TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

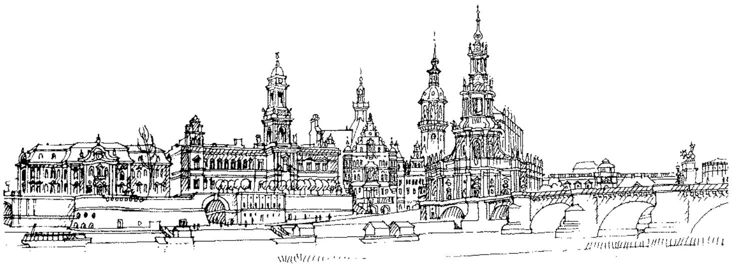
No. TUD / FI 96 / 19 - November 1996

Sebastian Maneth
Grundlagen der Programmierung
Institut für Softwaretechnik I

Technische Berichte
Technical Reports

ISSN 1430-211X

On the Generating Power of Deterministic Tree Transducers



Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden
Germany

URL: http://www.inf.tu-dresden.de/

On the Generating Power of Deterministic Tree Transducers

SEBASTIAN MANETH*

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

Abstract

Attributed tree transducers are abstract models 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. generating power. We give two applications of our results: (1) there is no attributed tree transducer which computes the set of all monadic trees with exponential height as output language and (2) there is a hierarchy of output languages of attributed tree transducers w.r.t. their number of attributes. The latter result is achieved by proving a hierarchy of tree languages generated by context–free hypergraph grammars w.r.t. their rank.

*The work of this author has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Vo 461/3–1.

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 occur as output languages of such string–to–tree translations of attribute grammars. This class is denoted by $OUT(AG, TERMS)$.

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, DKH96, Eng96]) have the same term–generating power as attribute grammars.

In this paper we will show that also 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 deterministic, total 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 concatenation 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 do 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. But we show that this is not true, i.e., that $OUT(AG, TERMS) = OUT(AT)$. Let us discuss this. For a class $\mathcal{L}$ of tree languages let $AT(\mathcal{L})$ denote the class of output languages of attributed tree transducers taking languages from $\mathcal{L}$ as input. Now, the term–generating power of attribute grammars is exactly the class $AT(DT)$, where DT is the class of derivation tree languages of context–free grammars (in [Tha70] and [Eng75b] these are called *local* tree languages), i.e., $OUT(AG, TERMS) = AT(DT)$. Next, it is well known from [MW67], that the class DT is included in the class $RECOG$ of recognizable tree languages, and therefore $AT(DT) \subseteq AT(RECOG)$ holds. We will prove that $AT(RECOG) \subseteq OUT(AT)$, and $OUT(AT) \subseteq OUT(AG, TERMS)$, and thus in total:

$$OUT(AG, TERMS) = AT(DT) \subseteq AT(RECOG) \subseteq OUT(AT) \subseteq OUT(AG, TERMS)$$

Of course, $OUT(AT) \subseteq OUT(AG, TERMS)$ is straightforward: for every attributed tree transducer A with input ranked alphabet Σ there is an attribute grammar G with $out(A) = out(G, TERMS)$ (where *out* is the equivalent to OUT but for a particular attributed tree transducer or attribute grammar). The context–free grammar G_0 on which G is based, is constructed such that the set of derivation trees of G_0 is isomorphic to the set of all trees over Σ. The rules for the attributes can simply be taken over from A to G.

The inclusion which remains to be proved is $AT(RECOG) \subseteq OUT(AT)$. The "lemma tree" in Figure 1 shows the involved results and their dependencies, where d–T denotes the class of translations realized by deterministic top–down tree transducers. The prefix t denotes totality, the prefix l denotes linearity, and the prefix r denotes a restricted form of relabeling.

Let us first explain the notion of top–down tree transducers and of some subclasses that we consider. Top–down tree transducers were introduced in [Rou70] and [Tha70] as model for syntax–directed translation and transformational grammars. They are a natural extension of generalized sequential machine mappings (GSM) [GR62] to trees. Seen as a term rewriting system, a top–down tree transducer is a left–linear, confluent, terminating, constructor–based [You89], and strictly not subunifiable [Ech90] term rewriting system, such that every function symbol (or *state* due to the relation to GSM's) is of rank 1 and for every state q and constructor (or *input symbol*) σ there is at most one rewrite rule of the form

$$q(\sigma(x_1, \ldots, x_k)) \to \xi,$$

where ξ is a term over output symbols and state calls of the form $p(x_i)$ such that p is a state and x_i is a variable from the left–hand side. Such a transducer is *deterministic*, because there is at most one rule per state and input symbol. If we replace the requirement "at most" by "exactly", then the transducer is deterministic and *total*. A finite state relabeling (see, e.g., [ERS80]) is a top–down tree transducer, where all rules are of the form $q(\sigma(x_1, \ldots, x_k)) \to \delta(q_1(x_1), \ldots, q_k(x_k))$. The *restricted relabeling* as we define it, allows only *one* such rule per input symbol, whereas it also allows rules of the form $q(\sigma(x_1, \ldots, x_k)) \to \xi$ with $\xi \in T_\Delta$, i.e., ξ contains only output symbols.

Let us now go back to Figure 1 and recall that $AT(RECOG) \subseteq OUT(AT)$ remains to be proved. For a tree transducer A and a recognizable tree language L this means, that in respect to the generating power we can get rid of the recognizable tree language L. This possibility might facilitate proofs as it was already mentioned in [ERS80], where a

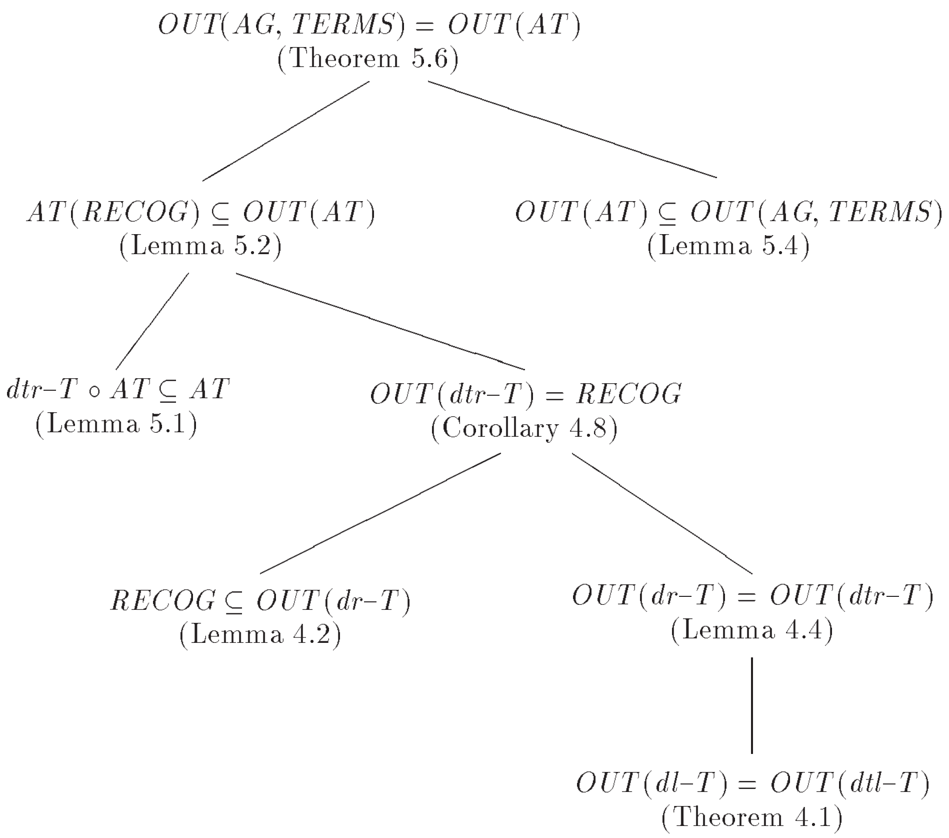


Figure 1: Dependencies of results presented in this paper.

similar result is obtained for (partial) top–down tree transducers, i.e., in Theorem 3.2.1 of [ERS80] it is shown that $T(RECOG) \subseteq OUT(T)$. This is has been proved as follows. Let L be a recognizable tree language and let M be a top–down tree transducer. Then we may assume that there is a deterministic 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 term is in L by encoding information 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/output. We use the following technique by Ganzinger and Giegerich [Gan83, Gie88] to prove Lemma 5.1: The right–hand sides of the top–down finite tree automaton's rules are translated by the attributed tree transducer A. In this way we obtain an attributed tree transducer A' which has the same output as A taking the domain of M as input.

