1))$. By Lemma 3.5 and the closure of $DT(K)$ under DT [17] this implies that $L_0 \in yDT(K)$. As in the proof of Theorem 3.9, M' is equivalent to M but keeps some more information in its finite control. First of all M' keeps track of the state-sets of N on the output tree of M, where the state-set of a subtree is the set of states occurring in its state-sequence, cf. [19]. This can be done as in the proof of Theorem 3.9 (using an obvious analogue of Lemma 2.1). Secondly, when arriving at the root of an input subtree s in state q and with state-set z of N, M' will predict for each $p \in z$ whether the string w_p, such that $q(s) \overset{*}{\underset{M}{\Rightarrow}} s'$ and $p(s') \overset{*}{\underset{N}{\Rightarrow}} s''$ and $\mathrm{yield}(s'') = w_p$, will contain 0, 1 or ≥ 2 occurrences of the symbol \$ (where $q(s) \overset{*}{\underset{M}{\Rightarrow}} s'$ is the derivation which M will follow from this point onwards). This can be done by another application of Lemma 2.2. In fact, for each p there exists a finite tree automaton A_p which, on a tree s', checks whether w_p satisfies the properties above (this is because if $N \in T$ and $L \in \mathrm{REG}$ then $(yN)^{-1}(L) \in \mathrm{REC}$, cf. [32, 17]; take $L = \Gamma^*$ or $\Gamma^*\$\Gamma^*$ or $\Omega_0^*\$\Omega_0^*\$\Omega_0^*$, where Ω is the output alphabet of N and $\Gamma = \Omega_0 - \{\$\}$). Combining these automata A_p into one finite tree automaton A, M' can simulate A on its output trees by Lemma 2.2. Thus, the states of M' have the form $\langle q, z, g\rangle$ where q is a state of M, z is a state-set of N and g is a mapping $g: z \to \{0, 1, \geq 2\}$ with the meaning described above. Actually, the third component is the state of A which is not necessarily equal to g, but determines it uniquely.

Consider the string $(a^n\$)^{f(n)}$ and two derivations $q'_0(t_1) \overset{*}{\Rightarrow} t_2$ and $q_0^N(t_2) \overset{*}{\Rightarrow} t_3$ in M' and N respectively, such that $\mathrm{yield}(t_3) = (a^n\$)^{f(n)}$. We now adopt the same terminology as in the proof of Theorem 3.9 (see also Figure 1), except that we have $\langle q, z, g\rangle$ instead of $\langle q, z\rangle$ and $z = \{p_1, \ldots, p_m\}$ instead of $z = \langle p_1, \ldots, p_m\rangle$. Note that there may now be an unbounded number of occurrences of $t(s_i, p_h)$ in t_3. We now show that the first derivation can be made uniform at the root of subtree s without changing the yield of t_3. Repetition of this process as in the proof of Theorem 3.6 leads to the desired result. We distinguish two cases.

(1) There exists h such that $yt(s_1, p_h)$ contains at least two occurrences of \$. Then we change s_2 into s_1. The yield of t_3 does not change because a substring $\$a^n\$$ remains unaltered and $f(n)$ is a function. Similarly with s_1 and s_2 interchanged.

(2) For all $i \in \{1,2\}$ and h, $yt(s_i, p_h)$ contains at most one occurrence of \$. Since we are considering state $\langle q, z, g\rangle$ of M', for fixed h both $yt(s_1, p_h)$ and $yt(s_2, p_h)$ contain the same number of occurrences of \$, viz. $g(p_h)$ which is 0 or 1. Change, say, s_2 into s_1. Then all occurrences of $yt(s_2, p_h)$ are changed into $yt(s_1, p_h)$ and consequently the number of occurrences of \$ in the yield of t_3 does not change. Hence, because f is injective, $\mathrm{yield}(t_3)$ does not change.

This concludes the proof of $(*)$. We show that $L_0 \notin yDT(\mathrm{REC})$ by the intercalation theorem of Perrault [33] as stated in Theorem 3.2.4 of [19]. Since $yDT(\mathrm{REC})$ is closed under deterministic gsm mappings [17], it may be assumed that each odd \$ in a string of L_0 is changed into ‡, i.e. it suffices to show that the language $L_0' = \{(a^n \sharp a^n \$)^{1/2 f(n)} \mid n \geq 1, f(n)\text{even}\} \cup \{(a^n \sharp a^n \$)^m a^n \sharp \mid n \geq 1, m = 1/2(f(n)-1), f(n) \text{ odd}\}$ is not in yDT(REC). According to the intercalation theorem for yDT(REC), each sufficiently long string $z \in L_0'$ has short substrings $z_1, \ldots, z_k$ which can be replaced by (longer) strings $v_1, \ldots, v_k$, such that for each i $(1 \leq i \leq k)$ the set of symbols occurring in v_i equals the set of symbols occurring in z_i, the result z' of replacing each z_i by v_i in z is in L_0', and $z \neq z'$. If we take $z = (a^n x)^{f(n)}$ for some large n (where the x is alternatingly $\sharp$ and \$), then the short substrings z_i are in $a^*, a^* \sharp a^*$ or $a^* \$ a^*$. Due to the condition on the set of symbols and the alternation of $\sharp$ and \$, v_i must be in $a^*, a^* \sharp a^*$ or $a^* \$ a^*$ respectively. But then z' too must contain $f(n)$ occurrences of $\sharp$ and \$. Since f is injective, $z = z'$, which is a contradiction. □

Theorem 3.16 implies that languages such as $\{(a^n b)^{2^n} \mid n \geq 1\}$, $\{(a^n b)^n \mid n \geq 1\}$ and $\{(a^{2^n} b)^n \mid n \geq 0\}$ are not in yT^∞(REC). We finally note that the languages of Theorem 3.16 are not in INDEXED, the class of indexed languages [1, 22]. In fact, it suffices to show that $L_0 = \{(a^n \$)^{f(n)-1} a^n \mid n \geq 1, f(n) \text{ defined}\}$ is not indexed. Suppose that it is. Since it clearly has Property (P1) of [21], it follows that $L_0 \in$ EDTOL ([22], see also Theorem 1 of [21]). But since EDTOL $\subseteq$ yDT(REC), Theorem 3.16 implies that $L_0 \notin$ EDTOL. From this contradiction it follows that $L_0 \notin$ INDEXED (cf. also [28]).

