Information and Computation **147**, 111–144 (1998)
Article No. IC982736

The Generating Power of Total Deterministic Tree Transducers

Sebastian Maneth*

Grundlagen der Programmierung, Institut für Softwaretechnik I, Fakultät Informatik, Technische Universität Dresden, 01062 Dresden, Germany
E-mail: maneth@inf.tu-dresden.de

Attributed tree transducers are abstract models used to study properties of attribute grammars. One abstraction which occurs when modeling attribute grammars by attributed tree transducers is that arbitrary trees over a ranked alphabet are taken as input, instead of derivation trees of a context-free grammar. In this paper we show that with respect to the generating power this is *not* an abstraction; i.e., we show that attributed tree transducers and attribute grammars generate the same class of term (or tree) languages. To prove this, a number of results concerning the generating power of top-down tree transducers are established, which are interesting in their own. We also show that the classes of output languages of attributed tree transducers form a hierarchy with respect to the number of attributes. The latter result is achieved by proving a hierarchy of classes of tree languages generated by context-free hypergraph grammars with respect to their rank.

1. INTRODUCTION

Attribute grammars were introduced by Knuth in [Knu68] (see also [DJL88]) to model syntax-directed semantics. An attribute grammar can be seen as a device which translates strings (viz., the strings generated by a context-free string grammar) into values. In fact, every computed value is the interpretation of an expression (or term). If we drop this interpretation, then an attribute grammar can be seen as a device which translates strings into terms (or, equivalently trees; in the following, these two words will be used as synonyms). The *term-generating power* of attribute grammars is the class of term languages which are output languages of such string-to-tree translations of attribute grammars. This class is denoted by $OUT(AG, TERMS)$.

* The work of this author has been supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant Vo 461/3-1. Present address: Department of Computer Science, Leiden University, PO Box 9512, 2300 RA Leiden, The Netherlands, E-mail: maneth@wi.leidenuniv.nl.

0890-5401/98 $25.00

Recently, an interesting result concerning the term-generating power of attribute grammars has been established; namely, in [EH92] it is shown that context-free hypergraph grammars (see, e.g., [Hab92, DKH97, Eng97]) have the same term-generating power as attribute grammars.

In this paper we will show that attributed tree transducers have the same term-generating power as attribute grammars. Let $OUT(AT)$ denote the class of term languages generated by attributed tree transducers as output languages. Thus, we prove that $OUT(AG, TERMS) = OUT(AT)$.

Attributed tree transducers were introduced by Fülöp in [Fül81] to model attribute grammars on a more abstract level. Here, we only consider total deterministic attributed tree transducers and, in particular, this means that every input tree is translated into exactly one output tree. Let us explain the steps of abstraction when modeling attribute grammars by attributed tree transducers (see also [FHVV93] for a discussion on this topic):

(i) instead of taking derivation trees of a context-free grammar as input, trees over a ranked alphabet are taken as input,

(ii) instead of taking values of some semantic domain as output, trees over some ranked alphabet are taken as output,

(iii) tree substitution is the only semantic function, and

(iv) with every input symbol *all* attributes are associated.

The consideration of the term-generating power of attribute grammars only makes sense after having applied steps (ii) and (iii). Thus, the question is how steps (i) and (iv) change the term-generating power. Clearly, step (iv) does not change the term-generating power, because we can simply associate all attributes with every nonterminal and then assign dummy values to the attributes which were originally not assigned to a particular nonterminal. This will not change the transformation, because these attributes are not used to produce the output.

Step (i) seems to cause a crucial change with respect to the term-generating power, in the sense that the generating power decreases, because the many-sortedness of the input tree no longer provides a means to control the generation of terms. For this reason it is common in the field of syntax-directed translation to take an arbitrary recognizable tree language as input instead of a set of derivation trees (cf. [Rou70, ERS80]). Thus, there are two ways of modeling the term-generating power of attribute grammars by attributed tree transducers, namely, $OUT(AT)$ and $AT(RECOG)$, where $AT(RECOG)$ is the class of output languages of attributed tree transducers taking recognizable tree languages as input. In this paper we show that both of these classes are equal to $OUT(AG, TERMS)$.

Let us discuss the equalities $OUT(AT) = OUT(AG, TERMS) = AT(RECOG)$ in more detail. The inclusions $OUT(AT) \subseteq OUT(AG, TERMS) \subseteq AT(RECOG)$ are straightforward to prove, whereas the inclusion $AT(RECOG) \subseteq OUT(AT)$ is more involved. Namely, for an attributed tree transducer A and a recognizable tree language L this inclusion means that with respect to the generating power we can get rid of the recognizable tree language L. A similar result for (partial) top-down tree transducers is obtained in [ERS80]; i.e., in Theorem 3.2.1 of [ERS80] it is

shown that $T(RECOG) \subseteq OUT(T)$, where T is the class of all (nondeterministic) top-down tree transductions. This was proved as follows. Let L be a recognizable tree language and let M be a top-down tree transducer. Then there is a top-down finite tree automaton M' which has L as domain. Now a top-down tree transducer M'' is constructed which simulates M and simultaneously checks whether the input tree is in L by encoding the states of M' into the states of M''.

In the case of attributed tree transducers the construction is more involved, because the attributed tree transducer A performs a tree walk over the input tree, i.e., over the output tree of a top-down finite tree automaton M, which has the recognizable tree language L as domain and output. Ganzinger and Giegerich [Gan83, Gie88] proposed a technique for the composition of attribute couplings which are devices similar to attributed tree transducers. It works in such a way that the right-hand sides of the rules of the first attributed tree transducer are translated by the second attributed tree transducer. However to apply this technique the first attributed tree transducer must be *syntactic single used restricted* (for short, ssur) which means that in the set of rules of an input symbol every outside attribute is used exactly once. Also, an attributed tree transducer is total and deterministic, whereas the top-down finite tree automaton M is in general *partial* and *nondeterministic*. Therefore, we must find a class of top-down tree transducers which are *total deterministic* and the class of output languages of which includes *RECOG*. It turns out that particular total deterministic, superlinear top-down tree transducers, called *semi-relabelings*, generate exactly the recognizable tree languages as output languages. Superlinear means that in all right-hand sides of rules for a given input symbol each variable occurs at most once. A top-down tree transducer can be seen as an attributed tree transducer without inherited attributes, where the states of the top-down tree transducer are the synthesized attributes. Then, the semi-relabelings are ssur and thus they can be composed with an attributed tree transducer to yield another attributed tree transducer. The fact that the class of output languages of semi-relabelings, denoted by $OUT(s\text{-}T)$, is equal to *RECOG* implies in particular that every recognizable tree language is the output language of a total deterministic, (super)linear top-down tree transducer. In [Dán96], a similar result was shown, namely, that the class of output languages of deterministic, superlinear (but *partial*) top-down tree transducers is equal to *RECOG*.

The proof of $OUT(s\text{-}T) = RECOG$ proceeds as follows. The inclusion $OUT(s\text{-}T) \subseteq RECOG$ follows from the fact that *RECOG* is closed under linear top-down tree transductions (cf. Corollary IV.6.6 of [GS84]). It remains to show that $RECOG \subseteq OUT(s\text{-}T)$. Let L be a recognizable tree language and let M be a top-down finite tree automaton which has L as domain and output. By encoding information into the input, a deterministic top-down tree transducer M' is constructed from M which has L as output. All restrictions of a semi-relabeling are fulfilled by M', except for totality. To turn M' into a semi-relabeling we show the following general result on the generating power of deterministic top-down tree transducers: the class of output languages of (partial) deterministic top-down tree transducers is equal to the class of output languages of *total* deterministic top-down tree transducers. This result contributes to the area of output languages of top-down tree transducers (see, e.g., [Eng75, Eng76, ERS80, Eng82, GS84, Dán96]).

Next we show in this paper that the classes of output languages of attributed tree transducers and the classes of output languages of ssur attributed tree transducers both form hierarchies with respect to the number of attributes. This strengthens a result of [KV94]. To show this we make use of the equivalence of the term-generating power of attribute grammars and context-free hypergraph grammars that was already mentioned above and prove a hierarchy of the classes of tree languages generated by context-free hypergraph grammars with respect to their rank.

This paper is organized as follows. Section 2 fixes some notions and notations used throughout the paper. Top-down tree transducers, attributed tree transducers, and attribute grammars are defined in Section 3. The results on the generating power of top-down tree transducers are established in Section 4. The comparison of the term-generating power of attribute grammars and attributed tree transducers is given in Section 5. The two hierarchy results are given in Section 6 and finally Section 7 lists some further research topics.

2. PRELIMINARIES

2.1. General Notations

The *set of nonnegative integers* is denoted by $\mathbb{N}$. The *empty set* is denoted by $\varnothing$. For $j \in \mathbb{N}$, $[j]$ denotes the set $\{1, ..., j\}$; therefore $[0] = \varnothing$. For a set A, the *set of words over* A is denoted by A^*. The empty word is denoted by ε. The *power set of* A, denoted by $\mathscr{P}(A)$, is the set of all subsets of A. The *cardinality of* A is denoted by $|A|$. For two sets A and B, the set $A - B$ denotes the set $\{a \in A \mid a \notin B\}$.

Let $f \subseteq A \times B$ be a relation. The *domain of* f, denoted by $dom(f)$, is the set $\{a \in A \mid (a, b) \in f \text{ for some } b \in B\}$. The *range of* f, denoted by $ran(f)$, is the set $\{b \in B \mid (a, b) \in f \text{ for some } a \in A\}$. Let $f \subseteq A \times B$ and $g \subseteq B \times C$ be relations. The *composition* $f \circ g \subseteq A \times C$ is defined by $(a, c) \in f \circ g$, if $a \in A$, $c \in C$, and there is a $b \in B$ such that $(a, b) \in f$ and $(b, c) \in g$. Note that we will view a partial function also as a relation.

Let $\Rightarrow \subseteq A \times A$ be a binary relation on A. Then, an element $a \in A$ is called *irreducible* (*with respect to* $\Rightarrow$) if there is no $a' \in A$ such that $a \Rightarrow a'$. If, for $b \in A$, there is an $a \in A$ such that $b \Rightarrow {}^* a$ and a is irreducible, then a is called a *normal form* (*with respect to* $\Rightarrow$) *of* b. If there is exactly one normal form of b, then it is denoted by $nf(\Rightarrow, b)$.

2.2. Ranked Alphabets, Variables, and Trees

A pair $(\Delta, rank_\Delta)$ is called *ranked alphabet* if Δ is a finite set and $rank_\Delta: \Delta \to \mathbb{N}$ is a total function. For every symbol $f \in \Delta$, the natural number $rank_\Delta(f)$ is called the *rank of* f. For every $i \geqslant 0$, the set $\Delta^{(i)} \subseteq \Delta$ consists of all symbols in Δ that are of rank i. A ranked alphabet can be specified by either (i) enumerating the finitely many nonempty subsets $\Delta^{(i)}$, or by (ii) decorating the symbols of Δ by their unambiguous rank (to be precise, by presenting $f \in \Delta^{(n)}$ as $f^{(n)}$). In this paper we will mostly use the latter of these methods. Then, it will always be clear from the context, if we mean a set of symbols or a ranked alphabet, when we are referring to Δ. If, for a ranked alphabet Δ, $\Delta = \Delta^{(1)}$, then Δ is called *unary*.

For the rest of this paper we choose the *set of variables* to be the set $X = \{x_1, x_2, ...\}$. The set $\{x_1, ..., x_k\}$ with $k \in \mathbb{N}$ is denoted by X_k.

Let S be an arbitrary set and let Δ be a ranked alphabet. The *set of trees over Δ indexed by S*, denoted by $T_\Delta(S)$, is the smallest set $T \subseteq (\Delta \cup S \cup \{(,), ,\})^*$, such that the following holds: (i) $S \subseteq T$ and (ii) if $f \in \Delta^{(n)}$ with $n \geqslant 0$ and $t_1, ..., t_n \in T$, then $f(t_1, ..., t_n) \in T$. For $\alpha \in \Delta^{(0)}$ we also denote the tree $\alpha()$ by α. If $S = \varnothing$, then we simply write T_Δ. If $S = X$, then $T_\Delta(X)$ is the set of trees (over Δ) with variables.

In the following, let Δ be a ranked alphabet and let S be a set. For every tree $t \in T_\Delta(S)$, the *set of occurrences* (or, *nodes*) *of t*, denoted by $O(t)$, is a subset of $\mathbb{N}^*$ which is inductively defined as follows: (i) if $t = x$ with $x \in S$, then $O(t) = \{\varepsilon\}$ and (ii) if $t = f(t_1, ..., t_n)$ with $f \in \Delta^{(n)}$ and $n \geqslant 0$, and for all $i \in [n] : t_i \in T_\Delta(S)$, then $O(t) = \{\varepsilon\} \cup \bigcup_{i \in [n]} \{iu \mid u \in O(t_i)\}$. Thus, the occurrence ε denotes the root of a tree. For an occurrence u we let $u0$ denote u. In particular this means that 0 denotes the occurrence ε.

For every tree $t \in T_\Delta(S)$ and every occurrence u of t, the *subtree of t at occurrence u* is denoted by t/u. The *label of t at occurrence u* is denoted by $t[u]$. The tree t with the subtree at occurrence u replaced by the tree s is denoted by $t[u \leftarrow s]$. The tree $t[t' \leftarrow s]$ with $t' \in T_\Delta(S)$ denotes the tree $t[u_1 \leftarrow s] \cdots [u_n \leftarrow s]$ where $u_1, ..., u_n$ are all occurrences u of t with $t/u = t'$. When using the notation $t[x \leftarrow s]$, it will always be clear from the context whether x is an occurrence or a subtree. Also, we use —similar to the set notation—the notation $[x \leftarrow s | P_1, ..., P_n]$, where $P_1, ..., P_n$ are conditions on x and/or s, to abbreviate the composition of the substitutions $[x \leftarrow s]$ for which the conditions P_i with $i \in [n]$ hold. This notation is used only if the composition does not depend on the order of the substitutions.

2.3. Tree Translations

Let Σ and Δ be ranked alphabets. A relation $\tau \subseteq T_\Sigma \times T_\Delta$ is called *tree translation* or simply translation. Let $\tau: T_\Sigma \to T_\Delta$ be a translation and let $L \subseteq T_\Sigma$ be a tree language. Then $\tau(L)$ denotes the set $\{t \in T_\Delta \mid (s, t) \in \tau \text{ for some } s \in L\}$. For a class Ψ of tree translations we denote by $OUT(\Psi)$ the corresponding class of output languages, i.e., $OUT(\Psi) = \{ran(\tau) \mid \tau \in \Psi\}$. Let Ψ and Φ be classes of tree translations which are all functions. The composition $\Psi \circ \Phi$ is defined by $\{\psi \circ \varphi \mid \psi \in \Psi, \varphi \in \Phi\}$.

3. TOP-DOWN TREE TRANSDUCERS, ATTRIBUTED TREE TRANSDUCERS, AND ATTRIBUTE GRAMMARS

We assume the reader to be familiar with the basic properties of top-down tree transducers, attributed tree transducers, and attribute grammars. In this section we will shortly recall the definitions and concepts that are relevant for this paper. Before we define top-down tree transducers let us define the set of right-hand sides of rules of top-down tree transducers.

DEFINITION 3.1 (Right-hand sides of top-down tree transducers). Let Q and Δ be disjoint ranked alphabets, where Q is unary, and let $k \in \mathbb{N}$. The *set of right-hand*

sides over Q, Δ, *and* k, denoted by $RHS(Q, \Delta, k)$, is the smallest subset RHS of $T_{Q \cup \Delta}(X_k)$ such that (i) for $q \in Q$ and $i \in [k]$ the tree $q(x_i)$ is in RHS, and (ii) for $\sigma \in \Sigma^{(n)}$ with $n \in \mathbb{N}$ and $t_1, ..., t_n \in RHS$, the tree $\sigma(t_1, ..., t_n)$ is in RHS.