However, in general, the top–down finite tree automaton M is *partial* and *nondeterministic*. And thus, the resulting attributed tree transducer A' is partial and nondeterministic, too. This is not sufficient for answering our question of $AT(RECOG) \subseteq OUT(AT)$, because, as mentioned above, AT denotes the class of translations induced by total, deterministic attributed tree transducers. Therefore, in order to apply Lemma 5.1, we have to find a class of top–down tree transducers which is *total* and *deterministic* and the output of which includes $RECOG$. The class $dtr\text{–}T$ of translation induced by total, deterministic, relabeling top–down tree transducers is such a class. In fact, it holds that the class $OUT(dtr\text{–}T)$ is exactly $RECOG$ (Corollary 4.8, see Figure 1).

The direction $OUT(dtr\text{–}T) \subseteq RECOG$ of Corollary 4.8 is immediately obtained by the facts that deterministic, total, relabeling top–down tree transducers are linear and that recognizable tree languages are closed under linear top–down tree transductions [Tha69].

The proof of the other direction proceeds in two steps: given a top–down finite tree automaton we construct a deterministic, total, relabeling top–down tree transducer by first "determinizing" (cf. Lemma 4.2) it and then "totalizing" (cf. Lemma 4.4) it. In fact, the latter result can be obtained for a much greater class of tree translations. Namely, in Theorem 4.1 it is proved that $OUT(dl\text{–}T) = OUT(dtl\text{–}T)$ holds.

A great number of publications deals with the classes of tree translations which can be realized by top–down tree transducers (e.g., [Eng75a, Eng77, Eng80, ERS80, GS84]). However, for the corresponding *classes of output languages*, there are only a few results. Let us now consider some related work on the generating power of tree transducers. Some of them are:

- The class of linear top–down tree transductions preserves regularity [Tha69]. This means in particular, that the class of output languages of linear top–down tree transductions is included in $RECOG$.
- There are various results on surface languages, i.e., output languages of tree transducers under the input of recognizable tree languages (see [Eng75b, Eng76, GS84] for an overview).

We give two applications of our results. First, by the equality result of [EH92] for the term–generating power of context–free hypergraph grammars and that of attribute grammars, it follows that context–free hypergraph grammars and attributed tree transducers have the same term–generating power. Therefore, we can make use of the pumping lemma for context–free hypergraph languages [HK87, Hab92]. This allows us to prove, that the set L_{exp} of all monadic trees with exponential height is not in $OUT(AT)$. This result sharpens a result of [KV94], where it has been proved that L_{exp} can not be generated by restricted attributed tree transducers. Second, it allows us to prove a hierarchy of tree languages generated by context–free hypergraph grammars w.r.t. their rank. This hierarchy implies a hierarchy of output languages of attributed tree transducers w.r.t. their number of attributes.

This paper is organized as follows. Section 2 fixes some notions and notations used throughout the paper. Top–down and attributed tree transducers and attribute grammars are defined in Section 3. Some 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 applications of our results are given in Section 6 and finally Section 7 lists some further research topics.

2 Preliminaries

2.1 General Notations

The *set of non–negative integers* is denoted by $I\!N$. The *empty set* is denoted by $\emptyset$. For some $j \in I\!N$, $[j]$ denotes the set $\{1, \dots, j\}$; therefore $[0] = \emptyset$. 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 $\mathcal{P}(A)$, is the set of all subsets of A. For two sets A and B, the set $A - B$ denotes the set $\{a \in A \mid a \notin B\}$.

Let $f : A \rightarrow B$ be a function. The *domain of f*, denoted by $dom(f)$, is the set $\{a \in A \mid f(a) = b \text{ for some } b \in B\}$. The *range of f*, denoted by $ran(f)$, is the set $\{b \in B \mid f(a) = b \text{ for some } a \in A\}$. Thus, we use the notion of function in the sense of partial functions. If $dom(f) = A$, then f is called *total* function. Let $f : A \rightarrow B$ and $g : B \rightarrow C$ be functions. The *composition* $g \circ f : A \rightarrow C$ is defined by $(g \circ f)(x) = g(f(x))$ for all $x \in A$. Note that, as usual, $(g \circ f)(x)$ is defined only if $x \in dom(f)$ and $f(x) \in dom(g)$.

2.2 Ranked Alphabets, Variables and Trees

A pair $(\Omega, rank_\Omega)$ is called *ranked alphabet* if Ω is a finite set and $rank_\Omega : \Omega \rightarrow I\!N$ is a total function. For every symbol $f \in \Omega$, the natural number $rank_\Omega(f)$ is called the *rank of f*. For every $i \geq 0$, the set $\Omega^{(i)} \subseteq \Omega$ consists of all symbols in Ω that are of rank i. A ranked alphabet can be defined by either (i) enumerating the finitely many non–empty subsets $\Omega^{(i)}$, or by (ii) a set Γ of symbols which are indexed by their unambiguous rank. 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 Γ. If, for a ranked alphabet Γ, $\Gamma = \Gamma^{(1)}$, then Γ is called *unary*.

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: $S \subseteq T$ and if $f \in \Delta^{(n)}$ with $n \geq 0$ and $t_1, \dots, t_n \in T$, then $f(t_1, \dots, t_n) \in T$. If $S = \emptyset$, then we simply write T_Δ.

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 of t*, denoted by $O(t)$, is a subset of $I\!N^*$ which is inductively defined as follows: (i) if $t = x$ with $x \in S$, then $O(t) = \{\varepsilon\}$, (ii) if $t = f$ with $f \in \Omega^{(0)}$, then $O(t) = \{\varepsilon\}$, and (iii) if $t = f(t_1, \dots, t_n)$ with $f \in \Omega^{(n)}$ and $n > 0$, and for all $i \in [n] : t_i \in T_\Delta(S)$, then $O(t) = \{\varepsilon\} \cup \bigcup_{i \in [n]} \{iu \mid u \in O(t_i)\}$. Obviously this representation becomes ambiguous if the ranked alphabet contains a symbol with rank larger than 9. Then one should be adapting *Dewey's notation* by placing dots between the i's.

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 in which every occurrence of the subtree t' is replaced by s, is denoted by $t[t' \leftarrow s]$. When using the notion $t[x \leftarrow s]$, 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 notion $[x \leftarrow s \mid P_1, \dots, P_n]$, where $P_1, \dots, P_n$ 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.

Let t be a tree. The *root of* t is denoted by $root(t)$.

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 some $b \in A$, there is an $a \in A$ such that $b \Rightarrow^* a$ and a is irreducible, then a is called a *normal form (w.r.t.* $\Rightarrow$*) of* b. If there is exactly one normal form of b, then this is denoted by $nf(\Rightarrow, b)$.

2.3 Tree Translations

Let Σ and Δ be ranked alphabets. A function $\tau : T_\Sigma \to T_\Delta$ 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 $\{\tau(t) \mid t \in L\}$. For a class of translations Ψ we denote by $OUT(\Psi)$ the corresponding class of output languages, i.e., $OUT(\Psi) = \{ran(\tau) \mid \tau \in \Psi\}$. Let $\mathcal{L}$ be a class of tree languages. The class of output languages of translations in Ψ taking languages of $\mathcal{L}$ as input, denoted by $\Psi(\mathcal{L})$, is the class $\{\tau(L) \mid L \in \mathcal{L}\}$.

2.4 Context–Free Grammars, Semantic Domain

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

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 and $\Sigma - \Delta$ is the nonterminal alphabet. P is a finite set of productions of the from $X \to H$, where $X \in \Sigma - \Delta$, H is a hypergraph over Σ (see [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. Let $k \in I\!N$, then G *is of rank* k, if k is equal to the maximum of the ranks of the (used) nonterminals of G.

Let us only fix some notations concerning cfhg's, for more details cf. [EH92]. Let G be a cfhg. In general, G generates a set of hypegraphs. 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. The *tree–language generated by* G is denoted by $TREE(G)$. Let $k \in I\!N$. The class of all tree languages generated by (tree–generating) cfhg's of rank less than or equal to k is denoted by $\mathcal{TREE}_k$.

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 mapping $\gamma_D : V^n \to V$ with $n = rank_\Gamma(\gamma)$; i.e., D is a Γ–algebra, where Γ is a one–sorted signature (see, e.g., [CF82]).

3 Top–down Tree Transducers, Attributed Tree Transducers, and Attribute Grammars

In this section we will define the notion of top–down tree transducers, attributed tree transducers and attribute grammars.