4. String Transducer Hierarchies

The bridge theorems from which the tree transducer hierarchy was obtained can also be used to establish proper hierarchies for two (closely related) string transducers. The first class of string transducers to be considered is yT restricted to monadic input trees. Such a string transducer consists of a top-down tree transducer which receives monadic input trees (i.e., vertical strings) as input and produces a tree as output of which the yield is taken.

This type of string transducer is of interest because, for monadic K, $yT(K)$ is equal to ETOL(K): the class of languages generated by K-controlled ETOL systems [36, 6, 19]. The hierarchy $\{\text{ETOL}^n(K)\}$, constructed by iterating the mechanism of control on ETOL systems, is proper under similar assumptions as for tree transducers. In particular $\{\text{ETOL}^n(\text{REG})\}$, the hierarchy considered in [8, 14], is proper (the proof of properness in [8] is, however, incomplete). Since each ETOL$^n(K)$ is a full hyper-AFL (i.e., a full AFL closed under iterated substitution), see [6], the above result can be used to obtain proper hierarchies of full hyper-AFLs.

The second class of string transducers to be considered is the well-known class of (nondeterministic) 2-way generalized sequential machines (2gsm, and 2dgsm for its deterministic version), see [3, 13, 34]. A relationship between finite-copying top-down tree transducers and a tree-walking tree-to-string transducer was established in [4] and generalized to the arbitrary case in [19]. By restricting this tree-walking transducer to monadic input trees one obtains the 2-way gsm (and the cs-pd of [42] in the general case). The resulting relationship between 2-way gsm's and ETOL systems was investigated in [34, 42]. For a survey of these connections see [19]. Under suitable initial conditions the 2-way gsm hierarchy $\{2\mathrm{GSM}^n(K)\}$ is proper (this was shown independently in [27]). Since $2\mathrm{GSM}(K)$ is also the class of languages accepted by K-controlled checking stack automata [26] (which start with an arbitrary string from a given language in K on their checking stack), $\{2\mathrm{GSM}^n(K)\}$ is also the hierarchy obtained by iterating this control mechanism for checking stack automata.

We show that properness of the 2GSM hierarchy implies properness of the ETOL hierarchy (of full hyper-AFLs) but not vice versa. Finally, the hierarchy $\{2\mathrm{GSM}^n(\mathrm{REG})\}$ is a subhierarchy of both $\{\mathrm{ETOL}^n(\mathrm{REG})\}$ and $\{yT^n(\mathrm{REC})\}$, and examples proving the properness of these hierarchies can be found in $\{2\mathrm{GSM}^n(\mathrm{REG})\}$.

In order to apply the results of the previous section to the two string transducers mentioned above we need a few relationships between 2gsm's and top-down tree transducers expressed in the following lemma. Let 2GSM and 2DGSM denote the classes of 2-way gsm's and 2-way deterministic gsm's respectively (see [3]; they are gsm's which move both forwards and backwards on their input string; the input string is surrounded by endmarkers; the output string is produced in the ordinary one-way fashion). We note that, since we will be interested mainly in classes of output languages, the precise details of the model of a 2gsm are not important.

Lemma 4.1 [19].

(1) *Let K be a class of languages (i.e., monadic tree languages) which is a trio. Then* $2\mathrm{DGSM}(K) = yDT_{fc}(K)$ *and* $2\mathrm{GSM}(K) \subseteq yT(K)$.

(2) *Let K be a tree trio. Then* $2\mathrm{DGSM}(yK) \subseteq yDT_{fc}(K)$ and $2\mathrm{GSM}(yK) \subseteq yT(K)$.

Proof. For a proof of (1) see Corollaries 4.6 and 4.10 of [19] and for a proof of (2) see Theorem 5.5 of [19]. Statement (1) expresses the relationship mentioned above between 2gsm's and top-down tree transducers [4, 34, 42]. In fact, for an arbitrary tree trio K, $yDT_{fc}(K) = DTWT(K)$ where $DTWT$ denotes the (deterministic) tree-walking transducer of [4], and $TWT(K) \subseteq yT(K)$ where TWT is its nondeterministic variant (denoted DCT and CT in [19] respectively). The proof of statement (2) in [19] is by showing that a 2dgsm walking on the yield of a tree can be simulated by a tree-walking transducer walking on the tree itself, and so $2\mathrm{DGSM}(yK) \subseteq DTWT(K) = yDT_{fc}(K)$. Similarly for $2\mathrm{GSM}(yK)$. □

We first consider the yT string transducer. For a monadic class K, we denote $yT(K)$, $yDT(K)$ and $yDT_{fc}(K)$ also by $\mathrm{ETOL}(K)$, $\mathrm{EDTOL}(K)$ and $\mathrm{EDTOL}_{fc}(K)$ respectively. In [19], $\mathrm{EDTOL}_{fc}(K)$ is denoted by $\mathrm{EDTOL}_{\mathrm{FIN}}(K)$; $\mathrm{EDTOL}_{\mathrm{FIN}}(\mathrm{REG})$ is the class of ETOL languages of finite index (see also [31]). For a trio K, the class $\mathrm{ETOL}(K)$ is equal to CS-PD(K), where CS-PD is a class of restricted 2-way pushdown transducers discussed in [42, 20, 19]: the transducer pushes a symbol on the pushdown only when moving its input head one square to the right, and pops a symbol only when moving one square to the left.