DEFINITION 3.2 (Top-down tree transducer). A *top-down tree transducer* M is a tuple $(Q, \Sigma, \Delta, q_0, R)$, where

- Q is a unary ranked alphabet; the elements of Q are called *states*.
- Σ and Δ are ranked alphabets which are disjoint with Q; Σ is called *input alphabet* and elements of Σ are called *input symbols*, Δ is called *output alphabet* and elements of Δ are called *output symbols*.
- q_0 is a state in Q, called the *initial state*.
- R is a finite set of *rules* of the following form: $q(\sigma(x_1, ..., x_k)) \to \zeta$, with $q \in Q$, $k \geqslant 0$, $\sigma \in \Sigma^{(k)}$, and $\zeta \in RHS(Q, \Delta, k)$. Such a rule is called (q, σ)-*rule* or just σ-rule. If for a $q \in Q$ and a $\sigma \in \Sigma$ there is exactly one rule of the above form, then the right-hand side ζ is denoted by $rhs_M(q, \sigma)$ (M is dropped if it is clear from the context).

DEFINITION 3.3 (Derivation relation induced by M). Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a top-down tree transducer. The *derivation relation induced by* M, denoted by $\Rightarrow_M$, is a binary relation over $T_{Q \cup \Sigma \cup \Delta}$, such that $s_1 \Rightarrow_M s_2$, with $s_1, s_2 \in T_{Q \cup \Sigma \cup \Delta}$, if there are $(q(\sigma(x_1, ..., x_k)) \to \zeta) \in R$, $t_1, ..., t_k \in T_\Sigma$, and an occurrence $u \in O(s_1)$ with $s_1/u = q(\sigma(t_1, ..., t_k))$ such that

$$s_2 = s_1[u \leftarrow \zeta[x_i \leftarrow t_i \mid i \in [k]]].$$

Where appropriate we might, for a particular derivation step, also add the occurrence u or the (q, σ)-rule, or the pair (q, σ) as index for $\Rightarrow_M$.

Note that the derivation relation of Definition 3.3 is only defined, if for $\delta \in \Sigma^{(m)}$ and $\delta \in \Delta^{(n)}$, $m = n$. This is a (minor) technical problem that we disregard.

A top-down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$ is a device that translates trees over Σ into trees over Δ. Thus, M induces a binary relation $\tau_M \subseteq T_\Sigma \times T_\Delta$ which is called the translation realized by M.

DEFINITION 3.4 (Translation realized by M, *out*, *dom*). Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a top-down tree transducer. The *translation realized by* M, denoted by τ_M, is the relation $\{(s, t) \in T_\Sigma \times T_\Delta \mid q_0(s) \Rightarrow_M^* t\}$.

The *class of all translations* that can be realized by top-down tree transducers is denoted by T.

The *domain of* M, denoted by $dom(M)$, is the set $dom(\tau_M)$. The *output language of* M, denoted by $out(M)$, is the set $ran(\tau_M)$.

Let us now define various subclasses of top-down tree transducers. Note that we define top-down finite tree automata in the context of tree transducers (cf., e.g., Definition 3.1.7 of [ERS80]); i.e., there is no transition function as it is common for automata. Thus, strictly speaking, our top-down finite tree automata are not really accepting devices, but rather translational devices which realize the identity on those trees which are accepted.

DEFINITION 3.5 (Subclasses of T). Let $M=(Q, \Sigma, \Delta, q_0, R)$ be a top-down tree transducer.

- If, for every $q \in Q$ and $\sigma \in \Sigma$, there is *at most one* (q, σ)-rule in R, then M is called *deterministic*. The class of translations which can be realized by deterministic top-down tree transducers is denoted by *d-T*.
- If, for every $q \in Q$ and $\sigma \in \Sigma$, there is *at least one* (q, σ)-rule in R, then M is called *total*. The class of translations which can be realized by total top-down tree transducers is denoted by *t-T*.
- If, in each right-hand side of the rules in R, every variable occurs at most once, then M is called *linear*. The corresponding class of translations is denoted by *l-T*.
- If $\Sigma=\Delta$ and every rule in R has the form $q(\sigma(x_1, ..., x_k)) \to \sigma(q_1(x_1), ..., q_k(x_k))$ with $q, q_1, ..., q_k \in Q$ and $\sigma \in \Sigma^{(k)}$, then M is called *top-down finite tree automaton.*

If a combination of these restrictions holds, then the corresponding class of translations is denoted by appending the prefixes. For example, the class of translations realized by total, deterministic top-down tree transducers is denoted by *td-T*.

Recall that if M is a deterministic top-down tree transducer, then the translation τ_M is a function from T_Σ to T_Δ and if M is a total deterministic top-down tree transducer, then the translation τ_M is a total function (see, e.g., [FHVV93]).

The class of domains (or, equivalently: output languages) of top-down finite tree automata is called the *class of recognizable tree languages* and is denoted by *RECOG*. Note that the input or output alphabet of a top-down tree transducer may be empty and this implies that $\varnothing \in OUT(\Psi)$ for all classes Ψ of translations induced by top-down tree transducers that we consider here (and in particular for the case of total top-down tree transducers). The reason that we allow this is that we will compare classes of output languages of top-down tree transducers, in particular of total ones, with the class of recognizable tree languages and that $\varnothing \in RECOG$.

Let us now define a property of top-down tree transducers.

DEFINITION 3.6 (Reduced). Let $M=(Q, \Sigma, \Delta, q_0, R)$ be a deterministic top-down tree transducer. M is called *reduced*, if either $\Sigma=\varnothing$ or for every state $q \in Q$, there are $s, s' \in T_\Sigma$, $\xi \in T_{Q \cup \Sigma \cup \Delta}$, and a $t \in T_\Delta$, such that $q(s')$ occurs in ξ and $q_0(s) \Rightarrow_M^* \xi \Rightarrow_M^* t$, i.e., Q does not contain superfluous states.

Observation 3.7. For every deterministic top-down tree transducer M there is a deterministic top-down tree transducer M' which is reduced and realizes the same translation as M. M' can be obtained from M by simply deleting the superfluous states and the rules containing them.

Since we will only need this fact for deterministic, linear top-down tree transducers, we only discuss it in detail for the linear case. Let $M=(Q, \Sigma, \Delta, q_0, R)$ be a deterministic, linear top-down tree transducer. Let $Q_{prod}=\{q \in Q \mid dom(M_q) \neq \varnothing\}$, where for every $q \in Q$, M_q is the top-down tree transducer $(Q, \Sigma, \Delta, q, R)$.

Note that Q_{prod} can be constructed effectively, because the domain of a deterministic top-down tree transducer is recognizable (Theorem 1 of [Rou70]). For every $q \in Q$ and $\sigma \in \Sigma$ let $Q(q, \sigma) = \varnothing$ if there is no (q, σ)-rule in R and otherwise let $Q(q, \sigma) = \{q' \in Q \mid q' \text{ occurs in } rhs_M(q, \sigma)\}$. If $q_0 \notin Q_{prod}$, then $M' = (Q, \varnothing, \Delta, q_0, \varnothing)$ and otherwise let F be defined by the following procedure.

$F' := \{q_0\}$
repeat
  $F := F'$
  $F' := F \cup \{q' \in Q(q, \sigma) \mid q \in F \text{ and } \sigma \in \Sigma \text{ with } Q(q, \sigma) \subseteq Q_{prod}\}$
until $F = F'$.

Clearly this procedure terminates, because $F \subseteq F' \subseteq Q$. Now define $M' = (F, \Sigma, \Delta, q_0, R')$ where R' consists of all (q, σ)-rules in R for which $q \in F$ and $Q(q, \sigma) \subseteq F$. It should be clear that M' realizes the same translation as M and that M' is reduced. For the nonlinear case a similar but more complicated construction can be applied.

Let us now turn to attributed tree transducers. In fact, we only consider total deterministic attributed tree transducers. Before we define the notion of an attributed tree transducer let us define the set of right-hand sides of its rules.

DEFINITION 3.8 (Right-hand sides of attributed tree transducers). Let S and I be unary ranked alphabets, Δ a ranked alphabet, and $k \geqslant 0$ an integer. The set of *right-hand sides over* S, I, Δ, *and* k, denoted by $RHS(S, I, \Delta, k)$, is the smallest subset RHS of $T_{S \cup I \cup \Delta}(\{\pi, \pi 1, ..., \pi k\})$ such that the following conditions hold:

(i) For every $a \in S$ and $1 \leqslant i \leqslant k$, the tree $a(\pi i)$ is in RHS.

(ii) For every $b \in I$, the tree $b(\pi)$ is in RHS.

(iii) For every $\delta \in \Delta^{(l)}$ with $l \geqslant 0$ and $\xi_1, ..., \xi_l \in RHS$, the tree $\delta(\xi_1, ..., \xi_l)$ is in RHS.

The symbol π is called *occurrence variable*.

During the computation of an attributed tree transducer the occurrence variable π will be replaced by actual occurrences of the input tree.

DEFINITION 3.9 (Attributed tree transducer). An *attributed tree transducer* A is a tuple $(Syn, Inh, \Sigma, \Delta, root, a_0, R)$, where

- Syn and Inh are disjoint, unary ranked alphabets, the elements of which are called *synthesized attributes* and *inherited attributes*, respectively.
- Σ and Δ are ranked alphabets which are disjoint with Syn and Inh; Σ is called *input alphabet* and elements of Σ are called *input symbols*, Δ is called *output alphabet* and elements of Δ are called *output symbols*.
- $root$ is a symbol of rank 1 with $root \notin \Sigma$, called the *root marker*.
- a_0 is a synthesized attribute, called the *initial attribute*.

- $R = \bigcup_{\sigma \in \Sigma \cup \{root\}} R_\sigma$ is a finite set of *rules*, such that

1. The set R_{root} contains exactly one rule of the form $a_0(\pi) \to \zeta$ with $\zeta \in RHS(Syn, \varnothing, \Delta, 1)$ and for every $b \in Inh$ the set R_{root} contains exactly one rule of the form $b(\pi 1) \to \zeta$ with $\zeta \in RHS(Syn, \varnothing, \Delta, 1)$.

2. For every $\sigma \in \Sigma^{(k)}$ with $k \geqslant 0$ and $a \in Syn$, the set R_σ contains exactly one rule of the form $a(\pi) \to \zeta$ with $\zeta \in RHS(Syn, Inh, \Delta, k)$.

For every $\sigma \in \Sigma^{(k)}$ with $k \geqslant 0$, $b \in Inh$, and $i \in [k]$, the set R_σ contains exactly one rule of the form $b(\pi i) \to \zeta$ with $\zeta \in RHS(Syn, Inh, \Delta, k)$.

Let $Att = Syn \cup Inh$. A particular rule $\alpha(p) \to \zeta$ in R_σ with $\sigma \in (\Sigma \cup \{root\})^{(k)}$, $\alpha \in Att$, and $p \in \{\pi, \pi 1, ..., \pi k\}$ is called (α, p, σ)-rule; the tree ζ is denoted by $rhs_A(\alpha, p, \sigma)$ (A is dropped if it is clear from the context). We might also denote ζ by $rhs(\alpha(p) \to \zeta)$.

For every $\sigma \in \Sigma^{(k)}$, the *set of inside attributes of* σ, denoted by ins_σ, is the set $\{\alpha(\pi) \mid \alpha \in Syn\} \cup \{\beta(\pi i) \mid \beta \in Inh, i \in [k]\}$, and the set of *outside attributes of* σ, denoted by $outs_\sigma$, is the set $\{\beta(\pi) \mid \beta \in Inh\} \cup \{\alpha(\pi i) \mid \alpha \in Syn, i \in [k]\}$. For the *root* marker, $ins_{root} = \{a_0(\pi)\} \cup \{b(\pi 1) \mid b \in Inh\}$ and $outs_{root} = \{a(\pi 1) \mid a \in Syn\}$.

The set $\{root(s) \mid s \in T_\Sigma\}$ is called the *set of control trees.*

Note that our definition of attributed tree transducers in Definition 3.9 is different from the original definition in [Fül81]. There, for every inherited attribute b, the right-hand side of the $(b, \pi 1, root)$-rule is restricted to trees over Δ. In the appendix of [Gie88] this difference was pointed out and the term *full attributed tree transducer* was used to refer to the tree transducers of our Definition 3.9. However, in the following we will simply use the notion of attributed tree transducer as defined.

DEFINITION 3.10 (Derivation relation induced by A). Let $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be an attributed tree transducer and let s be a tree in $T_{\Sigma \cup \{root\}}$. The *derivation relation induced by A on s*, denoted by $\Rightarrow_{A,s}$, is the binary relation over $T_{\Delta \cup Att}(O(s))$, such that $\xi_1 \Rightarrow_{A,s} \xi_2$ for $\xi_1, \xi_2 \in T_{\Delta \cup Att}(O(s))$, if there is an attribute $\alpha \in Att$ and occurrences $v \in O(s)$, $u \in O(\xi_1)$ such that $\xi_1/u = \alpha(v)$ and

- either $\alpha \in Syn$, $s[v] = \sigma$, with $\sigma \in (\Sigma \cup \{root\})^{(k)}$, $k \geqslant 0$, and

$$\xi_2 = \xi_1[u \leftarrow \zeta] \qquad \text{with} \quad \zeta = rhs(\alpha, \pi, \sigma)[\beta(\pi i) \leftarrow \beta(vi) \mid \beta \in Att, 0 \leqslant i \leqslant k]$$

- or $\alpha \in Inh$ and $v = \bar{v}i$ for some $\bar{v} \in O(s)$, $s[\bar{v}] = \sigma$ with $\sigma \in (\Sigma \cup \{root\})^{(k)}$, $k \geqslant 0$, $i \in [k]$, and

$$\xi_2 = \xi_1[u \leftarrow \zeta] \qquad \text{with} \quad \zeta = rhs(\alpha, \pi i, \sigma)[\beta(\pi i) \leftarrow \beta(\bar{v}i) \mid \beta \in Att, 0 \leqslant i \leqslant k],$$

where $\pi 0$ denotes π (this is in accordance with the convention that $v0$ denotes v for an occurrence v, as defined in the preliminaries).

In the same sense as for attribute grammars, attributed tree transducers can be *circular* (see [Fül81] for the precise definition of circularity for attributed tree transducers). However, in the rest of this paper we always mean *noncircular*

attributed tree transducers when referring to attributed tree transducers. The same is true when referring to attribute grammars, to be defined in Definition 3.14.

If an attributed tree transducer is noncircular, then the derivation relation on any tree s in $T_{\Sigma \cup \{root\}}$ is confluent and terminating (cf. Lemmas 4.8 and 4.9 of [FHVV93]). Thus, every tree in $T_{\Delta \cup Att}(O(s))$ has a unique normal form [New42]. If s is a control tree, then $a_0(\varepsilon)$ has a normal form in T_Δ.

Let us now define the translation realized by an attributed tree transducer. It is a total function from T_Σ to T_Δ (where Σ and Δ are the input and output alphabet of the transducer, respectively).

DEFINITION 3.11 (Translation realized by A, *out*). Let $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be an attributed tree transducer. The *translation realized by* A, denoted by τ_A, is the set

$$\{(s, t) \mid s \in T_\Sigma, t \in T_\Delta \quad \text{and} \quad t = nf(\Rightarrow_{A, root(s)}, a_0(\varepsilon))\}.$$

The *output language of* A, denoted by $out(A)$, is the set $ran(\tau_A)$. The class of translations realized by attributed tree transducers is denoted by AT. For $k \in \mathbb{N}$, the class of translations realized by attributed tree transducers of which the number of attributes $|Att|$ is less than or equal to k, is denoted by AT_k.

Note again that the input or output alphabet of an attributed tree transducer may be empty and therefore $\varnothing \in OUT(AT)$.