Definition 3.1 (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; 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; it is called *initial state*.
- R is a finite set of *rules* of the following form: $q(\sigma(x_1, \ldots, x_k)) \rightarrow \xi$, with $q \in Q, k \geq 0, \sigma \in \Sigma^{(k)}, \xi \in T_\Delta(\{p(x_i) \mid p \in Q, i \in [k]\})$. Such a rule is called (q, σ)*–rule* or just σ–rule. The right–hand side ξ of a (q, σ)–rule is denoted by $rhs(q, \sigma)$. □

Definition 3.2 (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 $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, $t_1, \ldots, t_k \in T_\Sigma$ and an occurrence $u \in O(s_1)$ with $s_1[u] = q(\sigma(t_1, \ldots, t_k))$ such that

$$s_2 = s_1[u \leftarrow rhs(q, \sigma)[x_i \leftarrow t_i \mid i \in \{1, \ldots, k\}]].$$

Where appropriate we might, for a particular derivation step, also use the occurrence u and/or the (q, σ) rule, as index for $\Rightarrow_M$. □

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 3.3 (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* which can be realized by top–down tree transducers is denoted by T.

For a top–down tree transducer M, 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 take a look at a small example of a top–down tree transducer.

Example 3.4 (for a top–down tree transducer)

Let $M = (\{q_0, q\}, \{suc^{(1)}, 0^{(0)}\}, \{+^{(2)}, suc^{(1)}, 0^{(0)}\}, q_0, R)$ be a top–down tree transducer, where the set R of rules contains the following rules

$$\begin{array}{lcl} q_0(0) & \to & suc(0) \\ q_0(suc(x_1)) & \to & +(q_0(x_1), q(x_1)) \\ q(0) & \to & 0 \\ q(suc(x_1)) & \to & q_0(x_1). \end{array}$$

This tree transducer computes the Fibonacci function, i.e., given a natural number n, represented as input tree $suc^n(0)$, it produces as output the n-th Fibonacci number, represented as a tree over the symbols $+$, suc, and 0 which are interpreted in the usual way.

Consider the rules of M. Since, due to the syntactic restrictions on rules of top–down tree transducers, it is only possible to call the predecessor of the left–hand side's argument in the right–hand side, we need, besides q_0, a second state q which computes q_0 on the predecessor of the predecessor of the left–hand side's argument. In other words, we turn course of value recursion into simultaneous recursion.

Consider, for example, the input tree $suc(suc(0))$, then a derivation by M looks as follows: $q_0(suc(suc(0))) \Rightarrow_M +(q_0(suc(0)), q(suc(0))) \Rightarrow_M^2 +(+(q_0(0), q(0)), q_0(0)) \Rightarrow_M^3 +(+(suc(0), 0), suc(0))$. And $\tau_M = \{(suc^n(0), fib(n)) \mid n \geq 0\}$, where $fib : I\!N \to T_\Sigma$ is the function defined recursively by the following three equations: $fib(0) = suc(0)$, $fib(1) = +(suc(0), 0)$, and $fib(n) = +(fib(n-1), fib(n-2))$ for $n \geq 2$. Of course $out(M) = \{fib(n) \mid n \geq 0\}$. □

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. Definition 3.1.7 or [ERS80]), i.e., there is no transition function as 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 every 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, for every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, there is at most one $q \in Q$ such that $rhs(q, \sigma)$ has the form $q(\sigma(x_1, \ldots, x_k)) \to \delta(q_1(x_1), \ldots, q_k(x_k))$ with $q, q_1, \ldots, q_k \in Q$ and

$\delta \in \Delta^{(k)}$ and for every other $p \in Q$, $rhs(p, \sigma) \in T_\Delta$, then M is called *restricted finite–state relabeling*. The corresponding class of translations is denoted by r–T. In the following we will simply use the term *relabeling* top–down tree transducer.

- If $\Sigma = \Delta$ and every rule in R has the form $q(\sigma(x_1, \ldots, x_k)) \to \sigma(q_1(x_1), \ldots, q_k(x_k))$ with $q, q_1, \ldots, q_k \in Q$ 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. E.g., the class of translations realized by deterministic, total top–down tree transducers is denoted by dt–T. □

Recall that if M is a deterministic top–down tree transducer, then the translation τ_M is a function and if M is a deterministic, total top–down tree transducer, then the translation $\tau_{M'}$ is a total function (see, e.g., [FHVV93]). Note, however, that in the case of a deterministic top–down tree transducer M, $nf(\Rightarrow_M, q(s))$ does not have to be in T_Δ. For instance, $M = (\{q_0\}, \Sigma, \Delta, q_0, R)$ with $\Sigma = \Delta = \{\alpha^{(0)}, \beta^{(0)}\}$, and $R = \{q_0(\alpha) \to \beta\}$. Then, $nf(\Rightarrow_M, q_0(\beta)) = q_0(\beta) \notin T_\Delta$.

The tree transducer in Example 3.4 is a deterministic, total top–down tree transducer, i.e., $\tau_M \in dt$–T. Note that it is neither a restricted relabeling nor a linear top–down tree transduction because the first rule has two occurrences of the variable x_1 in its right–hand side.

The class of domains/output languages of top–down finite tree automata is called *class of recognizable tree languages* and it is denoted by $RECOG$. Note that the input and/or output alphabet of a top–down tree transducer may be empty and this implies that $\{\emptyset\} \in OUT(\Psi)$ for all classes Ψ of translations induced by top–down tree transducers that we consider here. The reason that we allow this is that we are comparing classes of output languages of top–down tree transducers with the class of recognizable tree languages and it holds that $\{\emptyset\} \in RECOG$.

Let us now define some auxiliary notions.

Definition 3.6 (auxiliary notions)

Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a deterministic top–down tree transducer.

- For every state $q \in Q$, the *set of input trees on which q is successful*, denoted by $I(q)$, is the set $\{s \in T_\Sigma \mid q(s) \Rightarrow_M^* t, t \in T_\Delta\}$.

- Let $q \in Q$. If $I(q) \neq \emptyset$, then let s_q be an arbitrary but fixed tree in $I(q)$ and let the tree $t_q \in T_\Delta$ be determined by $q(s_q) \Rightarrow_M^* t_q$. Note that it is decidable whether $I(q) = \emptyset$, because $I(q)$ is a recognizable tree language. This follows from the fact that the domain of a translation induced by a deterministic top–down tree transducer is a recognizable tree language (see Theorem 1 of [Rou70]) and $I(q) = dom(M_q)$ with $M_q = (Q, \Sigma, \Delta, q, R)$. □

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

Definition 3.7 (reduced)
Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a deterministic top–down tree transducer. M is called *reduced*, if the following two conditions hold:

(i) For every state $q \in Q$, there are $s, s' \in T_\Sigma$, $s'' \in T_{Q \cup \Sigma \cup \Delta}$, and a $t \in T_\Delta$, such that $q(s')$ occurs in s'' and $q_0(s) \Rightarrow_M^* s'' \Rightarrow_M^* t$, i.e., Q does not contain *superfluous states.*

(ii) For every input symbol $\sigma \in \Sigma^{(k)}$, $k \geq 0$, there are $t_1, \ldots, t_k \in T_\Sigma$, $q \in Q$, and a $t \in T_\Delta$ such that $q(\sigma(t_1, \ldots, t_k)) \Rightarrow_M^* t$, i.e., Σ does not contain *superfluous input symbols.* □

Observation 3.8 Note that 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 useless states and input symbols and the rules containing them. The construction is similar to the construction of a reduced context–free grammar.

Let us now turn to attributed tree transducers. Before we define the notion of attributed tree transducers let us define the set of right–hand sides.

Definition 3.9 (Right–hand sides of attributed tree transducers)
Let S and I be unary ranked alphabets, Δ a ranked alphabet, and $k \geq 0$ 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 $T_{S \cup I \cup \Delta}(\{\pi, \pi 1, \ldots, \pi k\})$ such that the following conditions hold:

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

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

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

Definition 3.10 (Attributed tree transducer)
An *attributed tree transducer* A is a tuple $(Syn, Inh, \Sigma, \Delta, root, a_0, R)$, where

- Syn and Inh are unary ranked alphabets, the elements of which are called *synthesized attributes* and *inherited attributes*, respectively.
- Σ and Δ are ranked alphabets; 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 *root marker*
- a_0 is a symbol of rank 1 with $a_0 \notin Syn \cup Inh$, called *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 $a_0(\pi) \to \xi$ with $\xi \in RHS(Syn, \emptyset, \Delta, 1)$ and for every $b \in Inh$ the set R_{root} contains exactly one rule of the form $b(\pi 1) \to \xi$ with $\xi \in RHS(Syn, \emptyset, \Delta, 1)$.
2. For every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$ and $a \in Syn$, the set R_σ contains exactly one rule of the form $a(\pi) \to \xi$ with $\xi \in RHS(Syn, Inh, \Delta, k)$.
   For every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, $b \in Inh$ and $j \in [k]$, the set R_σ contains exactly one rule of the form $b(\pi j) \to \xi$ with $\xi \in RHS(Syn, Inh, \Delta, k)$.