The composition of these yT transducers is quite strange from the tree transducer point of view: the yield of the output tree of the first tree transducer is turned vertical and then used as input to the second tree transducer! Thus, e.g., ETOL(ETOL(K))=$yT(yT(K))$ which class is quite different from $yT^2(K)=yT(T(K))$. From the point of view of control on ETOL systems the composition is however quite natural.

The hierarchy result for these controlled ETOL systems is stated in the next theorem. Properness of this hierarchy is caused, as for the tree transducer hierarchy, by the alternation of copying and nondeterminism present in ETOL systems.

Theorem 4.2. *Let K be a trio. If* $\mathrm{EDTOL}_{fc}(K) \subsetneq \mathrm{ETOL}(K)$, *then for all* $n \geq 1$ *the following inclusions are proper, where K_n denotes* $\mathrm{ETOL}^n(K)$: $K_n \subsetneq \mathrm{EDTOL}_{fc}(K_n) \subsetneq \mathrm{EDTOL}(K_n) \subsetneq \mathrm{ETOL}(K_n) = K_{n+1}$.

Proof. The proof is completely analogous to that of Theorem 3.12. It is easy to show that if K is a trio then ETOL(K) is a trio (by Lemma 2.5, ETOL(K) is closed under regular substitution; by Lemma 4.3 of [6], ETOL(K) is closed under intersection with regular languages). Hence K_n is a trio for all n. Now Theorems 3.1 and 3.6 are applicable without change (take the monadic case). The key remark is that instead of Theorem 3.9 we can use the same result with $yDT_{fc}(T(K))$ replaced by $yDT_{fc}(yT(K))$; this result holds because $yDT_{fc}(yT(K)) = \mathrm{2DGSM}(yT(K)) \subseteq yDT_{fc}(T(K))$ by Lemma 4.1 (1) and (2), and hence Theorem 3.9 can be used! To obtain the analogues of Corollaries 3.3, 3.8 and 3.11 we use the monadic cases of Lemmas 2.5 and 2.6.

Thus, similar to the tree transducer hierarchy, if L_n is a language in $K_n - \mathrm{EDTOL}(K_{n-1})$, then

$c_2(L_n) \in \mathrm{EDTOL}_{fc}(K_n) - K_n$,
$c_*(L_n) \in \mathrm{EDTOL}(K_n) - \mathrm{EDTOL}_{fc}(K_n)$, and
$\mathrm{rub}(c_*(L_n)) \in \mathrm{ETOL}(K_n) - \mathrm{EDTOL}(K_n)$. □

The analogue of Corollary 3.13 also holds for ETOL.

Note that if K is a trio and $\mathrm{ETOL}_{fc}(K) \subsetneq \mathrm{ETOL}(K)$, then both $\{\mathrm{ETOL}^n(K)\}$ and $\{yT^n(K)\}$ are proper hierarchies.

We now turn to the ETOL hierarchy which starts with the regular languages [8, 14]. Note that ETOL(REG) is equal to the class of usual (uncontrolled) ETOL languages [36, 6].

Theorem 4.3. *The* ETOL *hierarchy* $\{\mathrm{ETOL}^n(\mathrm{REG})\}$ *is proper. More precisely, for* $K_n = \mathrm{ETOL}^n(\mathrm{REG})$ *and* $n \geq 1$ *the inclusions in Theorem* 4.2 *are proper.*

Proof. Since $\mathrm{EDTOL}_{fc}(\mathrm{REG}) \subseteq yDT_{fc}(\mathrm{REC})$, all languages in $\mathrm{EDTOL}_{fc}(\mathrm{REG})$ have semi-linear Parikh-sets [19] and hence languages such as $\{a^{2^n} \mid n \geq 0\}$ and $\{a^n \mid n \text{ is not prime}\}$ are in EDTOL(REG) but not in $\mathrm{EDTOL}_{fc}(\mathrm{REG})$. Now

Theorem 4.2 is applicable. Thus the same sequence of languages proves the properness of both hierarchies ETOL^n(REG) and yT^n(REC). Compare this proof with the proof of Theorem 3.14. □

It is open whether EDTOL^n(REG) is a proper hierarchy.

The ETOL hierarchy contains a language not in the tree transducer hierarchy. In fact the language $\{(a^nb)^n | n \geq 1\}$ is an example. It is not in yT^n(REC) by Theorem 3.16, but it is in EDTOL(ETOL(REG)): in fact it is $yM(L)$ where L is the monadic language $\{a^nbc^n | n \geq 1\}$ and M copies bc^n n times, using the a's, and translates each bc^n into a tree with yield a^nb; clearly $\{a^nbc^n | n \geq 1\} \in$ ETOL(REG).

It is open whether there exists a language in yT^∞(REC) not in ETOL$^\infty$(REG), but we conjecture that there is already one in yDT_{fc}(REC).

It was shown in [6] that if K is, say, a full AFL, then ETOL(K) is a full hyper-AFL (i.e., a full AFL closed under iterated substitution, see [37, 6]), and similarly EDTOL(K) is a "full dhyper-QAFL" [7]. Thus Theorem 4.2 can be used to obtain proper hierarchies of full hyper-AFLs, such as $\{$ETOL^n(REG)$\}$. Moreover, if K is included in the class CS of context-sensitive languages, then the hierarchy $\{$ETOL^n(K)$\}$ is also inside CS. This follows from the next lemma which was shown in [8].