As mentioned before, "attributed tree transducer" always means "noncircular attributed tree transducer", and so AT is the class of translations which can be realized by noncircular attributed tree transducers. However, to compare the class of output languages of attributed tree transducers with that of attribute grammars we have to consider attributed tree transducers which are *noncircular on* L, where L is a tree language taken as input for an attributed tree transducer. This means that they might be circular on input trees which are not in L. By definition, the class AT contains only translations that can be realized by noncircular attributed tree transducers. Let us therefore define the class AT_{all} which contains the translations that can be computed by arbitrary attributed tree transducers (of course, not every τ in AT_{all} is a total function anymore). The definition of the translation τ_A realized by a circular attributed tree transducer A can be taken over literally from Definition 3.11; the normal form $nf(\Rightarrow_{A, root(s)}, a_0(\varepsilon))$ exists iff A is noncircular on $root(s)$. Let $\mathscr{L}$ be a class of tree languages. The class of output languages of attributed tree transducers taking languages of $\mathscr{L}$ as input, denoted by $AT(\mathscr{L})$, is the class $\{\tau_A(L) \mid L \in \mathscr{L}, \tau_A \in AT_{all}$ such that A is noncircular on $L\}$. Indeed, we will use this notion merely for $\mathscr{L} = RECOG$ (cf. the discussion of $AT(RECOG)$ and $OUT(AT)$ in the Introduction).

Observation 3.12. Let $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be a (possibly circular) attributed tree transducer which is noncircular on s, where $s = \sigma(s_1, ..., s_k)$, $\sigma \in (\Sigma \cup \{root\})^{(k)}$, $s_1, ..., s_k \in T_\Sigma$, and $k \geqslant 0$. Let $a \in Syn$ (with $a = a_0$ if $\sigma = root$), $b \in Inh$, and $j \in [k]$.

Then $nf(\Rightarrow_{A, s}, a(\varepsilon))$ and $nf(\Rightarrow_{A, s}, b(j))$ are trees in $T_{\Delta \cup Inh}(\{\varepsilon\})$ which can be obtained from $rhs(a, \pi, \sigma)$ and $rhs(b, \pi j, \sigma)$, respectively, by replacing every outside

attribute by the corresponding normal form. Making use of the derivation relations $\Rightarrow_{A, s_i}$ for $i \in [k]$, this means that in $rhs(a, \pi, \sigma)$ and $rhs(b, \pi j, \sigma)$, respectively, every $\alpha(\pi i)$ with $\alpha \in Syn$ and $i \in [k]$ is replaced by the term

$$nf(\Rightarrow_{A, s_i}, \alpha(\varepsilon))[\beta(\varepsilon) \leftarrow nf(\Rightarrow_{A, s}, \beta(i)) \mid \beta \in Inh]$$

and every $\beta(\pi)$ with $\beta \in Inh$ is replaced by $\beta(\varepsilon)$.

Let us now give a small example of an attributed tree transducer.

EXAMPLE 3.13. Let $A_{bin} = (Syn, Inh, \Sigma, \Delta, root, a, R)$ be the attributed tree transducer defined as follows: $Syn = \{a\}$, $Inh = \{b\}$, $\Sigma = \{1^{(1)}, 0^{(1)}, e^{(0)}\}$, $\Delta = \{+^{(2)}, exp^{(1)}, s^{(1)}, 0^{(0)}\}$ (note that in the following we use infix notation for the $+$ symbol), $R = R_{root} \cup R_1 \cup R_0 \cup R_e$, where

$$R_{root} = \{a(\pi) \to a(\pi 1), b(\pi 1) \to 0\},$$
$$R_1 = \{a(\pi) \to exp(b(\pi)) + a(\pi 1), b(\pi 1) \to s(b(\pi))\},$$
$$R_0 = \{a(\pi) \to a(\pi 1), b(\pi 1) \to s(b(\pi))\},$$
$$R_e = \{a(\pi) \to 0\}.$$

This attributed tree transducer takes a binary number represented as monadic tree over the ranked alphabet Σ (with the least significant bit at the root) as input and produces its decimal value represented as tree over Δ. In Fig. 1 we can see what kind of tree traversal over the input tree $\xi = 1(0(1(e)))$, representing the binary number 101, produces the output tree t, which represents the decimal number 5, if $s(x)$ is interpreted as $x + 1$ and $exp(x)$ is interpreted as 2^x.

Let us also take a look at a derivation by $\Rightarrow_{A_{bin}, root(\xi)}$. Let the substitutions $[\beta(\pi i) \leftarrow \beta(i) \mid \beta \in Att, 0 \leqslant i \leqslant 1]$ and $[\beta(\pi i) \leftarrow \beta(1i) \mid \beta \in Att, 0 \leqslant i \leqslant 1]$ be denoted by Θ and Θ', respectively.

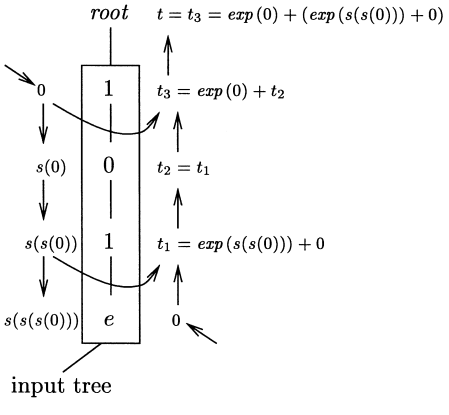

FIG. 1. Evaluation of attributes for A_{bin}.

$$a(\varepsilon) \Rightarrow_{A, root(\xi)} a(\varepsilon)[\varepsilon \leftarrow \underbrace{\underbrace{rhs(a, \pi, root)}_{a(\pi 1)} \Theta}_{a(1)}] = a(1)$$

$$\Rightarrow_{A, root(\xi)} a(1)[\varepsilon \leftarrow \underbrace{\underbrace{rhs(a, \pi, 1)}_{exp(b(\pi)) + a(\pi 1)} \Theta'}_{exp(b(1)) + a(11)}] = exp(b(1)) + a(11)$$

$$\Rightarrow_{A, root(\xi)} (exp(b(1)) + a(11))[1 \leftarrow rhs(b, \pi 1, root)\Theta] = exp(0) + a(11)$$

$$\Rightarrow^*_{A, root(\xi)} exp(0) + (exp(s(s(0))) + 0) = t$$

We will now very shortly define the notion of attribute grammars. For more details see [Knu68, DJL88]. We mostly refer to the notations used in [EH92]. To do so, let us first define the notion of context-free (string) grammars and that of semantic domains.

A *context-free* (*string*) *grammar* G is a tuple (N, T, S, P), where N is a finite set; the elements of N are called *nonterminals*. T is a finite set; the elements of T are called *terminals*. $S \in N$ is called *initial nonterminal*. P is a finite set of *productions* of the form $X \to \zeta$, where $X \in N$ and $\zeta \in (N \cup T)^*$.

A *semantic domain* is a pair $D = (V, \Gamma)$, where V is a set of values, Γ is a ranked alphabet, and each $\gamma \in \Gamma$ denotes a total function $\gamma_D: V^n \to V$ with $n = rank_\Gamma(\gamma)$; i.e., D is a Γ-algebra, where Γ is a one-sorted signature (see, e.g., [CF82]).

DEFINITION 3.14 (Attribute grammar). An *attribute grammar* G is a tuple (G_0, D, B, R), where

- $G_0 = (N, T, S, P)$ is a context-free grammar, called the *underlying grammar of* G.
- $D = (V, \Gamma)$ is a semantic domain.
- $B = (Syn, Inh, att, a_0)$, where Syn and Inh are finite, disjoint sets the elements of which are called *synthesized* and *inherited attributes*, respectively; and att: $N \to \mathscr{P}(Syn \cup Inh)$ is a mapping which associates attributes with the nonterminals of G_0, such that $a_0 \in att(S)$ and $att(S) \subseteq Syn$. The attribute $a_0 \in Syn$ is called the *designated attribute*.
- $R = (R(p) \mid p \in P)$ is a family of sets of *semantic rules*, such that for every production $p = (X_0 \to w_0 X_1 w_1 \cdots X_n w_n) \in P$, with $X_i \in N$ and $w_i \in T^*$ for $0 \leqslant i \leqslant n$, $R(p)$ is a finite set of semantic rules. For every $\langle a, j \rangle \in ins(p)$, $R(p)$ contains one rule of the form $\langle a, j \rangle = t$ with $t \in T_\Gamma(outs(p))$, where $ins(p) = \{\langle b, i \rangle \mid b \in (att(X_i) \cap Inh), i \in [n]\} \cup \{\langle a, 0 \rangle \mid a \in (att(X_0) \cap Syn)\}$ is the set of *inside attributes* of p and $outs(p) = \{\langle a, i \rangle \mid a \in (att(X_i) \cap Syn), i \in [n]\} \cup \{\langle b, 0 \rangle \mid b \in (att(X_0) \cap Inh)\}$ is the set of *outside attributes* of p.

Again, we will only consider noncircular attribute grammars. The evaluation of attributes is as usual. Let G be a noncircular attribute grammar and let t be a derivation tree of the underlying grammar G_0 of G. Consider an occurrence

$x \in O(t)$ with label $t[x] = X \in N$. The set of attribute instances at x is the set $\{\langle \alpha, x \rangle \mid \alpha \in att(X)\}$. Since G is noncircular, there is exactly one function $val_{G,t}$, called *decoration*, which associates values with the attribute instances of t, such that the semantic rules are respected by the values of the local attribute instances. The *output language generated by* G, denoted by $out(G)$, is $\{val_{G,t}(\langle a_0, \varepsilon \rangle) \mid t$ is a derivation tree of $G_0\}$.

According to [EH92] we say that an attribute grammar G with semantic domain $D = (V, \Gamma)$ is *term-generating*, if D is the free Γ-algebra, i.e., $V = T_\Gamma$. Thus, for a term-generating attribute grammar G the output language $out(G)$ is a subset of T_Γ. The class of output languages that can be generated by term-generating attribute grammars is denoted by $OUT(AG, TERMS)$. In Example 5.4 we will give a short example for a term-generating attribute grammar.

4. GENERATING POWER OF DETERMINISTIC TOP-DOWN TREE TRANSDUCERS

The main result of this section is that the generating power of a particular type of total deterministic, linear top-down tree transducer, called *semi-relabeling* (for short, *s-relabeling*), is exactly the class of recognizable tree languages. This is achieved by proving the following two results. First, it is shown that the generating power of deterministic top-down tree transducers and that of *total* deterministic top-down tree transducers is equal. Second, it is shown that the class of recognizable tree languages is included in the class of output languages of s-relabelings. This is done by proving that $RECOG$ is included in the class of output languages of a type of top-down tree transducer which satisfies all requirements of s-relabelings, except totality. Then the (proof of the) first result can be used to achieve totality, and thus to obtain s-relabelings. In fact to prove the main theorem, i.e., that $RECOG = OUT(s\text{-}T)$, it would be sufficient to prove the linear case of the first result, i.e., that $OUT(dl\text{-}T) = OUT(tdl\text{-}T)$ as it was shown in [Man96]. However, since the stronger result for the nonlinear case is an interesting result on the generating power of deterministic tree transducers we decided to present it here.

Let us now discuss the first of these results, i.e., $OUT(d\text{-}T) = OUT(td\text{-}T)$. By definition, $OUT(td\text{-}T) \subseteq OUT(d\text{-}T)$. Let us consider an (arbitrary) deterministic top-down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$. Suppose that for an input tree $s \in T_\Sigma$ the derivation by M stops at a tree ξ which is irreducible and contains subtrees of the form $q(\sigma(\ldots))$. Thus, in order to obtain a total deterministic top-down tree transducer M' which generates the same output language as M, we must construct the missing (q, σ)-rule for M'. If there is no $s_q \in T_\Sigma$ with $q(s_q) \Rightarrow_M^* t_q \in T_\Delta$, then the state q is not needed and can be removed. If there *is* such an s_q, then we can try to add the rule $q(\sigma(x_1, \ldots, x_k)) \to t_q$ to the rules of M'. This means that in the original input tree s the subtree $\sigma(\ldots)$ is replaced by s_q. However, due to copying there might be another state $q' \neq q$ which, in ξ, processes the same input subtree $\sigma(\ldots)$ as q. But, of course the tree $\sigma(\ldots)$ is different from s_q and thus, when adding rules of the form $q(\sigma(x_1, \ldots, x_k)) \to t_q$, we can only be sure that trees in $out(M)$ are derived by M', if q is the only state processing the input subtree $\sigma(\ldots)$, i.e., if M is

linear. Hence, for the nonlinear case we have to keep track of which states are processing the same input subtree. Thus, if $F \subseteq Q$ is the set of states processing the same input, then we must find a tree $s_F \in T_\Sigma$ such that

$$\text{for every} \quad q \in F : q(s_F) \Rightarrow_M^* t_{q,F} \qquad \text{with} \quad t_{q,F} \in T_\Delta. \tag{$*$}$$

We will use the states of M' to keep track of which states are processing the same input tree. More precisely, the states of M' will be pairs of the form $\langle q, F\rangle$, where $q \in Q$ is the state processing the particular input tree and F is a nonempty subset of Q consisting of *all* states which are processing this input tree. The set F models the *state set* at each node of the input tree (cf. Definition 3.1.8 of [ERS80]). Moreover, we only allow states $\langle q, F\rangle$ for which there exists an input tree s_F with $(*)$. Now consider a state $\langle q, F\rangle$ of M' and a $\sigma \in \Sigma$ such that there is no (q, σ)-rule in R. Then for every $q' \in F$ we let the rule $\langle q', F\rangle(\sigma(x_1, ..., x_k)) \to t_{q',F}$ be a rule of M'.

LEMMA 4.1. *For every deterministic top-down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$ there is a total deterministic top-down tree transducer $M' = (Q', \Sigma', \Delta, q_0', R')$ such that $out(M') = out(M)$. Moreover, if M is linear, then R' is of the form $R' = R_1 \cup R_2$ such that $R_1 \subseteq R$ and the right-hand side of every rule in R_2 is in T_Δ.*

Proof. Before we construct the total deterministic top-down tree transducer M' let us define some auxiliary notions which are needed in the proof.

Let $\sigma \in \Sigma^{(k)}$, $i \in [k]$, and let F be a nonempty subset of Q. Then $Q_{\sigma,i}(F)$ denotes the set $\{q' \in Q \mid q'(x_i)$ occurs in $rhs_M(q, \sigma)$ for a $q \in F\}$.

Let F be a nonempty subset of Q. Now define $Inp(F) = \bigcap_{q \in F} dom(M_q)$ where for every $q \in Q$ the deterministic top-down tree transducer M_q is defined as $(Q, \Sigma, \Delta, q, R)$. If $Inp(F) \neq \emptyset$, then let s_F be an arbitrary but fixed tree in $Inp(F)$. For every $q \in F$ let $t_{q,F}$ be the tree in T_Δ such that $q(s_F) \Rightarrow_M^* t_{q,F}$. Note that it is decidable whether $Inp(F) = \emptyset$, because $Inp(F)$ is a recognizable tree language. This follows from the facts that the domain of a deterministic top-down tree transducer is recognizable (see Theorem 1 of [Rou70]) and that recognizable tree languages are closed under intersection.

Let us now define M'. If $Inp(\{q_0\}) = \emptyset$, then $Q' = Q$, $\Sigma' = \emptyset$, $q_0' = q_0$, and $R' = \emptyset$. Otherwise let $Q' = \{\langle q, F\rangle \mid F \subseteq Q, q \in F, Inp(F) \neq \emptyset\}$, $\Sigma' = \Sigma$, $q_0' = \langle q_0, \{q_0\}\rangle$, and let R' be the set of rules defined as follows. For $\langle q, F\rangle \in Q'$ and $\sigma \in \Sigma^{(k)}$ let the rule

$$\langle q, F\rangle(\sigma(x_1, ..., x_k)) \to \zeta$$

be in R', such that

- if for every $q' \in F$ there is a (q', σ)-rule in R and for every $i \in [k]$: $Q_{\sigma,i}(F) \neq \emptyset$ implies $Inp(Q_{\sigma,i}(F)) \neq \emptyset$, then

$$\zeta = rhs_M(q, \sigma)[q'(x_i) \leftarrow \langle q', Q_{\sigma,i}(F)\rangle(x_i) \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]]$$

and

- otherwise $\zeta = t_{q,F}$.