In the rules of R, the symbol π is called *path variable.* A particular rule $\alpha(p) \to \xi$ in R_σ with $\sigma \in \Sigma^{(k)}$, $\alpha \in Syn \cup Inh$ and $p \in \{\pi, \pi 1, \ldots, \pi k\}$ is called (α, p, σ)–rule; the tree ξ is denoted by $rhs(\alpha, p, \sigma)$.

For every $\sigma \in \Sigma^{(k)}$, the *set of inside attributes of* σ is the set $\{\alpha(\pi) \mid \alpha \in Syn\} \cup \{\beta(\pi i) \mid \beta \in Inh, i \in [k]\}$.

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

Note that our definition of attributed tree transducers in Definition 3.10 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 3.10. However, in the following we will simply use the notion of attributed tree transducer as defined.

Definition 3.11 (Derivation relation induced by A)

Let $A = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be an attributed tree transducer, let $\Sigma' \supseteq \Sigma$ be a ranked alphabet and let s be tree in $T_{\Sigma'}$‡. The *derivation relation induced by A on s*, denoted by $\Rightarrow_{A,s}$, is a binary relation over $T_{I \cup \Delta}$, where $I = \{\beta(w) \mid \beta \in Syn \cup Inh, w \in O(s)\}$, such that $\xi_1 \Rightarrow_{A,s} \xi_2$ for $\xi_1, \xi_2 \in T_{I \cup \Delta}$, if there is an attribute $\alpha \in Syn \cup Inh$, occurrences v in $O(s)$ and $u \in O(\xi_1)$, such that $\xi_1/u = \alpha(v)$ and

- either $\alpha \in Syn$, $root(s)[v] = \sigma$, with $\sigma \in (\Sigma \cup \{root^{(1)}\})^{(k)}$, $k \geq 0$ and
$$\xi_2 = \xi_1[u \leftarrow \zeta] \text{ with } \zeta = rhs(\alpha, \pi, \sigma)[\pi \leftarrow v]$$
- or $\alpha \in Inh$ and $v = \bar{v}i$ for some $\bar{v} \in O(s)$, $root(s)[\bar{v}] = \sigma$ with $\sigma \in (\Sigma \cup \{root^{(1)}\})^{(k)}$, $k \geq q$, $i \in [k]$ and
$$\xi_2 = \xi_1[u \leftarrow \zeta] \text{ with } \zeta = rhs(\alpha, \pi i, \sigma)[\pi \leftarrow \bar{v}].$$

□

In the same sense as attribute grammars, attributed tree transducers can be *circular* (see [Fül81] for the notion of circularity for attributed tree transducers). However, in the rest of this paper we always mean <u>noncircular</u> attributed tree transducers when referring to

‡Usually s will be a tree in $\{root(s') \mid s' \in T_\Sigma\}$, but for technical reasons emerging in the proof of Lemma 5.1, we also let it contain other symbols.

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 $root(s)$ is confluent and terminating (cf. Lemma 4.8 and Lemma 4.9 of [FHVV93]). Thus, every sentential form has a unique normal form [New42] and this normal form is in T_Δ. Let us now define the translation realized by an attributed tree transducer.

Definition 3.12 (Translation realized by A)

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

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

The class of translations realized by attributed tree transducers is denoted by AT. Let $k \in I\!N$. The class of translations realized by attributed tree transducers, of which the total number of attributes is less than or equal to k, is denoted by AT_k. $\square$

Note again that the input and/or output alphabet of an attributed tree transducer may be empty and therefore $\{\emptyset\} \in OUT(AT)$. Let us now give a small example of an attributed tree transducer.

Example 3.13 (for an attributed tree transducer)

Let $A_{bin} = (Syn, Inh, \Sigma, \Delta, root, a_0, R)$ be an 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 might use infix notation for the + symbol), $R = R_{root} \cup R_1 \cup R_0 \cup R_e$, where

$$\begin{array}{lcl} R_{root} & = & \{a_0(\pi) \to a(\pi 1),\ b(\pi 1) \to 0\}, \\ R_1 & = & \{a(\pi) \to exp(b(\pi)) + a(\pi 1),\ b(\pi 1) \to s(b(\pi))\}, \\ R_0 & = & \{a(\pi) \to a(\pi 1),\ b(\pi 1) \to s(b(\pi))\}, \\ R_e & = & \{a(\pi) \to 0\}. \end{array}$$

This attributed tree transducer takes a binary number represented as monadic tree over the ranked alphabet Σ as input and produces its decimal value represented as tree over Δ. In Figure 2 we can see what kind of tree traversal over the input tree $s = 1(0(1(e)))$, representing the binary number 101, produces the output tree t which represents the decimal number 5.

Let us also take a look at a derivation by $\Rightarrow_{A_{bin},root(s)}$:

$$\begin{array}{lll} a_0(\varepsilon) & \Rightarrow_{A,root(s)} & a_0(\varepsilon)[\varepsilon \leftarrow \underbrace{\underbrace{rhs(a_0,\pi,root)}_{a(\pi 1)}[\pi \leftarrow \varepsilon]}_{a(1)}] = a(1) \\ & \Rightarrow_{A,root(s)} & a(1)[\varepsilon \leftarrow \underbrace{\underbrace{rhs(a,\pi,1)}_{exp(b(\pi))+a(\pi 1)}[\pi \leftarrow 1]}_{exp(b(1))+a(11)}] = exp(b(1)+a(11)) \\ & \Rightarrow_{A,root(s)} & exp(b(1)+a(11))[1 \leftarrow rhs(b,\pi 1,root)[\pi \leftarrow \varepsilon]] = exp(0)+a(11) \\ & \Rightarrow^*_{A,root(s)} & exp(0)+(exp(s(s(0)))+0) = t \end{array}$$

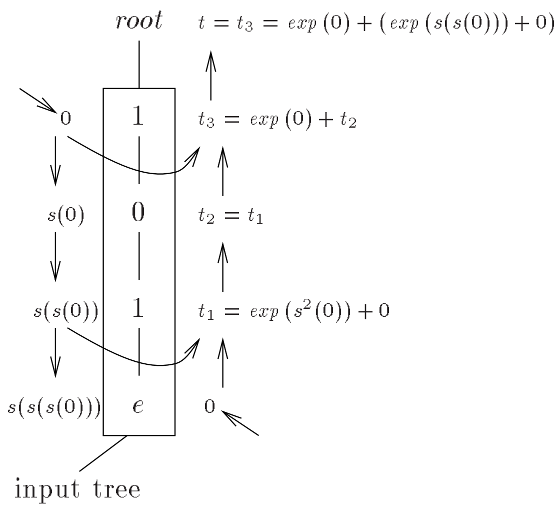


Figure 2: Evaluation of attributes for A_{bin}

Note that in the first step of this calculation, $a_0(\varepsilon)$ is considered as a tree with one node only, and this node is labeled by $a_0(\varepsilon)$. □

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].

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

- $G_0 = (N, T, P, S)$ is a context–free grammar, called the *underlying grammar of* G.
- $D = (V, \Gamma)$ is a semantic domain.
- $B = (Syn, Inh, A, a_0)$, where Syn and Inh are finite sets the elements of which are called *synthesized* and *inherited attributes*, respectively; and $A : N \to \mathcal{P}(Syn \cup Inh \cup \{a_0\})$ is a mapping which associates attributes with the nonterminals of G_0, such that $A(S) = \{a_0\}$ and $a_0 \notin A(X)$ for $X \in N$ with $X \neq S$. The attribute $a_0 \notin Syn \cup Inh$ 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 $p = (X_0 \to X_1 \ldots X_n) \in P$, $R(p)$ 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 $ins(p) = \{\langle b, i\rangle \mid b \in Inh(X_i), i \in [n]\} \cup \{\langle a, 0\rangle \mid a \in Syn(X_0)\}$ is the set of *inside attributes* of p and $outs(p) = \{\langle a, i\rangle \mid a \in Syn(X_i), i \in [n]\} \cup \{\langle b, 0\rangle \mid b \in Inh(X_0)\}$ 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 a node x of t with label $X \in N$. The set of attribute instances at n is the set $\{\langle \alpha, x \rangle \mid \alpha \in A(X)\}$. 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 values are respected by the values of the local attribute instances. The *output language generated by* G, denoted by $out(G)$, is $\{val_{G,t}(\langle a_0, root(t) \rangle) \mid t \text{ is a derivation tree of } G_0\}$.

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., $V = T_\Gamma$. This means that symbols in Γ represent top–concatenation. Thus, for a term–generating attribute grammar G the output language $out(G)$ is in T_Γ. The class of output languages that can be generated by term–generating attribute grammars is denoted by $OUT(AG, TERMS)$. In Example 5.5 we will give a short example for a term–generating attribute grammar.