Lemma 4.4. [8]. *If K is a full* AFL *included in CS, then so is* ETOL(K).

Proof. The following proof is based on similar results in [45] and [26]. Let K be a full AFL and $K \subseteq CS$. By a result in [26, 30] this implies that 2DGSM(K) $\subseteq CS$, i.e., $yDT_{fc}(K) \subseteq CS$ by Lemma 4.1 (1). Now note that $DT_{fc}(K) \subseteq yDT_{fc}(K)$ where it should be recalled that each tree is a string (in fact, it is easy to see that there is an $M \in DT_{fc(1)}$ such that, for every input tree t, yield($M(t)$)$=t$. Hence $DT_{fc}(K) \subseteq yM(DT_{fc}(K)) = yDT_{fc}(K)$ because $DT_{fc}(K)$ is closed under DT_{fc} [19]). Hence $DT_{fc}(K) \subseteq CS$. It was proved in [45] that if a class of tree languages K_1 is closed under linear top-down tree transducers and $K_1 \subseteq CS$, then $yT(K_1) \subseteq CS$. Since it is easy to show that $T_{fc}(K)$ is closed under linear top-down tree transducers, we get that $yT(DT_{fc}(K)) \subseteq CS$ and hence certainly $yT(K) =$ ETOL(K) is included in CS. □

In the next theorem we show the existence of a proper hierarchy of full hyper-AFLs between the tree transducer hierarchy yT^∞(REC) and CS. It was shown in [45, 46] that yT^∞(REC) is a full AFL included in CS.

Theorem 4.5. *There is a proper hierarchy of full hyper-*AFL*s between yT^∞*(REC) *and CS.*

Proof. We show that yT^∞(REC) satisfies the conditions of Theorem 4.2. First, EDTOL$_{fc}$(yT^∞(REC)) $= yDT_{fc}(yT^\infty$(REC)) and, by Lemma 4.1 (1) and (2), this class is included in $yDT_{fc}(T^\infty$(REC)) $= yT^\infty$(REC). Hence yT^∞(REC) is closed under EDTOL$_{fc}$. But it is not closed under ETOL because the language $\{(a^nb)^n | n \geq 1\}$ is in ETOL(ETOL(REG)) $\subseteq$ ETOL(yT^∞(REC)) as shown above but not in yT^∞(REC) by Theorem 3.16. Consequently ETOL^n(yT^∞(REC)) is a proper hierarchy of full hyper-AFLs containing yT^∞(REC). Since yT^∞(REC) is a full AFL included in CS [9], Lemma 4.4 implies that ETOL^n(yT^∞(REC)) $\subseteq CS$. □

The second string transducer hierarchy to be considered is that of the 2-way gsm. The class 2DGSM is closed under composition [10], but 2GSM is not [30]. The fact that a hierarchy theorem will hold for 2GSM could be guessed from Lemma 4.1 and the fact that the examples constructed for the tree transducer and ETOL hierarchies by the rub and c_* operations can be realized by (compositions of) 2-way gsm's. Thus properness of the 2GSM hierarchy is again caused by the combined presence of copying and nondeterminism in a 2-way gsm.

The next lemma shows that the 2GSM hierarchy is a subhierarchy of both the ETOL hierarchy and (if possible) of the tree transducer hierarchy.

Lemma 4.6.

(1) *Let K be a trio. Then for all $n \geq 0$*

$$\mathrm{2GSM}^n(K) \subseteq \mathrm{ETOL}^n(K) \text{ and } \mathrm{2DGSM}(\mathrm{2GSM}^n(K)) \subseteq \mathrm{EDTOL}_{fc}(\mathrm{ETOL}^n(K)).$$

(2) *Let K be a tree trio. Then for all $n \geq 0$*

$$\mathrm{2GSM}^n(yK) \subseteq yT^n(K) \text{ and } \mathrm{2DGSM}(\mathrm{2GSM}^n(yK)) \subseteq yDT_{fc}(T^n(K)).$$

Proof. If K is a trio then so is ETOL(K), and if K is a tree trio then so is $T(K)$. Statements (1) and (2) now follow directly by induction from (1) and (2) of Lemma 4.1, respectively. □

The hierarchy result for 2GSM is stated in the next theorem. Recall the notion of a correct diagram from Section 2.

Theorem 4.7. *Let K be a trio such that $\mathrm{2DGSM}(K) \subsetneq \mathrm{2GSM}(K)$. Then the following holds.*

(1) *For all $n \geq 1$, $\mathrm{2GSM}^n(K) \subsetneq \mathrm{2DGSM}(\mathrm{2GSM}^n(K)) \subsetneq \mathrm{2GSM}^{n+1}(K)$.*

There even exist (for all $n \geq 1$) languages $L_n \in \mathrm{2GSM}^n(K)$ and $L'_n \in \mathrm{2DGSM}(\mathrm{2GSM}^n(K))$ such that $L_n \notin \mathrm{EDTOL}(\mathrm{ETOL}^{n-1}(K))$ and $L'_n \notin \mathrm{ETOL}^n(K)$. If, moreover, $\mathrm{ETOL}(K) - \mathrm{2GSM}^\infty(K) \neq \varnothing$, then the diagram of Figure 3A is correct.