To show the correctness of the construction we must show that $out(M) = out(M')$. Let us assume that $out(M) \neq \emptyset$, because otherwise $Inp(\{q_0\}) = \emptyset$ and so $out(M') = out(M)$, and the second statement of Lemma 4.1 clearly holds. Let us first show that $out(M) \subseteq out(M')$. For $F = \{q_0\}$ Claim 1 proves this inclusion.

Claim 1. Let F be a nonempty subset of Q. Let $s \in T_\Sigma$ and let $t_q \in T_\Delta$ for every $q \in F$. If $q(s) \Rightarrow_M^* t_q$ for every $q \in F$, then $\langle q, F\rangle(s) \Rightarrow_{M'}^* t_q$ for every $q \in F$.

This claim is proved by induction on the structure of s. Let $s = \sigma(s_1, ..., s_k)$ with $\sigma \in \Sigma^{(k)}$, $s_1, ..., s_k \in T_\Sigma$, and $k \geqslant 0$. Now let $q \in F$. Since there is a derivation $q(s) \Rightarrow_M^* t_q$ by M, $\zeta_q = rhs_M(q, \sigma)$ exists and for every $q' \in Q$ and $i \in [k]$ such that $q'(x_i)$ occurs in ζ_q (i.e., for every $q' \in Q_{\sigma,i}(\{q\})$) a tree $t_{q',i} \in T_\Delta$ exists such that $q'(s_i) \Rightarrow_M^* t_{q',i}$ and $t_q = \zeta_q[q'(x_i) \leftarrow t_{q',i} \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]]$.

Thus, by induction hypothesis, for every nonempty set $Q_{\sigma,i}(F)$ with $i \in [k]$ and for every $q' \in Q_{\sigma,i}(F)$: $\langle q', Q_{\sigma,i}(F)\rangle(s_i) \Rightarrow_{M'}^* t_{q',i}$. Then

$$\begin{aligned}\langle q, F\rangle(\sigma(s_1, ..., s_k)) &\Rightarrow_{M'} \zeta_q[q'(x_i) \leftarrow \langle q', Q_{\sigma,i}(F)\rangle(s_i) \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]] \\ &\Rightarrow_{M'}^* \zeta_q[q'(x_i) \leftarrow t_{q',i} \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]] = t_q,\end{aligned}$$

which proves the claim.

It remains to show that $out(M') \subseteq out(M)$. This follows from Claim 2 with $F = \{q_0\}$.

Claim 2. Let F be a nonempty subset of Q with $Inp(F) \neq \emptyset$. Let $s \in T_\Sigma$ and let $t_q \in T_\Delta$ for every $q \in F$. If $\langle q, F\rangle(s) \Rightarrow_{M'}^* t_q$ for every $q \in F$, then there exists $\tilde{s} \in T_\Sigma$ such that for every $q \in F$, $q(\tilde{s}) \Rightarrow_M^* t_q$.

Again we prove this by induction on the structure of s. Let $s = \sigma(s_1, ..., s_k)$ with $\sigma \in \Sigma^{(k)}$, $s_1, ..., s_k \in T_\Sigma$, and $k \geqslant 0$.

Case 1. There is a $q' \in F$ such that there is no (q', σ)-rule in R, or there is an $i \in [k]$ such that $Q_{\sigma,i}(F) \neq \emptyset$ and $Inp(Q_{\sigma,i}(F)) = \emptyset$. Then take $\tilde{s} = s_F$. By the definition of s_F and $t_{q,F}$ and the rules of M', $q(\tilde{s}) = q(s_F) \Rightarrow_M^* t_{q,F} = t_q$ for every $q \in F$.

Case 2. For every $q' \in F$ there is a (q', σ)-rule in R and for every $i \in [k]$: if $Q_{\sigma,i}(F) \neq \emptyset$, then $Inp(Q_{\sigma,i}(F)) \neq \emptyset$. Let $q \in F$ and let $\zeta_q = rhs_M(q, \sigma)$. Then by the definition of the rules of M', $\langle q, F\rangle(s) \Rightarrow_{M'} \zeta_q[q'(x_i) \leftarrow \langle q', Q_{\sigma,i}(F)\rangle(s_i) \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]] \Rightarrow_{M'}^* t_q$. Thus, for every $q' \in Q$ and $i \in [k]$ such that $q'(x_i)$ occurs in ζ_q there is a tree $t_{q',i} \in T_\Delta$ such that $\langle q', Q_{\sigma,i}(F)\rangle(s_i) \Rightarrow_{M'}^* t_{q',i}$ and $t_q = \zeta_q[q'(x_i) \leftarrow t_{q',i} \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]]$. For every $i \in [k]$ with $Q_{\sigma,i}(F) \neq \emptyset$ it follows by induction hypothesis that there is an $\tilde{s}_i$ such that $q'(\tilde{s}_i) \Rightarrow_M^* t_{q',i}$ for every $q' \in Q_{\sigma,i}(F)$. And for every $i \in [k]$ with $Q_{\sigma,i}(F) = \emptyset$ take $\tilde{s}_i = dummy$, where *dummy* is an arbitrary symbol in $\Sigma^{(0)}$. Now take $\tilde{s} = \sigma(\tilde{s}_1, ..., \tilde{s}_k)$. Then

$$\begin{aligned}q(\sigma(\tilde{s}_1, ..., \tilde{s}_k)) &\Rightarrow_M \zeta_q[q'(x_i) \leftarrow q'(\tilde{s}_i) \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]] \\ &\Rightarrow_M^* \zeta_q[q'(x_i) \leftarrow t_{q',i} \mid q' \in Q_{\sigma,i}(\{q\}), i \in [k]] = t_q.\end{aligned}$$

This ends the proof of Claim 2.

If M is linear, then only states of the form $\langle q, \{q\}\rangle$ occur in right-hand sides of the rules in R' that have such states in their left-hand sides. Thus states of the form $\langle q, F\rangle \in Q'$ with $|F| > 1$ are not needed. Leaving out those states and the rules containing them, clearly does not change the output language, because these states never occur in derivations by M'. If we additionally write q instead of $\langle q, \{q\}\rangle$, then M' can also be defined as $(Q', \Sigma, \Delta, q_0, R_1 \cup R_2)$ with

- $Q' = \{q \in Q \mid Inp(\{q\}) \neq \emptyset\}$,
- $R_1 = \{(q(\sigma(x_1, ..., x_k)) \to \zeta) \in R \mid q \in Q',\ \sigma \in \Sigma^{(k)},\ k \geqslant 0,\ \zeta \in RHS(Q', \Delta, k)\} \subseteq R$, and
- $R_2 = \{q(\sigma(x_1, ..., x_k)) \to t_{q,\{q\}} \mid q \in Q',\ \sigma \in \Sigma^{(k)},\ k \geqslant 0$ such that there is no (q, σ)-rule in R or $rhs_M(q, \sigma) \notin RHS(Q', \Delta, k)\}$.

This proves the second statement of Lemma 4.1. ∎

The following example illustrates the construction presented in the proof of Lemma 4.1.

EXAMPLE 4.2. Let us consider a deterministic top-down tree transducer M and let us apply the construction of the proof of Lemma 4.1. Let $M = (Q, \Sigma, \Delta, q_0, R)$ with $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{\gamma^{(1)}, \gamma'^{(1)}, \alpha^{(0)}\}$, $\Delta = \{\sigma^{(2)}, \alpha^{(0)}\}$, and R consisting of the following rules (since Σ is monadic, we write x instead of x_1 throughout this example):

$$
\begin{aligned}
q_0(\gamma(x)) &\to \sigma(q_1(x), q_2(x))\\
q_0(\gamma'(x)) &\to q_1(x)\\
q_0(\alpha) &\to \alpha\\
q_1(\gamma(x)) &\to \sigma(q_0(x), q_2(x))\\
q_2(\gamma'(x)) &\to \sigma(q_0(x), q_1(x))\\
q_1(\alpha) &\to \alpha
\end{aligned}
$$

Let us now construct the total deterministic top-down tree transducer $M' = (Q', \Sigma, \Delta, \langle q_0, \{q_0\}\rangle, R')$ which has the same output language as M, following the construction of the proof of Lemma 4.1. The set of states Q' is $\{\langle q, F\rangle \mid F \subseteq Q, q \in F, Inp(F) \neq \emptyset\}$. It turns out that if $F = \{q_1, q_2\}$ or $F = \{q_0, q_1, q_2\}$, then $Inp(F) = \emptyset$ and otherwise $Inp(F) \neq \emptyset$. We choose $s_{\{q_0\}} = s_{\{q_1\}} = s_{\{q_0, q_1\}} = \alpha$ and $s_{\{q_2\}} = s_{\{q_0, q_2\}} = \gamma'(\alpha)$. The set R' consists of the following rules.

$$
\begin{aligned}
\langle q_0, \{q_0\}\rangle(\gamma(x)) &\to t_{q_0,\{q_0\}} = \alpha\\
&\qquad (\text{because } Q_{\gamma,1}(\{q_0\}) = \{q_1, q_2\} \neq \emptyset\\
&\qquad \text{but } Inp(\{q_1, q_2\}) = \emptyset)\\
\langle q_0, \{q_0\}\rangle(\gamma'(x)) &\to \langle q_1, \{q_1\}\rangle(x)\\
\langle q_0, \{q_0\}\rangle(\alpha) &\to \alpha
\end{aligned}
$$

$$\langle q_1, \{q_1\}\rangle(\gamma(x)) \to \sigma(\langle q_0, \{q_0, q_2\}\rangle(x), \langle q_2, \{q_0, q_2\}\rangle(x))$$

$$\langle q_1, \{q_1\}\rangle(\gamma'(x)) \to t_{q_1, \{q_1\}} = \alpha$$

(because there is no (q_1, γ')-rule in R)

$$\langle q_1, \{q_1\}\rangle(\alpha) \to \alpha$$

$$\langle q_0, \{q_0, q_2\}\rangle(\gamma(x)) \to t_{q_0, \{q_0, q_2\}} = \alpha$$

(because there is no (q_2, γ)-rule in R)

$$\langle q_0, \{q_0, q_2\}\rangle(\gamma'(x)) \to \langle q_1, \{q_0, q_1\}\rangle(x)$$

$$\langle q_0, \{q_0, q_2\}\rangle(\alpha) \to \alpha$$

$$\langle q_2, \{q_0, q_2\}\rangle(\gamma(x)) \to t_{q_2, \{q_0, q_2\}} = \sigma(\alpha, \alpha)$$

(because there is no (q_2, γ)-rule in R)

$$\langle q_2, \{q_0, q_2\}\rangle(\gamma'(x)) \to \sigma(\langle q_0, \{q_0, q_1\}\rangle(x), \langle q_1, \{q_0, q_1\}\rangle(x))$$

$$\langle q_2, \{q_0, q_2\}\rangle(\alpha) \to t_{q_2, \{q_0, q_2\}} = \sigma(\alpha, \alpha)$$

(because there is no (q_2, α)-rule in R)

The right-hand side of every $\langle q_0, \{q_0, q_1\}\rangle$-rule and of every $\langle q_1, \{q_0, q_1\}\rangle$-rule is α. The state $\langle q_2, \{q_2\}\rangle$ is superfluous. It should be clear that $out(M) = out(M') = \{\alpha, \sigma(\alpha, \sigma(\alpha, \alpha))\}$.

From Lemma 4.1 and the fact that $OUT(td\text{-}T) \subseteq OUT(d\text{-}T)$ holds by definition we obtain the following theorem.

THEOREM 4.3. $OUT(d\text{-}T) = OUT(td\text{-}T)$.

For every recognizable tree language L there is a top-down tree automaton M such that $dom(M) = out(M) = L$. By encoding the nondeterministic state behavior of M into the input symbols, i.e., changing the input ranked alphabet to contain one symbol for every (q, σ)-rule, we can "determinize" M to get a deterministic top-down tree transducer M' with $out(M) = out(M')$. Finally we can apply the construction of the proof of Lemma 4.1 to M' to obtain a total deterministic top-down tree transducer M''. Besides changing the input alphabet and adding rules with trees in T_Δ as right-hand side, the rules of M have not changed in M''. That is, they are either of the form $q(\sigma(x_1, ..., x_k)) \to \delta(q_1(x_1), ..., q_k(x_k))$ or of the form $q(\sigma(x_1, ..., x_k)) \to t$ with $t \in T_\Delta$; moreover, for given σ there is at most one rule of the first form. This type of tree transducer will be called *semi-relabeling*, for short *s-relabeling*. If an s-relabeling M is seen as an attribute grammar G, where the states of M are the synthesized attributes of G, then G satisfies the ssur property of Ganzinger and Giegerich [Gan83, Gie88]. This fact will be important in Section 5.

DEFINITION 4.4 (semi-relabeling). Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a total deterministic top-down tree transducer. If for every $\sigma \in \Sigma^{(k)}$ with $k \geqslant 1$, either $rhs(q, \sigma) \in T_\Delta$ for every $q \in Q$, or there is a (unique) $q \in Q$ such that (1) $rhs(q, \sigma)$ has the form $\delta(q_1(x_1), ..., q_k(x_k))$ with $\delta \in \Delta^{(k)}$ and $q_1, ..., q_k \in Q$ and (2) for every $p \in Q$ with $p \neq q : rhs(p, \sigma) \in T_\Delta$, then M is called *semi-relabeling*.

The corresponding class of translations is denoted by s-T.

A deterministic top-down tree transducer $M=(Q, \Sigma, \Delta, q_0, R)$ is called *partial* s-relabeling, if there exists an s-relabeling $M'=(Q, \Sigma, \Delta, q_0, R')$ with $R \subseteq R'$.

Note that all s-relabelings are total deterministic and linear (and even super-linear). Thus $s\text{-}T \subseteq tdl\text{-}T$.

LEMMA 4.5. $RECOG \subseteq OUT(s\text{-}T)$.

Proof. Let $L \in RECOG$. Then there exists a top-down finite tree automaton $M=(Q, \Sigma, \Sigma, q_0, R)$ with $dom(M)=out(M)=L$. Let us now construct a partial s-relabeling with $out(M')=out(M)$. Let $M'=(Q, \Gamma, \Sigma, q_0, R')$ with $\Gamma=\{\sigma^{(k)}_{p, p_1, ..., p_k} \mid (p(\sigma(x_1, ..., x_k)) \to \sigma(p_1(x_1), ..., p_k(x_k))) \in R$ with $p, p_1, ..., p_k \in Q$, $\sigma \in \Sigma^{(k)}$, and $k \geqslant 0\}$ and $R'=\{p(\sigma_{p, p_1, ..., p_k}(x_1, ..., x_k)) \to \sigma(p_1(x_1), ..., p_k(x_k)) \mid \sigma_{p, p_1, ..., p_k} \in \Gamma\}$. It should be clear that $out(M)=out(M')$.

First, we prove that $out(M) \subseteq out(M')$. Consider a derivation $q_0(s) \Rightarrow_{M, u_1, r_1} \xi_1 \Rightarrow_{M, u_2, r_2} \cdots \Rightarrow_{M, u_n, r_n} \xi_n = s$ with $s \in T_\Sigma$, $u_1=\varepsilon$, $u_i \in O(\xi_{i-1})$ for $2 \leqslant i \leqslant n$, and for $i \in [n]: \xi_i \in T_{Q \cup \Sigma}$, $r_i=(p_i(\sigma_i(x_1, ..., x_{k_i})) \to \sigma_i(p_{i,1}(x_1), ..., p_{i,k_i}(x_{k_i}))) \in R$ with $\sigma_i \in \Sigma^{(k_i)}$, $k_i \geqslant 0$. Then, there is a derivation $q_0(s') \Rightarrow^*_{M'} s$, where $s' \in T_\Gamma$ is obtained from s by changing, for every $i \in [n]$, the label σ_i at occurrence u_i in s to the label $(\sigma_i)_{p_i, p_{i,1}, ..., p_{i,ki}}$ and therefore $out(M) \subseteq out(M')$.