4 Generating Power of Deterministic Top–Down Tree Transducers

Let us again consider Figure 1. In this section we will establish all results contained in the subtree of the lemma tree, the root of which is labeled by $OUT(dtr\text{–}T) = RECOG$, i.e., the results concerning the generating power of deterministic top–down tree transducers. The main theorem in this section states that the generating power of deterministic, linear top–down tree transducers and deterministic, *total*, linear top–down tree transducers is equal, i.e., $OUT(dl\text{–}T) = OUT(dtl\text{–}T)$.

By definition, the inclusion $OUT(dtl\text{–}T) \subseteq OUT(dl\text{–}T)$ is clear. Now, let us consider an (arbitrary) deterministic, linear top–down tree transducer $M = (Q, \Sigma, \Delta, q_0, R)$. Assume that M is reduced, i.e., M does neither contain superfluous states nor superfluous input symbols and let us assume that there is a $q \in Q$ and a $\sigma \in \Sigma$ such that there is *no* (q, σ)–rule in R. Because of reducedness, this means that there is a derivation by M, which stops with a tree t such that t is irreducible and contains occurrences of the form $q(\sigma(\ldots))$. Thus, in order to obtain a deterministic, total, linear top–down tree transducer M' which has the same output as M, we have to construct the missing (q, σ)–rule for M' to be total. The reason that this is possible, is the following. Because of the reducedness of M, for every state q, there is an input tree s_q, such that $q(s_q) \Rightarrow^*_M t_q$ with $t_q \in T_\Delta$. Thus, whenever there is a missing (q, σ)–rule, we can simply add the rule $q(\sigma(\ldots)) \to t_q$ to R' and we know that this will not change the output language of M' (cf. auxiliary notions of Definition 3.6).

Theorem 4.1 $OUT(dl\text{–}T) = OUT(dtl\text{–}T)$

<u>Proof:</u>

<u>*Case 1*</u>: $OUT(dtl\text{–}T) \subseteq OUT(dl\text{–}T)$.
This is trivially true, because every deterministic, total, linear top–down tree transducer M is a particular deterministic, linear top–down tree transducer.

<u>*Case 2*</u>: $OUT(dl\text{–}T) \subseteq OUT(dtl\text{–}T)$.
Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a deterministic, linear top–down tree transducer. We may assume that R is reduced (cf. Observation 3.8). We now have to construct a deterministic, total, linear top–down tree transducer M', such that $out(M) = out(M')$. Let $M' = (Q, \Sigma, \Delta, q_0, R')$, where R' contains all rules of R and if for a $q \in Q$ and a $\sigma \in \Sigma^{(k)}$, $k \geq 0$ there is no (q, σ)–rule in R, then let the rule

$$q(\sigma(x_1, \ldots, x_k) \to t_q$$

be in R'.

Now we must show that $out(M) = out(M')$. Obviously, $out(M) \subseteq out(M')$ holds, because all rules of R are also in R'. Thus, it remains show that $out(M') \subseteq out(M)$.

It is sufficient to show that for every derivation

$$q_0(s) \Rightarrow_{M',u_1,(p_1,\sigma_1)} \xi_1 \Rightarrow_{M',u_2,(p_2,\sigma_2)} \cdots \Rightarrow_{M',u_n,(p_n,\sigma_n)} \xi_n$$

with $s \in T_\Sigma, \xi_n \in T_\Delta$, $u_1 = \varepsilon$, and for $2 \le i \le n$: $u_i \in O(\xi_{i-1})$, $\xi_{i-1} \in T_{Q \cup \Sigma \cup \Delta}$, and for $i \in [n]$: $p_i \in Q$, there is an $\tilde{s} \in T_\Sigma$ and a derivation $q_0(\tilde{s}) \Rightarrow_M^* \xi_n$. Let us distinguish the following two cases.

Case a: For every $i \in [n]$, there is a (q_i, σ_i)–rule in R. Then there is a derivation

$$q_0(\tilde{s}) \Rightarrow_{M,u_1,(q_1,\sigma_1)} \xi_1 \Rightarrow_{M,u_2,(q_2,\sigma_2)} \cdots \Rightarrow_{M,u_n,(q_n,\sigma_n)} \xi_n$$

by M, where $\tilde{s} = s$.

Case b: There is a $j \in [n]$, such that there is no (q_j, σ_j)–rule in R. Since all rules which were added to R' by Step 2. of the construction have right–hand sides in T_Δ, we may assume that there is a k with $k \in [n]$ such that the derivation by M' is of the form

$$q_0(s) \Rightarrow_{M',u_1,(p_1,\sigma_1)} \xi_1 \Rightarrow_{M',u_2,(p_2,\sigma_2)} \cdots \Rightarrow_{M',u_k,(p_k,\sigma_k)} \xi_k \Rightarrow_{M',u_{k+1},(p_{k+1},\sigma_{k+1})} \cdots \Rightarrow_{M',u_n,(p_n,\sigma_n)} \xi_n \quad (\otimes)$$

and for every $\nu \le k$ there is a (q_ν, σ_ν)–rule in R and for every $j > k$ there is no (q_j, σ_j)–rule in R.

Let $C \subseteq Q \times T_\Sigma$ consist of every pair (q, ζ) such that there is a $u \in O(\xi_k)$ with $\xi_k/u = q(\zeta)$.

Now C contains all information needed to construct $\tilde{s}$ from s. Namely, for every $(q, \zeta) \in C$, we can go back in the derivation $(\otimes)$ to find out at which occurrence u_ζ of s the tree ζ occurs (this is clearly possible, because we could keep the input tree s and make derivations on paths in s, as it is the case for attributed tree transducers). Now we simply have to replace the term ζ at occurrence u_ζ in s by the term s_q.

Note that there are no two terms which are subject of replacement such that one is subterm of the other. This is due to the fact that M' is linear.

Thus, we define

$$\tilde{s} = s[u_\zeta \leftarrow s_q \mid (q, \zeta) \in C].$$

Then there is a derivation

$$q_0(\tilde{s}) \Rightarrow_M^* \tilde{\xi}_k \text{ with } \tilde{\xi}_k = \xi_k[q(\zeta) \leftarrow q(s_q) \mid (q, \zeta) \in C]$$

and by definition of s_q, for every $q \in Q$, there is a derivation $q(s_q) \Rightarrow_M^* t_q$ by M. Thus, there is a derivation $\tilde{\xi}_k \Rightarrow_M^* \tilde{t}$ with $\tilde{t} = \tilde{\xi}_k[q(s_q) \leftarrow t_q \mid (q, \zeta) \in C]$.

We know that $\xi_n = \xi_k[q(\zeta') \leftarrow t_q \mid \zeta' \in T_\Sigma$ and $q(\zeta')$ occurs in $\xi_k]$ and therefore $\tilde{t} = \xi_n$, because the corresponding t_q's are substituted at the same occurrences in ξ_k as in $\tilde{\xi}_k$. For M' this is done by the application of the terminal rules with right–hand sides of the form t_q and for M this is done by derivations on the input trees of the form s_q.

Thus, any tree $\xi_n \in T_\Delta$ which can be derived by M' can also be derived by M, and therefore, $out(M') \subseteq out(M)$. This concludes the proof of Theorem 4.1. □

It is not clear, if the result of Theorem 4.1 can be extended to the nonlinear case, i.e., if $OUT(d\text{–}T) = OUT(dt\text{–}T)$ holds. Let us shortly discuss what difficulties emerge when trying to prove $OUT(d\text{–}T) = OUT(dt\text{–}T)$ by the same techniques as used in the proof of Theorem 4.1. Consider a deterministic top–down tree transducer M and the derivation by M which is depicted in Figure 3. There is a "copy rule" r such that the states q_1 and q_2

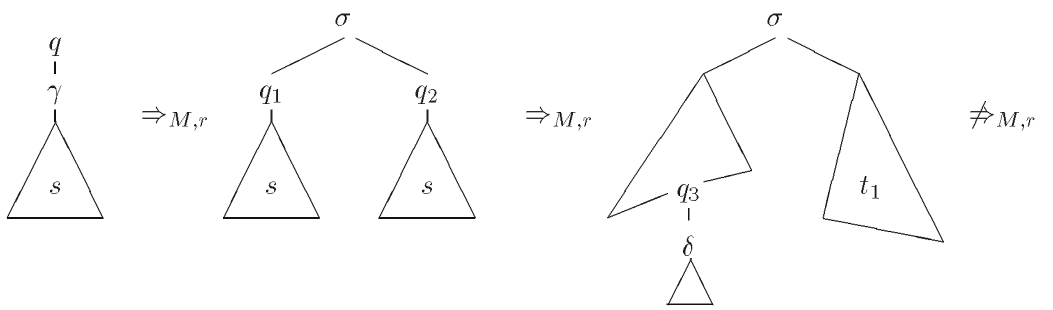


Figure 3: Nonlinear tree transducer M with missing (q_3, σ)–rule

are called on the same input tree s. In the right branch no missing rule is encountered and a terminal tree t_1 is derived. In the left branch, however, there is a $q_3(\sigma(\ldots))$ occurrence and there is no (q_3, σ)–rule. Now, the following problem emerges: we can not assume an arbitrary input tree s_{q_3} for q_3 and add a (q_3, σ)–rule with right–hand side t_{q_3}. Because then, the original input tree s would be changed, too, and thus the tree t_1 would change as well. Thus, it is not possible to simply introduce a terminal rule of the form $q_3(\sigma(\ldots)) \to t_{q_3}$ with $t_{q_3} \in T_\Delta$. Solving the problem by encoding information into the states seems not to be sufficient either. Maybe by changing the input alphabet, a solution could be found.