(2) *Let K_1 be a tree trio such that $K \subseteq yK_1$ and $\mathrm{2GSM}(K) - yDT_{fc}(K_1) \neq \varnothing$. Then there exist languages $L_n \in \mathrm{2GSM}^n(K)$ and $L'_n \in \mathrm{2DGSM}(\mathrm{2GSM}^n(K))$ such that $L_n \notin \mathrm{EDTOL}(\mathrm{ETOL}^{n-1}(K))$ and $L'_n \notin \mathrm{ETOL}^n(K)$, and moreover $L_n \notin yDT(T^{n-1}(K_1))$ and $L'_n \notin yT^n(K_1)$. If $yK_1 - \mathrm{2GSM}^\infty(K) \neq \varnothing$, then the diagram of Figure 3B is correct.*

Proof. (1) By Lemma 4.6 (1) the 2GSM hierarchy is included in the ETOL hierarchy as indicated in Figure 3A. Also, by Lemma 4.1 (1), $\mathrm{2DGSM}(K) = \mathrm{EDTOL}_{fc}(K)$. Hence, to prove (1), it suffices to show the existence of the languages L_n and L'_n (all other proper inclusions and incomparabilities follow from the existence of these languages and the assumption that $\mathrm{ETOL}(K) - \mathrm{2GSM}^\infty(K) \neq \varnothing$).

Let $L_0 \in \mathrm{2GSM}(K) - \mathrm{2DGSM}(K)$ and let $L_1 = \mathrm{rub}(L_0)$. Since $\mathrm{2GSM}(K)$ is clearly closed under regular substitution, $L_1 \in \mathrm{2GSM}(K)$. Moreover, by Theorem

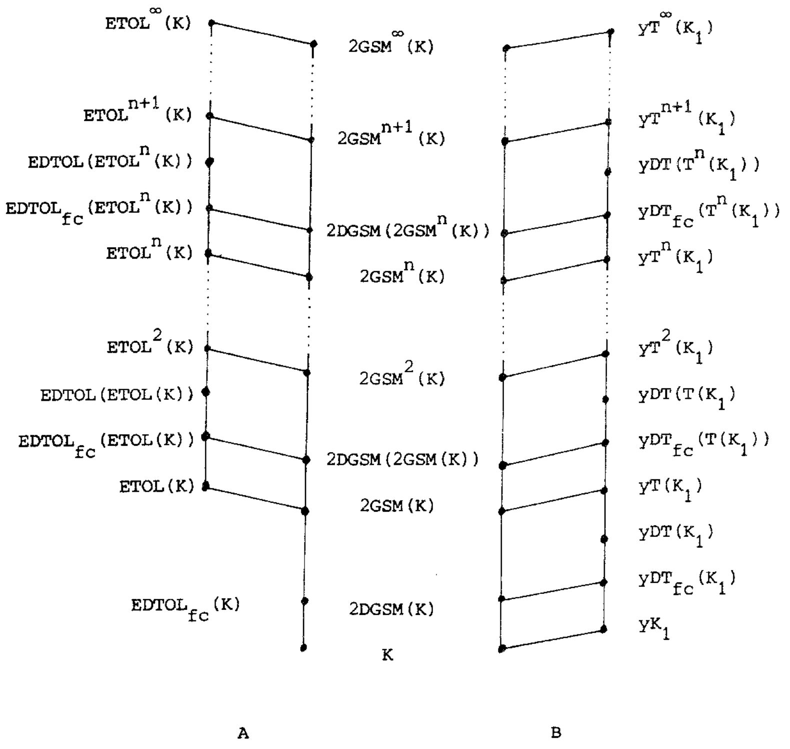

Fig. 3. Three hierarchies of transducers.

3.1, $L_1 \in \mathrm{EDTOL}(K)$ would imply that $L_1 \in \mathrm{EDTOL}_{fc}(K) = 2\mathrm{DGSM}(K)$. Hence L_1 satisfies the requirements. We now define by induction $L'_n = c_2(L_n)$ and $L_{n+1} = \mathrm{rub}(c_*(L_n))$. Obviously, if $L_n \in K'$ then $c_2(L_n) \in 2\mathrm{DGSM}(K')$, $c_*(L_n) \in 2\mathrm{GSM}(K')$, and $\mathrm{rub}(c_*(L_n)) \in 2\mathrm{GSM}(K')$. It follows from the proof of Theorem 4.2 that $c_2(L_n) \notin \mathrm{ETOL}^n(K)$ and $\mathrm{rub}(c_*(L_n)) \notin \mathrm{EDTOL}(\mathrm{ETOL}^n(K))$.

(2) By Lemma 4.6 (2) the 2GSM hierarchy is included in the tree transducer hierarchy as indicated in Figure 3B. Let $L_0 \in 2\mathrm{GSM}(K) - yDT_{fc}(K_1)$. Then, by Theorem 3.1, $L_1 = \mathrm{rub}(L_0)$ is not in $yDT(K_1)$. Since $2\mathrm{DGSM}(K) \subseteq yDT_{fc}(K_1)$, $L_0 \notin 2\mathrm{DGSM}(K)$ and $L_1 \notin \mathrm{EDTOL}(K)$ as above. Using the same argument as in (1) together with the proof of Theorem 3.12, it follows that L_n and L'_n, defined as in (1), satisfy all requirements. □

Note that, in Fig. 3A, EDTOL(K) could be added between $\mathrm{EDTOL}_{fc}(K)$ and ETOL(K). The only reason we did not do so is that it is open whether $\mathrm{EDTOL(REG)} - 2\mathrm{GSM}^{\infty}(\mathrm{REG}) \neq \varnothing$ and hence correctness of the diagram for $K = \mathrm{REG}$ is unknown.