Second, we prove that $out(M') \subseteq out(M)$. If there is a derivation $q_0(s) \Rightarrow^*_{M'} t$ by M', with $s \in T_\Gamma$ and $t \in T_\Sigma$, then there is a derivation $q_0(s') \Rightarrow^*_M t$, where $s' \in T_\Sigma$ is obtained from s by changing every label $\sigma_{p, p_1, ..., p_k}$ in s to the label σ. Thus, $out(M') \subseteq out(M)$.

By Lemma 4.1 there is a total deterministic top-down tree transducer M'' with $out(M'')=out(M')$. Since M' is linear it follows by the second sentence of Lemma 4.1 that the set of rules of M'' is equal to $R_1 \cup R_2$ with $R_1 \subseteq R'$ and the right-hand side of every rule in R_2 is in T_Δ. Thus, since M' is a partial s-relabeling, M'' is an s-relabeling. ∎

There is a relationship between Lemma 4.5 and the proof of Theorem 3.2.1 of [ERS80]. Namely, in the latter it is mentioned that every recognizable tree language L is the projection of the domain L_D of a deterministic top-down tree automaton. The partial s-relabeling constructed for L in the proof of Lemma 4.5 has L_D as domain and realizes such a projection (cf. Corollary 3.59(iv) of [Eng75]).

Let us now give an example which involves the constructions presented in Lemmas 4.1 and 4.5.

EXAMPLE 4.6. Let us consider a recognizable tree language L and show that there is an s-relabeling M' with $out(M')=L$. Analogous to the constructions presented in the proof of Lemma 4.1 and Lemma 4.5 we do this in two steps. First, we construct a partial s-relabeling M' with $out(M')=L$ following the construction of the proof of Lemma 4.5, and then we construct an s-relabeling M'' with $out(M'')=out(M')$.

Let $L=\{\sigma(\gamma^i(\alpha), \gamma^j(\beta)) \mid i, j \in \mathbb{N}\} \cup \{\sigma(\gamma^i(\beta), \gamma^j(\alpha)) \mid i, j \in \mathbb{N}\}$. It is well known that the class of tree languages recognized by deterministic top-down finite tree automata is a proper subclass of $RECOG$. Indeed, the language L is a recognizable

tree language which is *not* in this class (see, e.g., Section II.11 of [GS84]), However, the ability to relabel input symbols allows us to construct a (deterministic!) partial s-relabeling M' with $out(M') = L$.

Let $M = (Q, \Sigma, \Sigma, q_0, R)$ be the top-down finite tree automaton with $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$, and R consists of the following rules:

$$
\begin{aligned}
q_0(\sigma(x_1, x_2)) &\to \sigma(q_1(x_1), q_2(x_2)) \\
q_0(\sigma(x_1, x_2)) &\to \sigma(q_2(x_1), q_1(x_1)) \\
q_i(\gamma(x_1)) &\to \gamma(q_i(x_1)), \qquad \text{for} \quad i \in [2] \\
q_1(\alpha) &\to \alpha \\
q_2(\beta) &\to \beta
\end{aligned}
$$

Then $dom(M) = out(M) = L$. Following the proof of Lemma 4.5 we can construct the partial s-relabeling $M' = (Q, \Sigma', \Sigma, q_0, R')$ with $\Sigma' = \{\sigma^{(2)}_{q_0, q_1, q_2}, \sigma^{(2)}_{q_0, q_2, q_1}, \gamma^{(1)}_{q_1, q_1}, \gamma^{(1)}_{q_2, q_2}, \alpha^{(0)}_{q_1}, \beta^{(0)}_{q_2}\}$ and R' consists of the following rules:

$$
\begin{aligned}
q_0(\sigma_{q_0, q_1, q_2}(x_1, x_2)) &\to \sigma(q_1(x_1), q_2(x_2)) \\
q_0(\sigma_{q_0, q_2, q_1}(x_1, x_2)) &\to \sigma(q_2(x_1), q_1(x_1)) \\
q_i(\gamma_{q_i, q_i}(x_1)) &\to \gamma(q_i(x_1)), \qquad \text{for} \quad i \in [2] \\
q_1(\alpha_{q_1}) &\to \alpha \\
q_2(\beta_{q_2}) &\to \beta
\end{aligned}
$$

Obviously M' is deterministic and $out(M') = L$. Let us now apply the construction of the proof of Lemma 4.1. Since M' is linear and $Inp(\{q\}) \neq \varnothing$ for every $q \in Q'$, only the set of rules is changed in M''. Let $M'' = (Q, \Sigma', \Sigma, q_0, R_1 \cup R_2)$, where $R_1 = R'$ and $R_2 = \{q(\sigma(x_1, ..., x_k)) \to t_{q, \{q\}} \mid q \in Q, \sigma \in \Sigma'^{(k)}, k \geqslant 0$ such that there is no (q, σ)-rule in $R'\}$. Let $s_{\{q_0\}} = \sigma_{q_0, q_1, q_2}(\alpha_{q_1}, \beta_{q_2})$, $s_{\{q_1\}} = \alpha_{q_1}$, and $s_{\{q_2\}} = \beta_{q_2}$. The output trees which we will need are $t_{q_0, \{q_0\}} = \sigma(\alpha, \beta)$, $t_{q_1, \{q_1\}} = \alpha$, and $t_{q_2, \{q_2\}} = \beta$. The rules in R_2 are

$$
\begin{aligned}
q_0(\gamma_{q_i, q_i}(x_1)) &\to t_{q_0, \{q_0\}} = \sigma(\alpha, \beta), \qquad \text{for} \quad i \in [2] \\
q_0(\delta) &\to t_{q_0, \{q_0\}} = \sigma(\alpha, \beta), \qquad \text{for} \quad \delta \in \{\alpha, \beta\} \\
q_1(\delta(x_1, x_2)) &\to t_{q_1\{q_1\}} = \alpha, \qquad \text{for} \quad \delta \in \{\sigma_{q_0, q_1, q_2}, \sigma_{q_0, q_2, q_1}\} \\
q_1(\gamma_{q_2, q_2}(x_1)) &\to t_{q_1, \{q_1\}} = \alpha \\
q_2(\delta(x_1, x_2)) &\to t_{q_2\{q_2\}} = \beta, \qquad \text{for} \quad \delta \in \{\sigma_{q_0, q_1, q_2}, \sigma_{q_0, q_2, q_1}\} \\
q_2(\gamma_{q_1, q_1}(x_1)) &\to t_{q_2, \{q_2\}} = \beta
\end{aligned}
$$

Obviously, M'' is an s-relabeling which generates the same output language as M'.

We are now ready to prove the main result of this section.

THEOREM 4.7. $RECOG = OUT(s\text{-}T) = OUT(tdl\text{-}T)$.

Proof. By Lemma 4.5, $RECOG \subseteq OUT(s\text{-}T)$. Since s-relabelings are linear it follows that $OUT(s\text{-}T) \subseteq OUT(tdl\text{-}T)$. The inclusion $OUT(tdl\text{-}T) \subseteq RECOG$ follows from the closure of $RECOG$ under linear top-down tree transductions (cf. Corollary IV.6.6 of [GS84]). ∎

5. TERM-GENERATING POWER OF ATTRIBUTE GRAMMARS AND ATTRIBUTED TREE TRANSDUCERS

In this section we want to show that attribute grammars and attributed tree transducers have the same term-generating power, i.e., $OUT(AG, TERMS) = OUT(AT)$. We will show the composition result $s\text{-}T \circ AT \subseteq AT$. This result is not surprising, because the restriction of being an s-relabeling is quite strong. In particular, every such top-down tree transducer respects the "syntactic single use restriction" (ssur) of [Gan83] (defined for attribute coupled grammars). This restriction requires that for an input symbol σ, every outside attribute may be used at most once in the set R_σ of rules for σ. A top-down tree transducer can be seen as an attributed tree transducer without inherited attributes (where its states are the synthesized attributes). In the context of top-down tree transducers the ssur property means that for an input symbol σ, a tree $q(x_i)$ may only occur *once* in the set of σ-rules for a particular q and i. This is obviously true for s-relabelings, because for every input symbol σ there is at most *one* rule of the form $q(\sigma(x_1, ..., x_k)) \to \delta(q_1(x_1), ..., q_k(x_k))$ and all other σ-rules have right-hand sides in T_Δ.

In [Gan83] it is proved that the class of translations realized by ssur attribute coupled grammars (for short, ssur-AC) are closed under composition. Observation 3 of [Gie88] states that the composition of the class ssur-AC with the class AC of translations realized by attribute coupled grammars is included in AC, i.e., ssur-AC $\circ$ AC $\subseteq$ AC. Thus, the composition result for s-relabelings and attributed tree transducers follows in a more or less straightforward manner from the result by Ganzinger. However, since the class $s\text{-}T$ is rather small in comparison with ssur-AC, we can present a shorter construction for the proof of this composition.

Let us explain the construction in more detail. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an s-relabeling and let A be an attributed tree transducer with set Syn of synthesized attributes. The idea of the construction is that the attributed tree transducer A is used to translate the right-hand sides of the rules of the s-relabeling M. This yields the rules for the attributed tree transducer A', which computes the composition of M and A. Unlike usual product constructions, we only have to change the set of synthesized attributes of A' to be the set $Q \times Syn$. The set of inherited attributes can remain the same for A' as for A, because for every symbol $\sigma \in \Sigma$, there is only *one* rule of the form $q(\sigma(x_1, ..., x_k)) \to \delta(q_1(x_1), ..., q_k(x_k))$ and therefore, for every inside inherited attribute instance $\beta(\pi i)$ there can only be one particular state q_i when translating a right-hand side of M for the symbol σ.

Instead of showing that $s\text{-}T \circ AT \subseteq AT$, we show the following slightly more general result in which we only assume that the attributed tree transducer is noncircular on the output language of the s-relabeling.

LEMMA 5.1. *Let M be an s-relabeling and let A be an attributed tree transducer. If A is noncircular on $out(M)$, then there is an attributed tree transducer A' such that $\tau_{A'} = \tau_M \circ \tau_A$.*

Proof. Let $M = (Q, \Sigma, \Gamma, q_0, P)$ be an s-relabeling and let $A = (Syn, Inh, \Gamma, \Delta, root, a_0, R)$ be an attributed tree transducer which is noncircular on $out(M)$. We may assume that M is reduced (cf. Observation 3.7). Let $A' = (Syn', Inh, \Sigma, \Delta, root, (q_0, a_0), R')$ be the attributed tree transducer with $Syn' = Q \times Syn$ and R' consisting of the following rules.

Let $\sigma \in \Sigma^{(k)}$ with $k \geqslant 0$.

If $rhs_M(q, \sigma) \in T_\Gamma$ for every $q \in Q$, then for every $b \in Inh$ and $i \in [k]$ let the rule $b(\pi i) \to dummy$ be in R'_σ, where *dummy* is an arbitrary symbol in $\Delta^{(0)}$.

Let $q \in Q$.

(1) If $rhs_M(q, \sigma) = \delta(q_1(x_1), ..., q_k(x_k))$ for $\delta \in \Gamma^{(k)}$, $q_1, ..., q_k \in Q$, and $k \geqslant 1$, then let the substitution $[\alpha(\pi i) \leftarrow (q_i, \alpha)(\pi i) \mid \alpha \in Syn, i \in [k]]$ be denoted by Θ, and for every synthesized attribute $a \in Syn$ let the rule $(q, a)(\pi) \to rhs_A(a, \pi, \delta)\Theta$ be in R'_σ, and for every inherited attribute $b \in Inh$ and every $i \in [k]$ let the rule $b(\pi i) \to rhs_A(b, \pi i, \delta)\Theta$ be in R'_σ.

(2) If $rhs_M(q, \sigma) = \zeta \in T_\Gamma$, then let the substitution $[\beta(\varepsilon) \leftarrow \beta(\pi) \mid \beta \in Inh]$ be denoted by Φ, and for every synthesized attribute $a \in Syn$ let the rule $(q, a)(\pi) \to nf(\Rightarrow_{A,\zeta}, a(\varepsilon))\Phi$ be in R'_σ. Since M is reduced, there exists a tree $t \in out(M)$ such that ζ is a subtree of t. Therefore $nf(\Rightarrow_{A,\zeta}, a(\varepsilon))$ exists, because A is noncircular on $out(M)$ and hence also on ζ.

The set R'_{root} is defined as follows. Let the substitution $[\alpha(\pi 1) \leftarrow (q_0, \alpha)(\pi 1) \mid \alpha \in Syn]$ be denoted by Π. Let the rule $(q_0, a_0)(\pi) \to rhs_A(a_0, \pi, root)\Pi$ be in R'_{root} and for every $b \in Inh$ let the rule $b(\pi 1) \to rhs_A(b, \pi 1, root)\Pi$ be in R'_{root}.

Let us now prove the correctness of the construction; i.e., let us show that for every $s \in T_\Sigma$, $nf(\Rightarrow_{A', root(s)}, (q_0, a_0)(\varepsilon)) = nf(\Rightarrow_{A, root(t)}, a_0(\varepsilon))$, where $t = nf(\Rightarrow_M, q_0(s))$. Let us first prove a similar statement for the "non-*root*" case.

Claim 1. Let $s \in T_\Sigma$. Then A' is noncircular on s and for all $q \in Q$ and $a \in Syn$, $nf(\Rightarrow_{A', s}, (q, a)(\varepsilon)) = nf(\Rightarrow_{A, t}, a(\varepsilon))$, where $t = nf(\Rightarrow_M, q(s))$.

Note again that $nf(\Rightarrow_{A,t}, a(\varepsilon))$ exists, because t is, by reducedness of M, a subtree of a tree in $out(M)$, and A is noncircular on $out(M)$. We prove Claim 1 by induction on the structure of s. Let $s = \sigma(s_1, ..., s_k)$ with $\sigma \in \Sigma^{(k)}$, $s_1, ..., s_k \in T_\Sigma$, and $k \geqslant 0$. Let the induction hypothesis, i.e., that Claim 1 holds for $s_1, ..., s_k$, be denoted by IH1.

We first prove that A' is noncircular on s. If $rhs_M(q, \sigma) \in T_\Gamma$ for every $q \in Q$, then for every $b \in Inh$ and $i \in [k]$ the $(b, \pi i, \sigma)$-rule has a dummy symbol as right-hand side. Hence the rules in R'_σ cannot introduce a circularity. Since by IH1 A' is noncircular on s_i for $i \in [k]$, it follows that A' is noncircular on s.

If there is a $q \in Q$ such that $rhs_M(q, \sigma) = \delta(q_1(x_1), ..., q_k(x_k))$ for $\delta \in \Gamma^{(k)}$, $q_1, ..., q_k \in Q$, and $k \geqslant 1$, then assume that A' is circular on s. Then there exist $(q', a') \in Syn'$, $i \in [k]$, and $\xi \in T_{\Delta \cup Syn' \cup Inh}(O(s))$ such that

$$(q', a')(i) \Rightarrow^{+}_{A', s} \xi \qquad \text{and} \qquad (q', a')(i) \text{ occurs in } \xi.$$

More precisely, there are $m \geqslant 2$, $a_1, ..., a_m \in Syn$, $b_1, ..., b_{m-1} \in Inh$, $\xi_1, ..., \xi_{m-1} \in T_{\Delta \cup Inh}(\{\varepsilon\})$, $\xi'_1, ..., \xi'_{m-1} \in RHS(Syn', Inh, \Delta, k)$, and $i_1, ..., i_m \in [k]$ such that $(q', a')(i) = (q_{i_1}, a_1)(i_1) = (q_{i_m}, a_m)(i_m)$ and for every $j \in [m-1]$, $\xi_j = nf(\Rightarrow_{A', s_{ij}}, (q_{i_j}, a_j)(\varepsilon))$, $\xi'_j = rhs_{A'}(b_j, \pi i_j, \sigma)$,

- $b_j(\varepsilon)$ occurs in ξ_j, and
- $(q_{i_{j+1}}, a_{j+1})(\pi i_{j+1})$ occurs in ξ'_j.