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 $out(M) = out(M')$. In fact, considering the rules of M', it is even a deterministic, relabeling top–down tree transducer.

Lemma 4.2 *$RECOG \subseteq OUT(dr\text{–}T)$*

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 $M' = (Q, \Gamma, \Sigma, q_0, R')$ be the deterministic, relabeling top–down tree transducer with

$$\Gamma = \{\sigma^{(k)}_{p,p_1,\ldots,p_k} \mid (p(\sigma(x_1, \ldots, x_k)) \to \sigma(p_1(x_1), \ldots, p_k(x_k))) \in R\} \text{ and}$$

$$R' = \{(p(\sigma_{p,p_1,\ldots,p_k}(x_1, \ldots, x_k)) \to \sigma(p_1(x_1), \ldots, p_k(x_k))) \mid (p(\sigma(x_1, \ldots, x_k)) \to \sigma(p_1(x_1), \ldots, p_k(x_k))) \in R\}.$$

It should be clear that $out(M) = out(M')$:

Case 1: $out(M) \subseteq out(M')$

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 $2 \le i \le n$, and for $i \in [n]$: $\xi_i \in T_{Q \cup \Sigma \cup \Delta}$, $r_i = (p_i(\sigma_i(x_1, \ldots x_k)) \to \sigma_i(p_{i,1}(x_1), \ldots, p_{i,k}(x_k))) \in R$ with $\sigma_i \in \Sigma^{(k)}$, $k \ge 0$. 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 $(\sigma_i)_{p_i,p_{i,1},\ldots,p_{i,k}}$ with $k = rank_\Sigma(\sigma_i)$; and therefore $out(M) \subseteq out(M')$.

Case 2: $out(M') \subseteq out(M)$
If there is a derivation $q_0(s) \Rightarrow^*_{M'} t$ by M', with $s \in T_\Gamma$, then there is a derivation $q_0(s') \Rightarrow^*_M t$, where $s' \in T_\Sigma$ is obtained from s by changing every label $\sigma_{p,p_1,\ldots,p_k}$ in s with $k \in I\!N$ to the label σ. Thus, $out(M') \subseteq out(M)$. □

There is a relationship between Lemma 4.2 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 deterministic, relabeling top–down tree transducer constructed for L in the proof of Lemma 4.2 has L_D as domain and realizes such a projection.

Example 4.3 (For the construction of Lemma 4.2)

Let us consider a recognizable tree language L and show that there is a deterministic, relabeling top–down tree transducer M' with $out(M') = L$. Let $L = \{S(a,b), S(b,a)\}$. It is well known that the class of tree languages recognized by deterministic top–down finite tree automata is a proper sub–class of *RECOG*. Indeed, the language L is a recognizable tree language which is *not* in this class (cf. Theorem 3.14 of [Eng75b]). However, the ability to relabel input symbols allows us to construct a *deterministic* relabeling top–down tree transducer M' with $out(M') = L$.

Let $M = (Q, \Sigma, \Sigma, q_0, R)$ be a top–down finite tree automaton with

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{S^{(2)}, a^{(0)}, b^{(0)}\}$
- R consists of the following rules:

$$\begin{array}{lcl} q_0(S(x_1,x_2)) & \to & S(q_1(x_1), q_2(x_2)) \\ q_0(S(x_1,x_2)) & \to & S(q_2(x_1), q_1(x_1)) \\ q_1(a) & \to & a \\ q_2(b) & \to & b. \end{array}$$

Then $dom(M) = out(M) = L$. By Lemma 4.2 there is a deterministic relabeling top–down tree transducer $M' = (Q, \Sigma', \Sigma, q_0, R')$ with

- $\Sigma' = \{S_{q_0,q_1,q_2}, S_{q_0,q_2,q_1}, a_{q_1}, b_{q_2}\}$
- R' consists of the following rules:

$$\begin{array}{llcl} r_1: & q_0(S_{q_0,q_1,q_2}(x_1,x_2)) & \to & S(q_1(x_1), q_2(x_2)) \\ r_2: & q_0(S_{q_0,q_2,q_1}(x_1,x_2)) & \to & S(q_2(x_1), q_1(x_1)) \\ r_3: & q_1(a_{q_1}) & \to & a \\ r_4: & q_2(b_{q_2}) & \to & b. \end{array}$$

Obviously M' is deterministic and $out(M') = L$. □

Consider a deterministic, relabeling top–down tree transducer M. Then we can apply the construction of Theorem 4.1 to get a deterministic, total top–down tree transducer M' with $out(M) = out(M')$. Since only rules with right–hand sides in T_Δ are added by the construction it is clear that M' is a deterministic, total, relabeling top–down tree transducer.

Lemma 4.4 $OUT(dr\text{–}T) = OUT(dtr\text{–}T)$

Proof:

This result follows by the construction of Theorem 4.1 and the facts that relabeling top–down tree transducers are linear and only rules with right–hand sides in T_Δ are added, thus, the conditions for being a restricted, finite–state relabeling are preserved. □

In Example 4.3 we have constructed a deterministic, relabeling top–down tree transducer M' for the (finite) recognizable tree language L. Let us now consider the language L again and construct a deterministic, *total*, relabeling top–down tree transducer M'' the output language of which is L. This is done by applying the construction of Theorem 4.1

Example 4.5 (For the construction of Theorem 4.1)

Let the language L and the deterministic, relabeling top–down tree transducer $M' = (Q, \Sigma', \Sigma, q_0, R')$ be defined as in Example 4.3.

Let us now apply the construction of Theorem 4.1. For every state $q \in Q$, let us first define the trees s_q and t_q.

$$\begin{array}{lcl@{\qquad\qquad}lcl}
s_{q_0} & = & S_{q_0,q_1,q_2}(a_{q_1}, b_{q_2}) & t_{q_0} & = & S(a,b) \\
s_{q_1} & = & a_{q_1} & t_{q_1} & = & a \\
s_{q_2} & = & b_{q_2} & t_{q_2} & = & b
\end{array}$$

Now we simply add the missing rules. Let $M'' = (Q, \Sigma', \Sigma, q_0, R'')$ be a top–down tree transducer, where R'' contains every rule of R' and the following additional rules:

$$\begin{array}{lcll}
q_0(a_{q_1}) & \to & t_{q_0} & (= S(a,b)) \\
q_0(b_{q_2}) & \to & t_{q_0} & \\
q_1(S_{q_0,q_2,q_1}(x_1, x_2)) & \to & t_{q_1} & (= a) \\
q_1(S_{q_0,q_1,q_2}(x_1, x_2)) & \to & t_{q_1} & \\
q_1(b_{q_2}) & \to & t_{q_1} & \\
q_2(S_{q_0,q_2,q_1}(x_1, x_2)) & \to & t_{q_2} & (= b) \\
q_2(S_{q_0,q_1,q_2}(x_1, x_2)) & \to & t_{q_2} & \\
q_2(a_{q_1}) & \to & t_{q_2} &
\end{array}$$

Obviously, M'' is a deterministic, total, relabeling top–down tree transducer which produces the output $out(M'') = \{S(a,b), S(b,a)\} = out(M') = L$. □

Now, from Lemma 4.2 and Lemma 4.4 and the fact that every deterministic, total, relabeling top–down tree transducer is also a deterministic, total top–down tree transducer, the following corollary follows immediately.

Corollary 4.6 $RECOG \subseteq OUT(dtr\text{–}T) \subseteq OUT(dt\text{–}T)$

Let us consider linear top–down tree transducers now and state a lemma which is a consequence of the fact that $RECOG$ is closed under linear tree transductions.

Lemma 4.7 $OUT(l\text{–}T) \subseteq RECOG$

Proof:

Recognizable tree languages are closed under linear top–down tree transformations [Tha69] and the domain of a top–down tree transformation is recognizable (see Theorem 1 of Section II of [Rou70]). Thus, $OUT(l\text{–}T) \subseteq RECOG$. □

Now, the following corollary follows immediately.