Hierarchy theorems for the 2-way gsm were obtained independently by Greibach. In fact, in [27] it is shown, as in Theorem 4.7 (1), that if K is a trio such that $2\mathrm{DGSM}(K) \subsetneq 2\mathrm{GSM}(K)$ then for all $n \geq 1$ $2\mathrm{GSM}^n(K) \subsetneq 2\mathrm{DGSM}(2\mathrm{GSM}^n(K)) \subsetneq 2\mathrm{GSM}^{n+1}(K)$. But in the proof in [27] the appropriate languages L_n and L'_n are defined by $L'_n = c_2(L_n)$ and $L_{n+1} = c_*(L_n)$. Hence for the hierarchy $\{2\text{GSM}^n(K)\}$ the combined copying and nondeterminism present in the operation c_* suffices to prove properness, and the rub-operation is needed only to find languages in the 2GSM hierarchy which prove properness of the ETOL hierarchy (and the tree transducer hierarchy).

We now turn to the case $K = \text{REG}$.

Theorem 4.8. *For* $K = \text{REG}$ *and* $K_1 = \text{REC}$ *both diagrams of Figure* 3 *are correct.*

Proof. Note that $\text{REG} \subseteq CF = y\text{REC}$ [39]. The language $L_0 = \{a^n \mid n \text{ is not prime}\}$ is in 2GSM(REG), but not in $yDT_{fc}(\text{REC})$, because its Parikh-set is not semi-linear. Hence Theorem 4.7 is applicable.

It is shown in [27] that $CF - 2\text{GSM}^\infty(\text{REG}) \neq \varnothing$. Note that $CF \subseteq \text{ETOL(REG)}$ [36] and $CF = y\text{REC}$. □

Thus (for the regular case) the 2GSM hierarchy is a "small" hierarchy inside both the ETOL hierarchy and the tree transducer hierarchy.

When gluing A and B of Fig. 3 together, note that $\text{ETOL(REG)} \subseteq yT(\text{REC})$ and hence, by Lemma 4.1, $\text{EDTOL}_{fc}(\text{ETOL(REG)}) \subseteq yDT_{fc}(T(\text{REC}))$. But, as noted before, EDTOL(ETOL(REG)) contains a language not in $yT^\infty(\text{REC})$, whereas it is open whether there is a tree transformation language not in the ETOL hierarchy.

It follows from Theorem 4.8 that $\{2\text{GSM}^n\}$ is a proper hierarchy of string transductions. However, as in the case of tree transducers [18], this hierarchy does not contain new partial functions, as stated in the next theorem.

Theorem 4.9. $2\text{GSM}^\infty \cap \{f \mid f \textit{ is partial function}\} = 2D\text{GSM}$.

Proof. (sketch). We first observe that if M is an arbitrary 2gsm then there exists a 2dgsm N with the same domain as M such that $N \subseteq M$, cf. [10]. In fact, if M has m states and M produces output y on input w, then there is a computation producing output y' which never visits a square of w more than m times. If M computes a partial function, then $y' = y$. It is possible to construct a 2dgsm N which keeps track of the "visiting sequences" of the first (in some order) m-visit computation of M on w and thus simulates the behaviour of M according to this computation.

Let f be a partial function realized by the composition of the 2gsm's $M_1, M_2, \ldots, M_n$. Since the domain of a 2gsm is regular, we may assume that M_i only produces output which is in the domain of M_{i+1}. Hence we may replace each M_i by the 2dgsm N_i described above. In fact, since f is a partial function, the "lost" computations cannot affect the final output of M_n. Since 2dgsm's are closed under composition [10], the result follows. □

Let K be a trio such that $\text{REG} \subseteq K \subseteq CF$. Then the $2\text{GSM}^n(K)$ hierarchy is proper, because $\{a^n \mid n \text{ is not prime}\}$ is in 2GSM(REG) but not in $2\text{DGSM}(CF)$: note that $2\text{DGSM}(CF) = 2\text{DGSM}(y\text{REC}) \subseteq yDT_{fc}(\text{REC})$ by Lemma 4.1, and see the proof of Theorem 4.8. The same result is proved in [27] by other means. In fact, the $2\text{GSM}^n(CF)$ hierarchy lies properly between the $2\text{GSM}^n(\text{REG})$ hierarchy and the tree transducer hierarchy $\{yT^n(\text{REC})\}$ (in the sense that if in the diagram of Fig. 3B for $K = \text{REG}$ and $K_1 = \text{REC}$ the classes $2\text{GSM}^n(CF)$ and

2DGSM(2GSM^n(CF)) are added properly between the two hierarchies at their appropriate places, then the resulting diagram is correct for $n \geq 1$; in particular, 2GSM^n(REG) $\subsetneq$ 2GSM^n(CF) $\subsetneq$ yT^n(REC) for all $n \geq 1$, and 2GSM$^\infty$(REG) $\subsetneq$ 2GSM$^\infty$(CF) $\subsetneq$ yT^∞(REC)). To show this we need the following result of [27]: let K_1 be a full principal substitution closed AFL with a generator which contains no infinite regular language; if $K_1 \subseteq$ 2GSM$^\infty$(K) and K is a full semi-AFL, then $K_1 \subseteq K$. Since CF satisfies the conditions for K_1, this implies that $CF-$ 2GSM$^\infty$(REG) $\neq \varnothing$ (which we already used in the proof of Theorem 4.8). Moreover yT(REC) also satisfies the conditions for K_1 (use a generator obtained from the CT-PD machine model of yT(REC) in [19] and insert brackets in the obvious way) and so yT(REC) $-$ 2GSM$^\infty$(CF) $\neq \varnothing$. Note that 2GSM$^\infty$(CF) is also (properly) included in ETOL$^\infty$(REG), because $CF \subseteq$ ETOL(REG).