For $i \in [k]$ let $t_i = nf(\Rightarrow_M, q_i(s_i))$. Then, for every $j \in [m-1]$,

- by IH1, $\xi_j = nf(\Rightarrow_{A, t_{ij}}, a_j(\varepsilon))$. Thus $b_j(\varepsilon)$ occurs in $nf(\Rightarrow_{A, t_{ij}}, a_j(\varepsilon))$. And
- by the definition of the rules of A', $\xi'_j = rhs_A(b_j, \pi i_j, \delta)\Theta$. But Θ replaces $a_{j+1}(\pi i_{j+1})$ by $(q_{i_{j+1}}, a_{j+1})(\pi i_{j+1})$ and thus $a_{j+1}(\pi i_{j+1})$ occurs in $rhs_A(b_j, \pi i_j, \delta)$.

Hence, there is a $\bar{\xi} \in T_{\Delta \cup Syn \cup Inh}(O(t))$ such that $a'(i) \Rightarrow^{+}_{A, t} \bar{\xi}$, with $t = nf(\Rightarrow_M, q(s)) = \delta(t_1, ..., t_k)$, and $a'(i)$ occurs in $\bar{\xi}$. This means that A is circular on t. By reducedness of M, t is a subtree of a tree in $out(M)$. This contradicts the fact that A is noncircular on $out(M)$ and therefore A' is noncircular on s.

Let us now prove that for all $q \in Q$ and $a \in Syn$, $nf(\Rightarrow_{A', s}, (q, a)(\varepsilon)) = nf(\Rightarrow_{A, t}, a(\varepsilon))$, where $t = nf(\Rightarrow_M, q(s))$. Let $q \in Q$.

Case 1. $rhs_M(q, \sigma) = \zeta \in T_\Gamma$. In this special case it follows clearly from Observation 3.12 that $nf(\Rightarrow_{A', s}, (q, a)(\varepsilon))$ is equal to $rhs_{A'}((q, a), \pi, \sigma)[\beta(\pi) \leftarrow \beta(\varepsilon) \mid \beta \in Inh]$. By the definition of the rules of A' this equals $nf(\Rightarrow_{A, \zeta}, a(\varepsilon))\, \Phi[\beta(\pi) \leftarrow \beta(\varepsilon) \mid \beta \in Inh]$ which is equal to $nf(\Rightarrow_{A, t}, a(\varepsilon))$ because $\Phi[\beta(\varepsilon) \leftarrow \beta(\pi) \mid \beta \in Inh] = id$ and $nf(\Rightarrow_M, q(s)) = \zeta$.

Case 2. $rhs_M(q, \sigma) = \delta(q_1(x_1), ..., q_k(x_k))$ for $\delta \in \Gamma^{(k)}$, $q_1, ..., q_k \in Q$, and $k \geqslant 1$.

Consider the *extended is-graph of* s, i.e., the graph (V, E) with $V = \{c(\pi i) \mid c \in Syn' \cup Inh, 0 \leqslant i \leqslant k\}$ and $E = \{(c(\pi i), c'(\pi i')) \in outs_\sigma \times ins_\sigma \mid c(\pi i)$ occurs in $rhs_{A'}(c', \pi i', \sigma)\} \cup \{(c(\pi i), c'(\pi i)) \in ins_\sigma \times outs_\sigma \mid c(\varepsilon)$ occurs in $nf(\Rightarrow_{A', s_i}, c'(\varepsilon))\}$. The nodes in this graph which have no incoming edges represent attribute instances which do not depend on any attribute instance in s. Let V_0 be the set of such nodes, i.e., $V_0 = \{v \in V \mid$ there is no $v' \in V$ such that $(v', v) \in E\}$ and for $n \geqslant 1$ let $V_n = \{v \in V \mid$ for all $v' \in V$, if $(v', v) \in E$ then $v' \in V_j$ for some $j < n\}$. To prove Claim 1 for Case 2 we need a stronger claim which holds for all attribute instances in V. Since A' is noncircular on s, the extended is-graph of s is noncircular and therefore every node in V belongs to some V_n.

Claim 2. Let $n \geqslant 0$, $v \in V_n$, and $t = nf(\Rightarrow_M, q(s))$. If $v = (q, a)(\pi)$ with $a \in Syn$, then $nf(\Rightarrow_{A', s}, (q, a)(\varepsilon)) = nf(\Rightarrow_{A, t}, a(\varepsilon))$. If $v = b(\pi j)$ with $b \in Inh$ and $j \in [k]$, then $nf(\Rightarrow_{A', s}, b(j)) = nf(\Rightarrow_{A, t}, b(j))$.

Let us prove Claim 2 by natural induction on n. We denote the induction hypothesis, i.e., that Claim 2 holds for every natural number smaller than n, by IH2. Then, the induction base is void and we merely have to prove the induction step, as follows.

We first consider the case that $v = (q, a)(\pi)$ with $a \in Syn$. Then $nf(\Rightarrow_{A', \sigma(s_1, ..., s_k)}, (q, a)(\varepsilon))$ is by Observation 3.12 equal to $rhs_{A'}((q, a), \pi, \sigma)[\ldots]$, where $[\ldots]$ denotes the following substitution

$$[(q', \alpha)(\pi i) \leftarrow nf(\Rightarrow_{A', s_i}, (q', \alpha)(\varepsilon))\ \Psi_i \mid (q', \alpha) \in Syn', i \in [k]]\ \Phi^{-1},$$

Φ is the substitution $[\beta(\varepsilon) \leftarrow \beta(\pi) \mid \beta \in Inh]$ (as in the definition of the rules of A'), and for $(q', \alpha) \in Syn'$ and $i \in [k]$, Ψ_i denotes the substitution $[\beta(\varepsilon) \leftarrow nf(\Rightarrow_{A', s}, \beta(i)) \mid \beta \in Inh]$. By the definition of the rules of A', $rhs_{A'}((q, a), \pi, \sigma)[\ldots]$ is equal to

$$rhs_A(a, \pi, \delta)[\alpha(\pi i) \leftarrow (q_i, \alpha)(\pi i) \mid \alpha \in Syn, i \in [k]][\ldots].$$

By IH1 we can replace $nf(\Rightarrow_{A', s_i}, (q_i, \alpha)(\varepsilon))$ by $nf(\Rightarrow_{A, t_i}, \alpha(\varepsilon))$, where $t_i = nf(\Rightarrow_M, q_i(s_i))$. For every $nf(\Rightarrow_{A', s}, \beta(i))$ that is actually used in the above substitution, there are edges in E from $\beta(i)$ to $(q_i, \alpha)(\pi i)$ and from $(q_i, \alpha)(\pi i)$ to $(q, a)(\pi) = v$, and so $\beta(i) \in V_j$ for some $j < n$. Hence, by IH2 we can replace $nf(\Rightarrow_{A', s}, \beta(i))$ by $nf(\Rightarrow_{A, t}, \beta(i))$, where $t = nf(\Rightarrow_M, q(s))$. Thus, the above tree becomes

$$rhs_A(a, \pi, \delta)[\alpha(\pi i) \leftarrow nf(\Rightarrow_{A, t_i}, \alpha(\varepsilon))[\beta(\varepsilon) \leftarrow nf(\Rightarrow_{A, t}, \beta(i)) \mid \beta \in Inh] \mid \alpha \in Syn, i \in [k]]\ \Phi^{-1}.$$

Since $t_i = nf(\Rightarrow_M, q_i(s_i))$ for $i \in [k]$ and $rhs_M(q, \sigma) = \delta(q_1(x_1), ..., q_k(x_k))$, it follows that $t = nf(\Rightarrow_M, q(s)) = \delta(t_1, ..., t_k)$. Then, by Observation 3.12 the above tree is equal to $nf(\Rightarrow_{A, t}, \alpha(\varepsilon))$.

Analogous to the above proof for $v = (q, a)(\pi)$, Claim 2 can be proved for $v = b(\pi j)$ where $b \in Inh$ and $j \in [k]$. Again this proof is based on Observation 3.12.

This finishes the proof of Claim 2 and Claim 1.

The *root* case can be proved by proving Claims 3 and 4, which are entirely similar to Claims 1 and 2 (with a (virtual) rule $q_0(root(x_1)) \to root(q_0(x_1))$ instead of $q(\sigma(x_1, ..., x_k)) \to \delta(q_1(x_1), ..., q_k(x_k))$). We omit this and merely present these claims.

Claim 3. Let $s \in T_\Sigma$. Then A' is noncircular on $root(s)$ and $nf(\Rightarrow_{A', root(s)}, (q_0, a_0)(\varepsilon)) = nf(\Rightarrow_{A, root(t)}, a_0(\varepsilon))$, where $t = nf(\Rightarrow_M, q_0(s))$.

Claim 4 is on the *extended is-graph of* $root(s)$, i.e., the graph (V, E) with $V = \{(q_0, a_0)(\pi)\} \cup \{c(\pi 1) \mid c \in Syn' \cup Inh\}$ and $E = \{(a(\pi 1), c(\pi i)) \in outs_{root} \times ins_{root} \mid a(\pi 1)$ occurs in $rhs_{A'}(c, \pi i, root)\} \cup \{(b(\pi 1), a(\pi 1)) \in ins_{root} \times outs_{root} \mid b(\varepsilon)$ occurs in $nf(\Rightarrow_{A', s}, a(\varepsilon))\}$. For $n \geqslant 0$ the sets V_n are defined as for the extended is-graph of s. Since A' is noncircular on $root(s)$, the extended is-graph of $root(s)$ is noncircular.

Claim 4. Let $n \geqslant 0$, $v \in V_n$, and $t = nf(\Rightarrow_M, q_0(s))$. If $v = (q_0, a_0)(\pi)$ then $nf(\Rightarrow_{A', root(s)}, (q_0, a_0)(\varepsilon)) = nf(\Rightarrow_{A, root(t)}, a_0(\varepsilon))$. If $v = b(\pi 1)$ with $b \in Inh$, then $nf(\Rightarrow_{A', root(s)}, b(1)) = nf(\Rightarrow_{A, root(t)}, b(1))$.

The proofs of Claim 3 and 4 are again based on Observation 3.12. ▮

Now we have all the tools to prove that the result $T(RECOG) = OUT(T)$ of Theorem 3.2.1 of [ERS80] for top-down tree transducers can be extended to attributed tree transducers. That is, we can get rid of the recognizable tree language as input for an attributed tree transducer A and simply consider the set T_Σ as input (where Σ is the input alphabet of A) while still being able to obtain the same output language.

LEMMA 5.2. $AT(RECOG) = OUT(AT)$.

Proof. Let L be a recognizable tree language and let A be an attributed tree transducer which is noncircular on L. By Theorem 4.7 there is an s-relabeling M with $out(M) = L$. Then, $\tau_A(L) = \tau_A(out(M)) = ran(\tau_M \circ \tau_A)$. By Lemma 5.1 there is an attributed tree transducer A' with $\tau_{A'} = \tau_M \circ \tau_A$. Thus, $AT(RECOG) \subseteq OUT(AT)$. The other direction of this inclusion is trivially true; an attributed tree transducer A with input ranked alphabet Σ takes trees in T_Σ as input, thus $out(A) = \tau_A(T_\Sigma)$. Moreover, T_Σ is a recognizable tree language and therefore $OUT(AT) \subseteq AT(RECOG)$. ▮

To prove that the term-generating power of attribute grammars and attributed tree transducers is equal, we need another straightforward result stating that attribute grammars can generate the same output languages as attributed tree transducers. This is true, because the set $\{root(s) \mid s \in T_\Sigma\}$ of control trees of an attributed tree transducer A with input alphabet Σ is a local tree language; i.e., there is a context-free grammar G_0, such that the set of derivation trees of G_0 is equal to $\{root(s) \mid s \in T_\Sigma\}$.

LEMMA 5.3. $OUT(AT) \subseteq OUT(AG, TERMS)$.

Proof. Let $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be an attributed tree transducer. Define an attribute grammar $G = (G_0, D, B, R')$ as follows:

- $G_0 = (N, T, S, P)$ with $N = \Sigma \cup \{root\}$, $T = \varnothing$, $S = root$, $P = \bigcup_{X \in N} P_X$, where if $k = rank_{\Sigma'}(X)$, then $P_X = \{X \to X_1 \cdots X_k \mid X_1, ..., X_k \in \Sigma\}$, where $\Sigma' = \Sigma \cup \{root^{(1)}\}$. Note that if $rank_{\Sigma'}(X) = 0$, then $P_X = \{X \to \varepsilon\}$.
- D is the free Δ-algebra; i.e., $D = (T_\Delta, \Delta)$.
- $B = (Syn, Inh, att, a_0)$ with $att(X) = Syn \cup Inh$ for all $X \in \Sigma$, and $att(root) = \{a_0\}$.
- $R' = (R'(p) \mid p \in P)$, where for all $X \in N$ and $p \in P_X$: $R'(p) = \{l\Theta = r\Theta \mid (l \to r) \in R_X\}$, where Θ is the substitution

$$[a(\pi i) \leftarrow \langle a, i \rangle \mid 0 \leqslant i \leqslant rank_{\Sigma'}(X) \text{ and } a \in Att],$$

and Σ' is as above.

The set of derivation trees of G_0 is the set $\{root(s) \mid s \in T_\Sigma\}$. It should be clear that the attributed tree transducer A and the attribute grammar G realize the same translation, if derivation trees are seen as trees over a ranked alphabet. Therefore, $out(A) = out(G)$. ▮

Let us give a short example for this simple construction.

EXAMPLE 5.4. Let $A_{bin} = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be the attributed tree transducer as defined in Example 3.13. Let us now define an attribute grammar $G_{bin} = (G_0, D, B, R')$ which generates the same output as A_{bin}, where

- $G_0 = (N, T, S, P)$ with $N = \Sigma \cup \{root\} = \{root, 1, 0, e\}$, $T = \varnothing$, $S = root$, $P = P_{root} \cup P_1 \cup P_0 \cup P_e$ with $P_{root} = \{root \to 1, root \to 0, root \to e\}$, $P_1 = \{1 \to 1, 1 \to 0, 1 \to e\}$, $P_0 = \{0 \to 1, 0 \to 0, 0 \to e\}$, $P_e = \{e \to \varepsilon\}$.
- $D = (T_\Delta, \Delta)$
- $B = (Syn, Inh, att, a)$ with $Syn = \{a\}$, $Inh = \{b\}$, $att(1) = att(0) = att(e) = Syn \cup Inh = \{a, b\}$ and $att(root) = \{a\}$.
- $R' = (R'(p) \mid p \in P)$, where

for $p \in P_{root}$: $R'(p) = \{\langle a, 0\rangle = \langle a, 1\rangle, \langle b, 1\rangle = 0\}$,

for $p \in P_1$: $R'(p) = \{\langle a, 0\rangle = exp(\langle b, 0\rangle) + \langle a, 1\rangle, \langle b, 1\rangle = s(\langle b, 0\rangle)\}$,

for $p \in P_0$: $R'(p) = \{\langle a, 0\rangle = \langle a, 1\rangle, \langle b, 1\rangle = s(\langle b, 0\rangle)\}$,

and for $p \in P_e$: $R'(p) = \{\langle a, 0\rangle = 0\}$.

Now, if we consider the binary number 101 again, then there is a derivation by G_0: $root \Rightarrow 1 \Rightarrow 0 \Rightarrow 1 \Rightarrow e$ and the corresponding derivation tree is $root(1(0(1(e))))$. This is exactly the tree shown in Fig. 1. The output of G for this input evaluates of course to $exp(0) + (exp(s(s(0))) + 0)$ as for the attributed tree transducer A_{bin}.

Since the set of rule trees of a context-free grammar is recognizable (see, e.g., Section 3.2 of [Eng75]), every attribute grammar can be simulated by an attributed tree transducer which takes a recognizable tree language as input.

LEMMA 5.5. $OUT(AG, TERMS) \subseteq AT(RECOG)$.