Corollary 4.8 $OUT(l\text{–}T) = OUT(tl\text{–}T) = OUT(dtl\text{–}T) = OUT(dtr\text{–}T) = RECOG$

Proof:

By definition, the following sequence of inclusions hold: $OUT(dtr\text{–}T) \subseteq OUT(dtl\text{–}T) \subseteq OUT(tl\text{–}T) \subseteq OUT(l\text{–}T)$ and therefore:

$$\begin{array}{rcll} RECOG & = & OUT(dtr\text{–}T) & \text{(Corollary 4.6)} \\ & \subseteq & OUT(dtl\text{–}T) & \\ & \subseteq & OUT(tl\text{–}T) & \\ & \subseteq & OUT(l\text{–}T) & \\ & \subseteq & RECOG & \text{[Tha69].} \end{array}$$

□

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., $OUT(AG, TERMS) = OUT(AT)$. The lemma tree in Figure 1 shows the outline of our proof for this equality. Let us first consider the composition result $dtr\text{–}T \circ AT \subseteq AT$. This result is not surprising, because the restriction of being a deterministic, total, 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. In the context of top–down tree transducers the ssur property then means that for an input symbol σ, a tree $q(x_i)$ for a particular i may only occur once in the set of σ–rules. This is obviously true for deterministic, total, relabeling top–down tree transducers, because for every input symbol σ there is at most *one* rule of the form $q(\sigma(x_1, \ldots, x_k)) \rightarrow \delta(q_1(x_1), \ldots, q_k(x_k))$ and all other σ–rules have right–hand sides in T_Δ (where Δ is the output alphabet).

In [Gan83] it is proved that the classes of translations induced 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 induced by attribute coupled grammars is included in AC, i.e., ssur–AC $\circ$ AC $\subseteq$ AC. Thus, the composition result for deterministic, total, relabeling top–down tree transducers and attributed tree transducers follows in a more or less straightforward manner from the result by Ganzinger. However, since the class $dtr\text{–}T$ 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 a deterministic, total, relabeling top–down tree transducer 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 rules of the deterministic, total, relabeling top–down tree transducer 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 $q(\sigma(x_1, \ldots, x_k)) \rightarrow \delta(q_1(x_1), \ldots, q_k(x_k))$ and therefore, for every inside inherited attribute occurrence $\beta(\pi i)$ there can only be one particular state q_i when translating a right–hand side of M for the symbol σ.

Note that the syntactic single used restriction of [Gan83, Gie88] holds for M, because for a $\sigma \in \Sigma^{(k)}$, $k \geq 1$ there can not be two rules of M, such that $q(x_i)$ occurs in their right–hand sides for an $i \in [k]$.

Lemma 5.1 *$dtr\text{–}T \circ AT \subseteq AT$*

Proof:

Let $M = (Q, \Sigma, \Gamma, q_0, P)$ be a deterministic, total, relabeling top–down tree transducer and let $A = (\mathit{Syn}, \mathit{Inh}, \Gamma, \Delta, \mathit{root}, a_0, R)$ be an attributed tree transducer. Let $A' = (\mathit{Syn}', \mathit{Inh}, \Sigma, \Delta, \mathit{root}, (q_0, a_0), R')$ be the attributed tree transducer with $\mathit{Syn}' = Q \times \mathit{Syn}$ and R' consists of the following rules:

Let $\sigma \in \Sigma^{(k)}$ and let $q(\sigma(x_1, \ldots, x_k)) \to \xi$ be a rule in P.

(1) Let $\xi = \delta(q_1(x_1), \ldots, q_k(x_k))$ for some $\delta \in \Delta$ and $q_1, \ldots, q_k \in Q$. Then

- for every synthesized attribute $a \in \mathit{Syn}$, the rule $(q, a)(\pi) \to \xi'$ is in R'_σ, where

  $$\xi' = \mathit{nf}(\Rightarrow_{A,\xi}, a(\varepsilon))[\alpha(i) \leftarrow (q_i, \alpha)(\pi i) \mid \alpha \in \mathit{Syn}, i \in [k]]\,[\beta(\varepsilon) \leftarrow \beta(\pi) \mid \beta \in \mathit{Inh}].$$

  Note that, by Definition 3.11, $\Rightarrow_{A,\xi}$ is defined by taking $\Sigma' = \Gamma \cup \{q(x_i) \mid q \in Q, i \in [k]\}$.

- for every inherited attribute $b \in \mathit{Inh}$ and every $j \in [k]$, the rule $b(\pi j) \to \xi'$ is in R'_σ, where

  $$\xi' = \mathit{nf}(\Rightarrow_{A,\xi}, b(j))[\alpha(i) \leftarrow (q_i, \alpha)(\pi i) \mid \alpha \in \mathit{Syn}, i \in [k]]\,[\beta(\varepsilon) \leftarrow \beta(\pi) \mid \beta \in \mathit{Inh}].$$

(2) Let $\xi \in T_\Delta$. Then

- for every synthesized attribute $a \in \mathit{Syn}$, the rule $(q, a)(\pi) \to \mathit{nf}(\Rightarrow_{A,\xi}, a(\varepsilon))$ is in R'_σ and

- for every inherited attribute $b \in \mathit{Inh}$ and every $j \in [k]$, the rule $b(\pi j) \to \mathit{nf}(\Rightarrow_{A,\xi}, b(j))$ is in R'_σ.

And let the set R'_{root} be defined as follows.

- The rule $(q_0, a_0)(\pi) \to \xi$ is in R'_{root}, where

  $$\xi = \mathit{rhs}(a_0, \pi, \mathit{root})[\alpha(\pi 1) \leftarrow (q_0, \alpha)(\pi 1) \mid \alpha \in \mathit{Syn}]$$

- and for every $\beta \in \mathit{Inh}$, the rule $(q_0, \beta)(\pi 1) \to \xi$ is in R'_{root}, where

  $$\xi = \mathit{rhs}(\beta, \pi 1, \mathit{root})[\alpha(\pi 1) \leftarrow (q_0, \alpha)(\pi 1) \mid \alpha \in \mathit{Syn}].$$

Now, for every $t \in T_\Sigma$, it holds that $\tau_A(\tau_M(t)) = \tau_{A'}(t)$. □

Now we have all the tools to prove that the result $T(\mathit{RECOG}) = \mathit{OUT}(\mathit{RECOG})$ 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. Let us first prove the result of $\mathit{AT}(\mathit{RECOG}) \subseteq \mathit{OUT}(\mathit{AT})$; the equality follows then immediately as a corollary.

Lemma 5.2 $AT(RECOG) \subseteq OUT(AT)$

Proof:

Let L be a recognizable tree language and let A be an attributed tree transducer. Then, by Corollary 4.8 there is a deterministic, total, relabeling top–down tree transducer M with $out(M) = L$. Then, $\tau_A(L) = \tau_A(out(M)) = out(\tau_M \circ \tau_A)$. By Lemma 5.1 there is an attributed tree transducer A' with $\tau_{A'} = \tau_M \circ \tau_A$. Thus $AT(RECOG) \subseteq OUT(AT)$.
□

Corollary 5.3 $AT(RECOG) = OUT(AT)$

Proof:

The $\subseteq$–direction follows from Lemma 5.2. The other direction 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 T_Δ.

Lemma 5.4 $OUT(AT) \subseteq OUT(AG, TERMS)$

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 $N = \Sigma \cup \{root\}$, $T = \emptyset$, $S = root$, $P = \bigcup_{X \in N} P_X$, where $P_X = \{X \to X_1 \cdots X_k \mid k = rank_{\Sigma'}(X), X_1, \ldots, X_k \in \Sigma\}$, where $\Sigma' = \Sigma \cup \{root^{(1)}\}$
- $D = (T_\Delta, \Delta)$
- $B = (Syn', Inh', A, a_0)$ with $Syn' = Syn$, $Inh' = Inh$, $A(X) = Syn \cup Inh'$ for all $X \in \Sigma$, and $A(root) = \{a_0\}$
- $R' = (R'(p) \mid p \in P)$, where for all $X \in N$ and $p \in P_X$: $R'(p) = \{(\varphi_{r'}(l) = \varphi_{r'}(r)) \mid r' = (l \to r) \in R_X\}$, where $\varphi_{r'}$ is the substitution
$$[a(\pi p) \leftarrow \langle a, p\rangle \mid a \in Syn \cup Inh \cup \{a_0\}, a(\pi p) \text{ occurs in } r'] \, [\varepsilon \leftarrow 0].$$