The criterion 2DGSM(K) $\subsetneq$ 2GSM(K) is necessary and sufficient for the 2GSM^n(K) hierarchy to be proper. Indeed, if 2DGSM(K) = 2GSM(K), then 2GSM^2(K) = 2GSM(2DGSM(K)) $\subseteq$ 2GSM(K) [30] and the hierarchy collapses to 2DGSM(K). Theorem 4.7 shows that properness of the 2GSM^n(K) hierarchy always implies properness of the ETOL^n(K) hierarchy. (This does not hold vice versa: we have shown that for yT^∞(REC) the ETOL hierarchy is proper, cf. the proof of Theorem 4.5, but by Lemma 4.1 (2) yT^∞(REC) is closed under 2GSM). Thus a proper 2GSM^n(K) hierarchy implies the existence of a proper hierarchy of full hyper-AFLs containing K. In particular, using another result of [27] we obtain the following result concerning the indexed languages [1].

Theorem 4.10. *There is a proper hierarchy of full hyper-*AFL*s between* INDEXED *and CS.*

Proof. By [27], the hierarchy {2GSM^n(INDEXED)} is proper. Since INDEXED is a full AFL contained in CS [1], Lemma 4.4 implies that the hierarchy {ETOL^n(INDEXED)} is inside CS. This hierarchy is proper by the above result and Theorem 4.7., and it consists of full hyper-AFLs [6]. □

Conclusion

Proper hierarchies of output languages of top-down tree transducers, ETOL systems and 2-way gsm's exist whenever two of these transducers can do more than one (for a given class of input languages). These proper hierarchies arise due to the alternation of copying and nondeterminism which can be realized by such transducers. We have shown that bounded copying tree transducers cannot do unbounded copying of a nondeterministic language, and that unbounded copying deterministic tree transducers cannot handle regular substitution. Knowledge of positive and negative closure properties of several classes of tree transducer languages was essential in proving these facts.

It is of interest to establish still finer subhierarchies of the tree transducer hierarchy than the one given in Figure 2 (Theorem 3.12). Closure properties of the other restricted classes of tree transducer languages may be investigated with respect to other operations. As an example we consider metalinear tree transducers (abbreviated by T_{ml}): these are finite copying tree transducers which copy subtrees only at the first step of their derivation and are linear in the rest of the derivation [19]. As in the finite copying case, determinism does not restrict the class of output languages. It was shown in Theorem 3.2.10 of [19] that, under certain conditions on K, the class $yDT_{ml}(K)$ is not closed under Kleene star. It is easy to show that if K is closed under the operation $\text{star}(L) = \{\$(t_1\$(t_2\ldots\$(t_{n-2}\$(t_{n-1}t_n))\ldots))\,|\,n \geq 1, t_i \in L\}$, where \$ is of rank 2, then $yDT_{fc}(K)$ is closed under Kleene star. Assuming that the class K from which we start is closed under union and star, $yDT_{ml}(K_n)$ can be added between yK_n and $yDT_{fc}(K_n)$ in the diagram of Figure 2 (Theorem 3.12), and similarly in the monadic case $K_n \subsetneq \text{EDTOL}_{ml}(K_n) \subsetneq \text{EDTOL}_{fc}(K_n)$ in Theorem 4.2. The corresponding class of 2-way gsm's is the class 2DGSM_{fp} of 2dgsm's which are "finite pass", i.e., which make a finite number of passes (from left to right or vice versa) over the input tape (equivalently, they are "finite turn" or "finite reversal", see [26] and [19]). Thus to Theorem 4.7 the statement $\text{2GSM}^n(K) \subsetneq \text{2DGSM}_{fp}(\text{2GSM}^n(K)) \subsetneq \text{2DGSM}(\text{2GSM}^n(K))$ can be added, together with the appropriate statement concerning the existence of languages which prove the properness of the inclusions in all three hierarchies.

In [47] a refinement of the tree transducer hierarchy is considered by adding $y\text{NHOM}(K_n)$ to Fig. 2, where NHOM denotes the class of one-state nondeterministic top-down tree transducers.

It is suggested in [43] that properness of the 2GSM hierarchy can also be shown by reducing it to the (known) properness of a hierarchy of space-complexity classes. It would be interesting to know whether a similar proof can be given for the tree transducer hierarchy.

Acknowledgments

I wish to thank Giora Slutzki for many days of stimulating discussions. I thank the referee for careful reading of the paper and for helpful suggestions.

References

1. A. V. Aho, Indexed grammars—an extension of context-free grammars, *JACM* 15, 647–671 (1968).
2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, A general theory of translation, *Math. Syst. Th.* 3, 193–221 (1969).
3. A. V. Aho and J. D. Ullman, A characterization of two-way deterministic classes of languages, *JCSS* 4, 523–538 (1970).
4. A. V. Aho and J. D. Ullman, Translations on a context-free grammar, *Inf. and Control* 19, 439–475 (1971).
5. A. Arnold and M. Dauchet, Translations de forets reconnaissables monadiques; forets coregulieres; RAIRO Informatique Theoretique 10, 5–28 (1976).
6. P. R. J. Asveld, Controlled iteration grammars and full hyper-AFLs, *Inf. and Control* 34, 248–269 (1977).
7. P. R. J. Asveld and J. Engelfriet, Iterated deterministic substitution, *Acta Informatica* 8, 285–302 (1977).
8. P. R. J. Asveld and J. van Leeuwen, Infinite chains of hyper-AFLs, Memorandum 99, Twente University of Technology, 1975.