Proof. Let $G = (G_0, D, B, R)$ be an attribute grammar where $G_0 = (N, T, S, P)$, $D = (T_\Delta, \Delta)$ for some ranked alphabet Δ, $B = (Syn, Inh, att, a_0)$, and $R = (R(p) \mid p \in P)$. Let $\Sigma = \{\bar{p} \mid p \in P\}$ and $rank_\Sigma(\bar{p}) = k$ if p is of the form $X_0 \to w_0 X_1 w_1 X_2 w_2 \cdots X_k w_k$ with $k \geqslant 0$, $X_0, ..., X_k \in N$, and $w_0, ..., w_k \in T^*$. Let $M = (Q, \Sigma, \Sigma, \bar{S}, R)$ be the top-down finite tree automaton with $Q = \{\bar{X} \mid X \in N\}$ and R defined as follows. For $p \in P$ of the form $X_0 \to w_0 X_1 w_1 X_2 w_2 \cdots X_k w_k$ let the rule

$$\bar{X}_0(\bar{p}(x_1, ..., x_n)) \to \bar{p}(\bar{X}_1(x_1), ..., \bar{X}_k(x_k))$$

be in R. Obviously, $out(M)$ is exactly the set of rule trees of G_0 (c.f., e.g., Section 3.2 of [Eng75]). Let $A = (Syn, Inh, \Sigma, \Delta, a_0, root, R')$ be the attributed tree transducer defined as follows. Let $\bar{p} \in \Sigma^{(k)}$ and let the substitution $[\langle d, i\rangle \leftarrow d(\pi i) \mid d \in Syn \cup Inh, 0 \leqslant i \leqslant k, \langle d, i\rangle \in outs(p)]$ be denoted by Θ. Let *dummy* be an arbitrary symbol

in $\Delta^{(0)}$. For every $c(\pi i) \in ins_{\bar{p}}$ with $c \in Att$, and $0 \leqslant i \leqslant k$ let the rule $c(\pi i) \to \zeta$ be in $R'_{\bar{p}}$, where

- $\zeta = dummy$, if $\langle c, i \rangle \notin ins(p)$, and
- $\zeta = t\Theta$ if $\langle c, i \rangle = t$ is a rule in $R(p)$.

The set R'_{root} is defined as $\{a_0(\pi) \to a_0(\pi 1)\} \cup \{b(\pi 1) \to dummy \mid b \in Inh\}$, where *dummy* is as above.

Since G is noncircular it should be clear that A is noncircular on $out(M)$ and that $out(G) = \tau_A(out(M))$. Since $out(M) \in RECOG$ it follows that $out(G) \in AT(RECOG)$. ▮

Finally, we can prove our main theorem; i.e., attribute grammars and attributed tree transducers have the same term-generation power.

THEOREM 5.6. $OUT(AG, TERMS) = AT(RECOG) = OUT(AT)$.

Proof. By Lemma 5.5, $OUT(AG, TERMS)$ is included in $AT(RECOG)$ which is by Lemma 5.2 equal to $OUT(AT)$ and by Lemma 5.3 included in $OUT(AG, TERMS)$. ▮

6. A HIERARCHY FOR THE GENERATING POWER OF ATTRIBUTED TREE TRANSDUCERS

In this section we want to show that there is a hierarchy for the classes of output languages of attributed tree transducers with respect to the number of attributes. In fact, the classes of output languages of ssur attributed tree transducers also form a hierarchy with respect to the number of attributes which will be shown as well.

To prove these hierarchies we make use of the equivalence of the term-generating power of attribute grammars and that of context-free hypergraph grammars, which was proved in [EH92], and apply the pumping lemma for context-free hypergraph languages by Habel and Kreowski. We assume the reader to be familiar with [EH92] and with the pumping lemma of [HK87, Hab92]. Let us now very briefly define the notion of context-free hypergraph grammars.

A *context-free hypergraph grammar* (for short, cfhg) G is a tuple (Σ, Δ, S, P), where Σ is a ranked alphabet, $\Delta \subseteq \Sigma$ is the terminal alphabet, $\Sigma - \Delta$ is the nonterminal alphabet, and P is a finite set of productions of the form $X \to H$, where $X \in \Sigma - \Delta$, H is a hypergraph over Σ (see, e.g., [EH92] for a definition of hypergraphs) and $rank_\Sigma(X) = rank(H)$, where $rank(H)$ denotes the number of external nodes of H. $S \in \Sigma - \Delta$ is the initial nonterminal. For $k \in \mathbb{N}$, G *is of rank* k, if k is equal to the maximum of the ranks of the nonterminals of G.

Let us only fix a few notations concerning cfhg's, for more details cf. [EH92]. Let G be a cfhg. In general, G generates a set $L(G)$ of hypergraphs. If every hypergraph in $L(G)$ is a jungle, then G is called *term-generating*. By unfolding jungles in the usual way one obtains trees. Note that by a jungle we always mean a *clean* jungle, i.e., there are no nodes or hyperedges that are not needed when unfolding it (it does not contain "garbage"); in [EH92] this is called "clean term-generating". The *tree-language generated by* G is denoted by $TREE(G)$. Let $k \in \mathbb{N}$. The class of all

tree languages generated by (term-generating) cfhg's of rank less than or equal to k is denoted by $\mathcal{TREE}_k$. The result of [EH92] is that $OUT(AG, TERMS)$ is the union of all $\mathcal{TREE}_k$.

Let us now give a first example of an application of the pumping lemma for context-free hypergraph languages. In Theorem 4.3 of [KV94] it is shown that the tree language $\{\gamma^{2^n}\alpha \mid n \geqslant 0\}$ with γ of rank 1 and α of rank 0, is not in the class of output languages of so called visiting and producing attributed tree transducers. This is proved by using their pumping lemma which is defined for the class of output languages of such attributed tree transducers. An attributed tree transducer is producing, if every application of a rule produces at least one new output symbol. An attributed tree transducer is visiting, if for every input tree s and for every node x of s, the value of at least one attribute instance of x is needed to compute the value of the initial attribute instance at the root. From a practical point of view these restrictions seem to be rather strong, because you may not even use any copy-rules for attributes. For instance, the attributed tree transducer A_{bin} of Example 3.13 is not producing, because it contains rules that do not produce any output symbols (i.e., copy-rules). It is not clear how the class of output languages of visiting and producing attributed tree transducers is related to the class $OUT(AT)$ of output languages of (unrestricted) attributed tree transducers. In [KV94] no statement is made on how these classes are related to each other. With the help of the pumping lemma for context-free hypergraph languages we can show that the language $\{\gamma^{2^n}\alpha \mid n \geqslant 0\}$ is not in the class of output languages of attributed tree transducers.

EXAMPLE 6.1. We want to show that $L_{exp} = \{\gamma^{2^n}\alpha \mid n \geqslant 0\} \notin OUT(AT)$. Assume that L_{exp} is in $OUT(AT)$, i.e., there is an attributed tree transducer A with $out(A) = L_{exp}$. Then, by Lemma 5.3 there is an attribute grammar G with output language L_{exp}. By Engelfriet and Heyker's result (Lemma 4.1 of [EH92]) there is a term-generating cfhg G' such that $TREE(G') = L_{exp}$. In fact, $L(G')$ contains (clean) jungles without sharing, i.e., trees, because there cannot be sharing in monadic trees; i.e., $L(G') = \{tree(\gamma^{2^n}\alpha) \mid n \geqslant 0\}$, where $tree(\gamma^{2^n}\alpha)$ is the hypergraph that represents the tree $\gamma^{2^n}\alpha$ (and thus has $2^n + 1$ edges). Now from the fact that the growth of the number of edges is linear for context-free hypergraph languages (Corollary 4.5 of [HK87], which is an application of their pumping lemma), it follows that such a grammar G' cannot exist and therefore the attributed tree transducer A cannot exist either.

In fact, it is already shown in [EF81] that L_{exp} is not in the class of path languages of attribute grammars; of course this implies that $L_{exp} \notin OUT(AG, TERMS)$.

We want to prove a hierarchy of output languages of attributed tree transducers with respect to the number of attributes. We do this by first proving a hierarchy of tree languages generated by cfhg's with respect to their rank.

In the following, let $\Sigma_k = \{a_1^{(1)}, ..., a_k^{(1)}, e^{(0)}\}$ and let $L_k = \{a_1^n(a_2^n(\cdots a_k^n(e))) \mid n \geqslant 0\} \subseteq T_{\Sigma_k}$.

LEMMA 6.2. *For every* $k \in \mathbb{N}$, $L_k \in \mathcal{TREE}_k$.

Proof. For every $k \in \mathbb{N}$ we construct a term-generating cfhg $G_k = (\Sigma_k, \Delta_k, S, P)$, such that $TREE(G_k) = L_k$, where $\Sigma_k = \{S^{(1)}, A^{(k)}, a_1^{(2)}, ..., a_k^{(2)}, e^{(1)}\}$, $\Delta_k = \{a_1^{(2)}, ..., a_k^{(2)}, e^{(1)}\}$. For $k = 2m$, $m \in \mathbb{N}$, i.e., k is even, $P = \{p_1, p_2, p_3\}$, where the rules p_1, p_2, p_3 are shown in Fig. 2.

For $k = 2m+1$, $m \in \mathbb{N}$, i.e., k is odd, $P = \{p'_1, p'_2, p'_3\}$, where the rules p'_1, p'_2, p'_3 are shown in Fig. 3.

Results of derivations by G_{2m} and G_{2m+1} are shown in Fig. 4. In this figure, undirected edges between two nodes mean that these nodes are identified. Clearly, $L(G_{2m}) = L_{2m}$ and $L(G_{2m+1}) = L_{2m+1}$ for $m \in \mathbb{N}$. Note that the arrangement of the hyperedges labeled by a_i's is chosen in correspondence with the notation used for attribute grammars and for attributed tree transducers (that is, nodes corresponding to inherited attributes are arranged on the left, and nodes corresponding to synthesized attributes are arranged on the right). ▮

LEMMA 6.3. *For every* $k \geqslant 1$, $L_k \notin \mathcal{TREE}_{k-1}$.

Proof. Recall the pumping lemma for context-free hypergraph languages as stated in Theorem IV.2.3 of [Hab92]. Now, assume that $L_k \in \mathcal{TREE}_{k-1}$ for a $k \geqslant 1$. Let $L'_k = \{tree(t) \mid t \in L_k\}$, where, as in Example 6.1, $tree(t)$ is the hypergraph representation of t. Then there is a cfhg G of rank $k-1$ such that $L(G) = L'_k$. Then, by the pumping lemma, for every sufficiently large hypergraph H, there is a partition into *FIRST*, *LINK*, and *LAST*, such that pumping of *LINK* will produce hypergraphs in L'_k. The part *LINK* is a hypergraph with at most $k-1$ external nodes and with a special hyperedge with unique label X which is incident with at most $k-1$ nodes. Also *LINK* is non-trivial, i.e., it contains at least one hyperedge. But then it must contain hyperedges labeled by $a_1, ..., a_k$ because otherwise pumping would lead to unequal numbers of edges labeled by the $a_1, ..., a_k$. No node of *LINK* can be incident with two edges labeled by distinct labels. This is true because pumping would produce more than one such node, but in L'_k there is exactly one

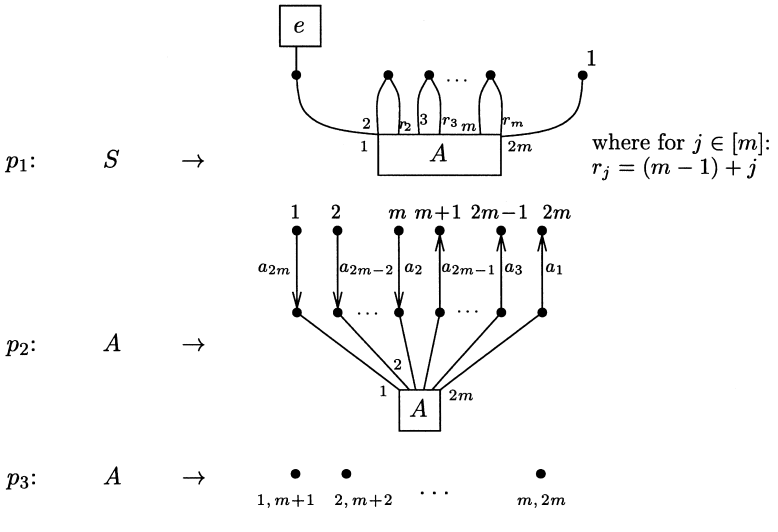

FIG. 2. Rules for the grammars G_k for even k's.

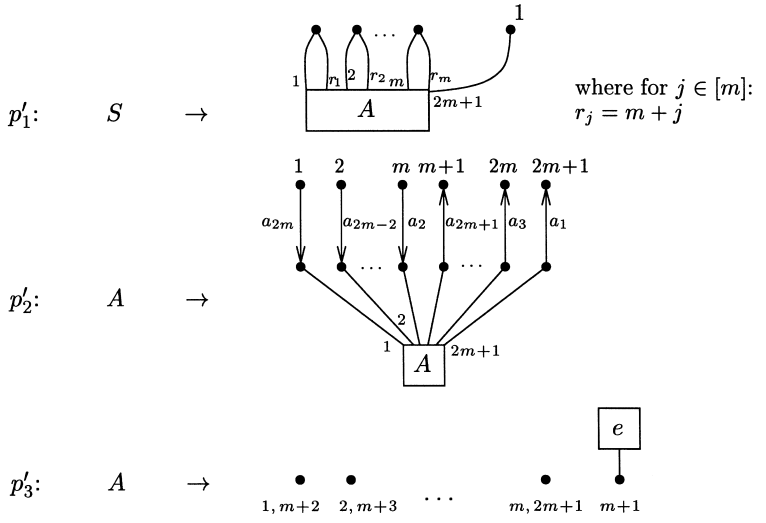

FIG. 3. Rules for the grammars G_k for odd k's.

node for every two consecutive labels a_l and a_{l+1}. For the same reason every internal node which is not incident with the X-edge must be incident with exactly two edges. Thus, for an $i \in [k]$ the edges labeled by a_i form a chain which is incident with at least two nodes that are either external or incident with the X-edge. As argued above no node is incident with edges with distinct labels and thus there must be at least $2k$ nodes that are external or incident with the X-edge. This is a contradiction to the fact that there are at most $2k-2$ such nodes, i.e., to the fact that $L_k \in \mathcal{TREE}_{k-1}$. ∎

Now, from Lemma 6.2 and Lemma 6.3, the following theorem is obtained.

THEOREM 6.4. *For every* $k \in \mathbb{N}$, $\mathcal{TREE}_k \subset \mathcal{TREE}_{k+1}$.

Note that this hierarchy of tree languages generated by cfhg's with respect to their rank is in contrast to the hierarchy of string languages generated by cfhg's with respect to their rank (cf. Theorem V.4.5 of [Hab92]), which increases only every second step. This is due to the definition of string-(hyper)graphs. That is,

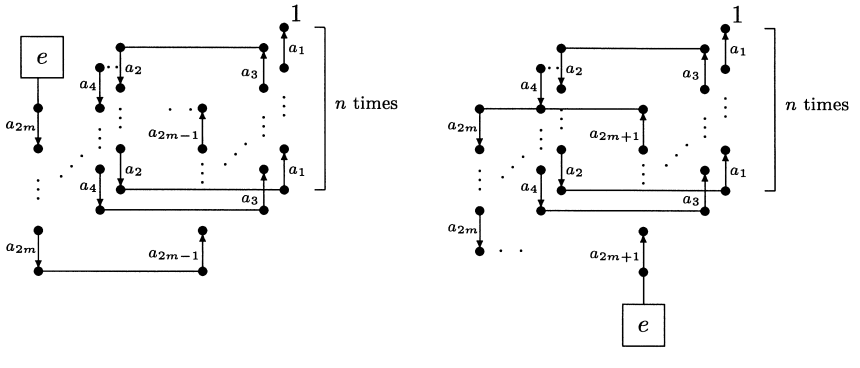

FIG. 4. Results of derivations of G_{2m} and G_{2m+1}.

string-graphs have, by definition, two external nodes. Thus, they are different from the jungles that represent the monadic trees of the languages L_k, which have one external node. This difference is comparable to the difference between the monoid of words on an alphabet A with binary concatenation and words with right concatenation by letters of A. A family of cfhg's G'_k which generate string-graphs, where the corresponding string languages are $\{a_1^n a_2^n \cdots a_k^n \mid n \in \mathbb{N}\}$, is very similar to the grammars G_k constructed in Lemma 6.2. The extra external node, which is needed by the definition of string-graphs, is needed exclusively to "hold on" to the end of the string-graphs. This difference was mentioned in Theorem 6.7 of [EH91] where it was shown that the class of string languages that can be generated by cfhg's that generate graph representations of strings that have two external nodes is equal to the class of string languages that can be generated by cfhg's that generate graph representations of strings that are of rank zero. From the results above it follows that by defining string-hypergraphs to have exactly one external node, a rank hierarchy is obtained that increases at each step.

We are now ready to prove a hierarchy for the output languages of attributed tree transducers with respect to the number of attributes.

THEOREM 6.5. *For every* $k \geqslant 1$, $OUT(AT_k) \subset OUT(AT_{k+1})$.

Proof. (1) For every $k \geqslant 1$, $L_k \in OUT(AT_k)$: Let $A = (Syn, Inh, \Sigma, \Delta, root, \alpha_1, R)$ be an attributed tree transducer with $\Sigma = \{\gamma^{(1)}, \#^{(0)}\}$ and $\Delta = \{a_1^{(1)}, ..., a_k^{(1)}, e^{(0)}\}$.

If $k = 2m$ for an $m \geqslant 1$, then $Syn = \{\alpha_1, ..., \alpha_m\}$, $Inh = \{\beta_1, ..., \beta_m\}$, $R_{root} = \{\alpha_1(\pi) \to \alpha_1(\pi 1), \beta_m(\pi 1) \to e\} \cup \{\beta_j(\pi 1) \to \alpha_{j+1}(\pi 1) \mid j \in [m-1]\}$, $R_\gamma = \{\alpha_j(\pi) \to a_{2j-1}(\alpha_j(\pi 1)) \mid j \in [m]\} \cup \{\beta_j(\pi 1) \to a_{2j}(\beta_j(\pi)) \mid j \in [m]\}$, and $R_\# = \{\alpha_j(\pi) \to \beta_j(\pi) \mid j \in [m]\}$.

If $k = 2m+1$ for an $m \in \mathbb{N}$, then $Syn = \{\alpha_1, ..., \alpha_{m+1}\}$, $Inh = \{\beta_1, ..., \beta_m\}$, $R_{root} = \{\alpha_1(\pi) \to \alpha_1(\pi 1)\} \cup \{\beta_j(\pi 1) \to \alpha_{j+1}(\pi 1) \mid j \in [m]\}$, $R_\gamma = \{\alpha_j(\pi) \to a_{2j-1}(\alpha_j(\pi 1)) \mid j \in [m+1]\} \cup \{\beta_j(\pi 1) \to a_{2j}(\beta_j(\pi)) \mid j \in [m]\}$, and $R_\# = \{\alpha_{m+1}(\pi) \to e\} \cup \{\alpha_j(\pi) \to \beta_j(\pi) \mid j \in [m]\}$.

The attributed tree transducer A works similar to the cfhg G_k defined in the proof of Lemma 6.2. It should be clear that $out(A) = L_k$.

(2) For every $k \geqslant 1$, $L_k \notin OUT(AT_{k-1})$: Assume that $L_k \in OUT(AT_{k-1})$. Then by Lemma 5.3 there is an attribute grammar G'_k with $out(G'_k, TERMS) = L_k$. The number of attributes of G'_k is equal to $k-1$, because the construction in the proof of Lemma 5.3 preserves the number of attributes. It then follows from the construction of Lemma 4.1 of [EH92] that there is a term-generating cfhg G' of rank $k-1$ with $TREE(G') = L_k$; i.e., $L_k \in \mathcal{TREE}_{k-1}$. However, by Lemma 6.3 this is not true, which contradicts our assumption. ∎

In [EF81] it is shown that the classes of output languages of n-visit attribute grammars form a proper hierarchy with respect to n. An attribute grammar (attributed tree transducer, respectively) is n-visit, if its attributes can be evaluated by a walk through the derivation tree (input tree, respectively) in such a way that each subtree is visited at most n times. There seems to be a connection between the number n of visits of an attributed tree transducer A and the number k of attributes

of A. For instance, a subtree need not be visited more than k times. If the output alphabet is monadic, then the class of translations realized by n-visit attributed tree transducers is clearly included in AT_{2n}, because in each visit at most 2 attributes can be evaluated. However, for nonmonadic output alphabets there seems to be no connection between the number of visits and the number of attributes, as can be seen as follows. Consider the language $L_{fork} = \{\delta(a_1^m(e), a_2^m(e), ..., a_k^m(e)) \mid m \geqslant 0\}$, where δ is a symbol of rank k, $a_1, ..., a_k$ are symbols of rank 1, and e is a symbol of rank 0. This language can be generated by an attributed tree transducer with k synthesized attributes and no inherited attributes. However, it seems that it cannot be generated by an attributed tree transducer with less than k attributes (cf. [Küh97b], where this result is proved for producing attributed tree transducers). On the other hand, L_{fork} can clearly be generated by a 1-visit attributed tree transducer.

Let us now shortly discuss the relationship between the generating power of ssur attributed tree transducers and that of (unrestricted) attributed tree transducers.

The ssur property for attributed tree transducers is defined similar to the original definition by Ganzinger, only that we allow that every outside attribute may be used *at most once* instead of his *exactly once* restriction. This was already discussed in Section 2.3 of [Gie88]. In [Küh97a] ssur was defined in this way for various types of tree transducers and in particular for attributed tree transducers. It is also shown there that the composition results of [Gan83, Gie88] mentioned in the beginning of Section 5 still hold. More formally, an attributed tree transducer $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ is *syntactic single used restricted* if for every $\sigma \in \Sigma$, $c(\pi j) \in outs_\sigma$, and $r_1, r_2 \in R_\sigma$ the following implication holds. If $rhs(r_1)/u_1 = rhs(r_2)/u_2 = c(\pi j)$ for some $u_1 \in O(rhs(r_1))$ and $u_2 \in O(rhs(r_2))$, then $r_1 = r_2$ and $u_1 = u_2$. The class of translations that can be realized by ssur attributed tree transducers is denoted by *ssur-AT* and for a $k \in \mathbb{N}$ the class of translations that can be realized by ssur attributed tree transducers with at most k attributes is denoted by *ssur-*AT_k.

The attributed tree transducer A defined in the proof of Theorem 6.5 has the ssur property; i.e., $L_k \in OUT(ssur\text{-}AT_k)$. Thus, the output languages of ssur attributed tree transducers also form a hierarchy with respect to the number of attributes; i.e., $OUT(ssur\text{-}AT_k) \subset OUT(ssur\text{-}AT_{k+1})$ for $k \in \mathbb{N}$.

LEMMA 6.6. $OUT(AT_1) - OUT(ssur\text{-}AT) \neq \emptyset$

Proof. Let $A = (Syn, Inh, \Sigma, \Delta, root, a, R)$ be the attributed tree transducer with $Syn = \{a\}$, $Inh = \emptyset$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, $\Delta = \{\sigma^{(2)}, \alpha^{(0)}\}$, $R_{root} = \{a(\pi) \to a(\pi 1)\}$, $R_\gamma = \{a(\pi) \to \sigma(a(\pi 1), a(\pi 1))\}$, and $R_\alpha = \{a(\pi) \to \alpha\}$. Then $out(A)$ is the set B of all full binary trees over Δ. Since $|Syn \cup Inh| = 1$, $B \in OUT(AT_1)$.

Let us now show that $B \notin OUT(ssur\text{-}AT)$. Consider the attributed tree transducer $A_{yield} = (\{a\}, \{b\}, \{\sigma^{(2)}, \alpha^{(0)}\}, \{\gamma^{(1)}, \alpha^{(0)}\}, root, a, R)$ with $R_{root} = \{a(\pi) \to a(\pi 1), b(\pi 1) \to \alpha\}$, $R_\sigma = \{a(\pi) \to a(\pi 2), b(\pi 1) \to b(\pi), b(\pi 2) \to a(\pi 1)\}$, and $R_\alpha = \{a(\pi) \to \gamma(b(\pi))\}$. Then A_{yield} computes the yield, i.e., the frontier of a binary tree. More precisely, for an input tree s with n leaves $\tau_{A_{yield}}(s) = \gamma^n\alpha$. Now assume that there is an ssur attributed tree transducer A with $out(A) = B$. Then by the composition

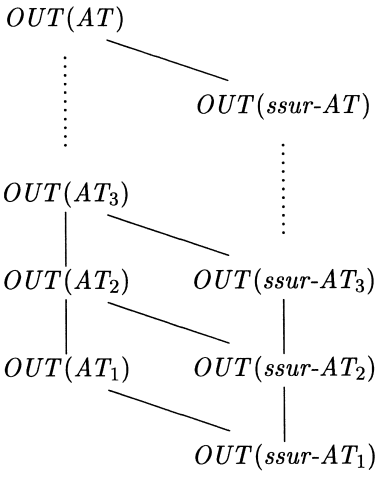

FIG. 5. A Hasse diagram for the generating power of attributed tree transducers.

result of [Gie88] mentioned in the beginning of Section 5, $\tau_A \circ \tau_{A_{yield}} \in AT$. Now $ran(\tau_A \circ \tau_{A_{yield}}) = \{\gamma^{2^n}\alpha \mid n \geqslant 0\} = L_{exp}$. But this contradicts the fact that $L_{exp} \notin OUT(AT)$, which was shown in Example 6.1 and therefore contradicts the assumption that there is an ssur attributed tree transducer A with $out(A) = B$. ∎

By Lemma 6.6 it follows that the diagram shown in Fig. 5 is a Hasse diagram.

THEOREM 6.7. *Figure 5 is a Hasse diagram.*

7. FURTHER RESEARCH TOPICS

In [KM] we want to investigate how the two pumping lemmas, the one for context-free hypergraph languages [HK87, Hab92] and the one for output languages of (restricted) attributed tree transducers [KV94] are related to each other. Maybe through this investigation it will become clear how the class of output languages of visiting and producing attributed tree transducers is related to the class of output languages of attributed tree transducers.

The proof of Theorem 6.5 also shows that the term-generating power of attribute grammars forms a hierarchy with respect to the number of attributes. It would be interesting to know the precise relationship between $OUT(AT_k)$ and $OUT(AG_k, TERMS)$, where $OUT(AG_k, TERMS)$ denotes the class of tree languages that can be generated by attribute grammars with at most k attributes. By Lemma 5.3, $OUT(AT_k) \subseteq OUT(AG_k, TERMS)$. From Lemmas 5.1 and 5.5 it follows that $OUT(AG_{s,i,n}, TERMS) \subseteq OUT(AT_{s \cdot n, i})$, where $OUT(AG_{s,i,n}, TERMS)$ is the class of output languages of (term-generating) attribute grammars with at most s synthesized attributes, at most i inherited attributes, and at most n nonterminals in the underlying context-free grammar and $OUT(AT_{s \cdot n, i})$ is the class of output languages of attributed tree transducers with at most $s \cdot n$ synthesized attributes and at most i inherited attributes. For the other direction of this inclusion the size of the input alphabet is important; however, it seems that the results of this paper are not sufficient to make more precise statements about this relationship.

ACKNOWLEDGMENTS

I thank Zoltan Fülöp, Armin Kühnemann, and Heiko Vogler for fruitful discussions. I am grateful to Joost Engelfriet for the tremendous amount of helpful comments and suggestions for improvements on an earlier version of this paper. In fact, some results of the present paper are exclusively due to those comments.

Received December 20, 1996; final manuscript received February 25, 1998.

REFERENCES

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

[Dán96] Dányi, G. (1996), On domain and range tree languages of superlinear deterministic top-down tree transformations, *Acta Cybernetica* **12**, 261–277.

[DJL88] Deransart, P., Jourdan, M., and Lorho, P. (1988), "Attribute Grammars, Definitions and Bibliography," LNCS, No. 323, Springer-Verlag, Berlin/New York.

[DKH97] Drewes, F., Kreowski, H.-J., and Habel, A. (1997), Hyperedge replacement graph grammars, *in* "Handbook of Graph Grammars and Computing by Graph Transformation" (G. Rozenberg, Ed.), Vol. 1, Chap. 2, pp. 95–162, World Scientific, Singapore.

[EF81] Engelfriet, J. and Filè, G. (1981), Passes and paths of attribute grammars, *Inform. and Control* **49**, 125–169.

[EH91] Engelfriet, J. and Heyker, L. (1991), The string-generating power of context-free hypergraph grammars, *J. Comput. System Sci.* **43**, 328–360.

[EH92] Engelfriet, J. and Heyker, L. (1992), Context-free hypergraph grammars have the same term-generating power as attribute grammars, *Acta Informatica* **29**, 161–210.

[Eng75] Engelfriet, J. (1975), "Tree Automata and Tree Grammars," lecture notes, DAIMI FN-10, University of Aarhus.

[Eng76] Engelfriet, J. (1976), Surface tree languages and parallel derivation trees, *Theoret. Comput. Sci.* **2**, 9–27.

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

[Eng97] Engelfriet, J. (1997), Context-free graph grammars, *in* "Handbook of Formal Languages" (G. Rozenberg and A. Salomaa, Eds.), Vol. 3, Chap. 3, Springer-Verlag, Berlin/New York.

[ERS80] Engelfriet, J., Rozenberg, G., and Slutzki, G. (1980), Tree tranducers, L systems, and two-way machines, *J. Comput. System Sci.* **20**, 150–202.

[FHVV93] Fülöp, Z., Herrmann, F., Vágvölgyi, S., and Vogler, H. (1993), Tree transducers with external functions, *Theoret. Comput. Sci.* **108**, 185–236.

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

[Gan83] Ganzinger, H. (1983), Increasing modularity and language-independency in automatically generated compilers, *Sci. Comput. Program.* **3**, 223–278.

[Gie88] Giegerich, R. (1988), Composition and evaluation of attribute coupled grammars, *Acta Informatica* **25**, 355–423.

[GS84] Gécseg, F. and Steinby, M. (1989), "Tree Automata," Akadémiai Kiadó, Budapest.

[Hab92] Habel, A. (1992), "Hyperedge-Replacement: Grammars and Languages," LNCS, Vol. 643, Springer-Verlag, Berlin/New York.

[HK87] Habel, A. and Kreowski, H.-J. (1987), Some structural aspects of hypergraph languages generated by hyperedge replacement, *in* "Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science, STACS 87" (F. Brandenburg, Ed.), LNCS, Vol. 247, pp. 207–219, Springer-Verlag, Berlin/New York.

[KM] Kühnemann, A. and Maneth, S. (in preparation), Relations and applications of pumping lemmata for attributed tree transducers and hyperedge-replacement grammars.

[Knu68] Knuth, D. E. (1968), Semantics of context-free languages, *Math. Systems Theory* **2**, 127–145. [Corrections in (1971), *Math. Systems Theory* **5**, 95–96]

[Küh97a] Kühnemann, A. (1997), "Berechnungsstärken von Teilklassen primitiv-rekursiver Programmschemata," Ph.D. thesis, Technical University Dresden.

[Küh97b] Kühnemann, A. (1997), A two-dimensional hierarchy for attributed tree transducers, *in* "Fundamentals of Computation Theory, FCT'97, 11th International Symposium, Kraków, Poland, Proceedings" (B. S. Chlebus and L. Czaja, Ed.), LNCS, Vol. 1279, pp. 281–292, Springer-Verlag, Berlin/New York.

[KV94] Kühnemann, A. and Vogler, H. (1994), A pumping lemma for output languages of attributed tree transducers, *Acta Cybernetica* **11**, 261–305.

[Man96] Maneth, S. (1996), "On the Generating Power of Deterministic Tree Transducers," Technical Report TUD/FI96/19, Technical University of Dresden.

[New42] Newman, M. H. A. (1942), On theories with a combinatorial definition "equivalence," *Ann. of Math.* **43**(2), 223–243.

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