9. B. S. Baker, Tree transductions and families of tree languages; Ph.D. Thesis; Harvard University, Report TR-9-73, 1973; see also references [44, 45, 46].
10. M. P. Chytil and V. Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings; in *Fourth Colloquium on Automata, Languages and Programming*, Turku, A. Salomaa and M. Steinby eds., *Lecture Notes in Computer Science* 52, Springer-Verlag, Berlin, 1977.
11. A. Ehrenfeucht and G. Rozenberg, Three useful results concerning L languages without interactions; in *L systems*, G. Rozenberg and A. Salomaa eds., pp. 72–77, *Lecture Notes in Computer Science* 15, Springer-Verlag, Berlin, 1974.
12. A. Ehrenfeucht and G. Rozenberg, On inverse homomorphic images of deterministic ETOL languages, in *Automata, Languages, Development*, A. Lindenmayer and G. Rozenberg eds., pp. 179–189, North-Holland, Amsterdam, 1976.
13. R. W. Ehrich and S. S. Yau, Two-way sequential transductions and stack automata, *Inf. and Control* 18, 404–446 (1971).
14. G. S. Eisman, Derivation controlled Lindenmayer systems; excerpt from Ph.D. Thesis at University of California, Berkeley, 1977.
15. J. Engelfriet, Surface tree languages and parallel derivation trees, *TCS* 2, 9–27 (1976).
16. J. Engelfriet, Bottom-up and top-down tree transformations—a comparison, *Math. Syst. Th.* 9, 198–231 (1975).
17. J. Engelfriet, Top-down tree transducers with regular look-ahead, *Math. Syst. Th.* 10, 289–303 (1977).
18. J. Engelfriet, On tree transducers for partial functions; *Inf. Proc. Letters* 7, 170–172 (1978).
19. J. Engelfriet, G. Rozenberg and G. Slutzki: Tree transducers, L systems and two-way machines, *JCSS* 20, 150–202 (1980).
20. J. Engelfriet, E. Meineche Schmidt and J. van Leeuwen, Stack machines and classes of nonnested macro languages, *JACM* 27, 96–117 (1980).
21. J. Engelfriet and S. Skyum, Copying theorems, *Inf. Proc. Letters* 4, 157–161 (1976).
22. M. J. Fischer, Grammars with macro-like productions, Ph.D. Thesis, Harvard University, 1968.
23. S. Ginsburg, *Algebraic and automata-theoretic properties of formal languages*, North-Holland/American Elsevier, Amsterdam/New York, 1975.
24. S. A. Greibach, Chains of full AFLs, *Math. Syst. Th.* 4, 231–242 (1970).
25. S. A. Greibach, Syntactic operators on full semi-AFLs, *JCSS* 6, 30–76 (1972).
26. S. A. Greibach, One-way finite visit automata, *TCS* 6, 175–221 (1978).
27. S. A. Greibach, Hierarchy theorems for two-way finite state transducers, *Acta Informatica* 11, 89–101 (1978).
28. T. Hayashi, On derivation trees of indexed grammars—an extension of the *uvwxy*-Theorem, Publ. of the Research Inst. for Math. Sciences (Kyoto University) 9 (1973) 61–92.
29. J. E. Hopcroft and J. D. Ullman, *Formal languages and their relation to automata*, Addison-Wesley, Reading, Mass., 1969.
30. D. Kiel, Two-way *a*-transducers and AFL, *JCSS* 10, 88–109 (1975).
31. M. Latteux, EDTOL-systemes ultralineaires et operateurs associes, Publication no. 100, Université de Lille, 1977.
32. W. F. Ogden and W. C. Rounds: Composition of n transducers; Fourth ACM Symposium on Theory of Computing, pp. 198–206, 1972.
33. C. R. Perrault, Intercalation lemmas for tree transducer languages, *JCSS* 13, 246–277 (1976).
34. V. Rajlich, Absolutely parallel grammars and two-way finite state transducers, *JCSS* 6, 324–342 (1972).
35. W. C. Rounds, Mappings and grammars on trees, *Math. Syst. Th.* 4, 257–287 (1970).
36. G. Rozenberg, Extensions of tabled *OL*-systems and languages, *Int. J. Comp. Inform. Sci.* 2, 311–336 (1973).
37. A. Salomaa, Macros, iterated substitution and Lindenmayer AFLs, *DAIMI* PB-18, University of Aarhus, 1973 (see also *L Systems*, G. Rozenberg and A. Salomaa eds., *Lecture Notes in Computer Science* 15, pp. 250–253, Springer-Verlag, Berlin, 1974).
38. S. Skyum, Decomposition theorems for various kinds of languages parallel in nature, *SIAM J. Comp.* 5, 284–296 (1976).

39. J. W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, *JCSS* 1, 317–322 (1967).
40. J. W. Thatcher, Generalized^2 sequential machine maps, *JCSS* 4, 339–367 (1970).
41. J. W. Thatcher, Tree automata: an informal survey; in *Currents in the Theory of Computing* (ed. A. V. Aho), Prentice-Hall, 1973.
42. J. van Leeuwen, Variations of a new machine model, 17th IEEE Symposium on Foundations of Computer Science, Houston, Texas, pp. 228–235 (1976).
43. J. Engelfriet, Two-way automata and checking automata, in *Foundations of Computer Science III*, eds. J. W. de Bakker and J. van Leeuwen, MC Tract 108, Mathematical Centre, Amsterdam, 1979.
44. B. S. Baker, Composition of top-down and bottom-up tree transductions, *Inf. and Control* 41, 186–213 (1979).
45. B. S. Baker, Generalized syntax directed translation, tree transducers, and linear space, *SIAM J. Comput.* 7, 376–391 (1978).
46. B. S. Baker, Tree transducers and tree languages, *Inf. and Control* 37, 241–266 (1978).
47. J. Engelfriet and S. Skyum, The copying power of one-state tree transducers, Report *DAIMI* PB-91, Aarhus University, 1978, to appear in *JCSS*.

Received April 1980, and in revised form April 1981.