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 that 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 5.5 (for the construction in Lemma 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 $N = \Sigma \cup \{root\} = \{root, 1, 0, e\}$, $T = \emptyset$, $S = root$, $P = P_{root} \cup P_1 \cup P_0 \cup P_e$ with $P_{root} = \{root \rightarrow 1, root \rightarrow 0, root \rightarrow e\}$, $P_1 = \{1 \rightarrow 1, 1 \rightarrow 0, 1 \rightarrow e\}$, $P_0 = \{0 \rightarrow 1, 0 \rightarrow 0, 0 \rightarrow e\}$, $P_e = \{e \rightarrow \varepsilon\}$.
- $D = (T_\Delta, \Delta)$
- $B = (Syn', Inh', A, a_0)$ with $Syn' = Syn = \{a\}$, $Inh' = Inh = \{b\}$, $A(1) = A(0) = A(e) = Syn' \cup Inh' = \{a, b\}$ and $A(root) = \{a_0\}$.
- $R' = (R'(p) \mid p \in P)$, where
  for $p \in P_{root}$: $R'(p) = \{\langle a_0, 0\rangle = \langle a, 1\rangle, \langle b, 1\rangle = 0\}$,
  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 Figure 2. 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}. □

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

Theorem 5.6 $OUT(AG, TERMS) = OUT(AT)$

Proof:

Let G be an attribute grammar with underlying context–free grammar G_0. The set $DT(G_0)$ of derivation trees of G_0, viewed as trees over a ranked alphabet, is a recognizable tree language [MW67]. Define an attributed tree transducer A, which has the same rules as G (via a similar substitution as φ in the construction of R' in the proof of Lemma 5.4) and add dummy rules for undefined rules. Clearly, $out(G) = \tau_A(DT(G_0))$ and thus $OUT(AG, TERMS) \subseteq AT(RECOG)$. Now we have:

$$\begin{array}{rcll} OUT(AG, TERMS) & \subseteq & AT(RECOG) & \\ & = & OUT(AT) & \text{(by Corollary 5.3)} \\ & \subseteq & OUT(AG, TERMS) & \text{(by Lemma 5.4)} \end{array}$$

□

6 Applications

In this paper we have shown that attribute grammars and attributed tree transducers have the same term–generating power. Let us now give applications of this result.

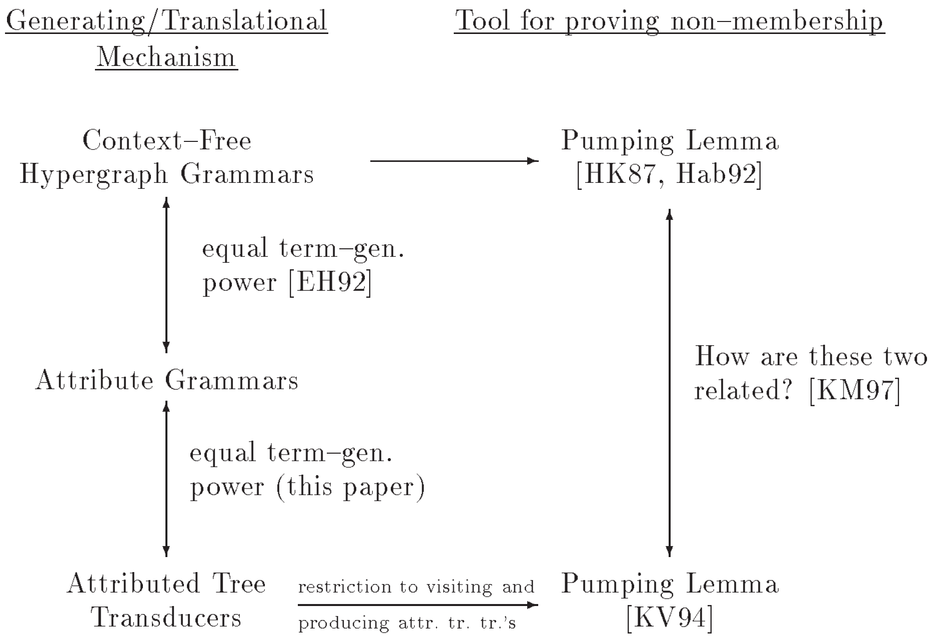


Figure 4: Three mechanisms with equal term–generating power

All three mechanisms listed on the left–hand side of Figure 4 have the same term–generating power. On the right–hand side we show some tools that sometimes can be used to prove that a particular language is *not* in the given class of languages. The first one is the pumping lemma by Habel and Kreowski for context–free hypergraph languages and the second one is the pumping lemma by Kühnemann and Vogler for output languages of restricted attributed tree transducers. The latter is defined for the class of output languages of so called visiting and producing 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 occurrence of x is needed to compute the value of the initial synthesized attribute occurrence 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.

Let us now give an example of an application of our results. In Theorem 4.3 of [KV94] it is shown that the language $\{\gamma^{2^n}\alpha \mid n \geq 0\}$ is not in the class of output languages of visiting and producing attributed tree transducers. However due to the limitation of their pumping lemma to this class, it is *not* proved that this language is not in the class $OUT(AT)$ (nor has this been proved anywhere else). With the help of the pumping lemma for context–free hypergraph languages we can now give such a proof which is much more elegant than the one presented in [KV94].

Theorem 6.1 $L_{exp} = \{\gamma^{2^n}\alpha \mid n \geq 0\} \notin OUT(AT)$

Proof:

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 5.4 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 cfhg G' such that the tree language generated by G' is equal to L_{exp}. Now by 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' can not exist and therefore the tree transducer A can not exist either. This concludes the proof. □

Let us now turn to our second application. We want to prove a hierarchy of output languages of attributed tree transducers w.r.t. their number of attributes. We do this by first proving a hierarchy of tree languages generated by cfhg's w.r.t. their rank.

In the following, let $\Sigma = \{a_1^{(1)}, \ldots, a_k^{(1)}, e^{(0)}\}$ and let $L_k = \{a_1^n(a_2^n(\ldots a_k^n(e))) \mid n \geq 0\} \subseteq T_\Sigma$. Recall the definition of $\mathcal{TREE}_k$ from Subsection 2.4.

Lemma 6.2 For every $k \in I\!N$, $L_k \in \mathcal{TREE}_k$.

Proof:

For every $k \in I\!N$ we construct a term–generating cfhg $G_k = (\Sigma_k, \Delta_k, S, P)$, such that $TREE(G_k) = L_k$, where $\Sigma_k = \{S^{(1)}, A^{(k)}, a_1^{(2)}, \ldots, a_k^{(2)}, e^{(1)}\}$, $\Delta_k = \{a_1^{(2)}, \ldots, a_k^{(2)}, e^{(1)}\}$. For $k = 2m$, $m \in I\!N$, i.e., k is even, $P = \{p_1, p_2, p_3\}$, where the rules p_1, p_2, p_3 are shown in Figure 5. For $k = 2m + 1$, $m \in I\!N$, i.e., k is odd, $P = \{p'_1, p'_2, p'_3\}$, where the rules p'_1, p'_2, p'_3 are shown in Figure 6.

Results of derivations by G_{2m} and G_{2m+1} are shown in Figure 7. 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 $m \in I\!N$. 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 $k \geq 1$, $L_k \notin \mathcal{TREE}_{k-1}$.

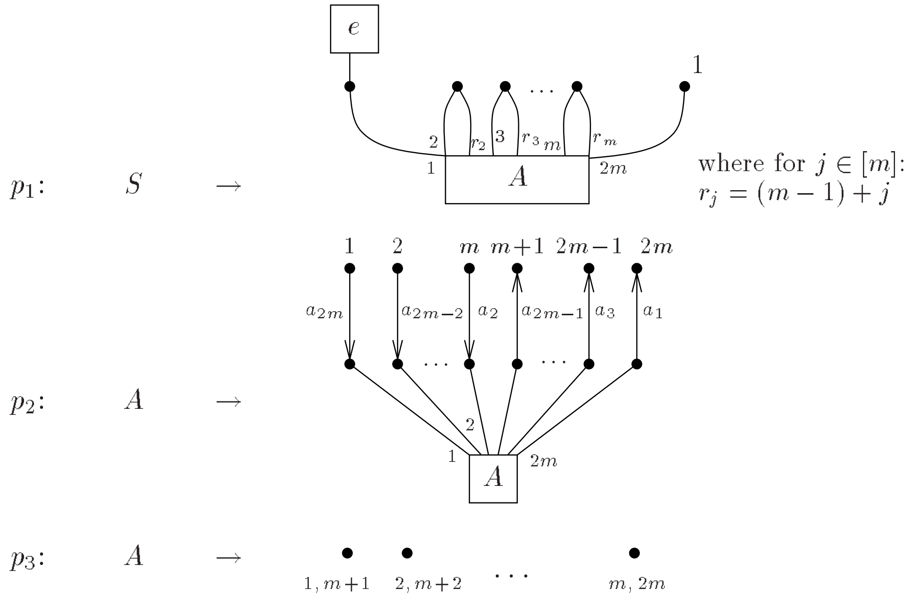


Figure 5: Rules for the grammars G_k for even k's

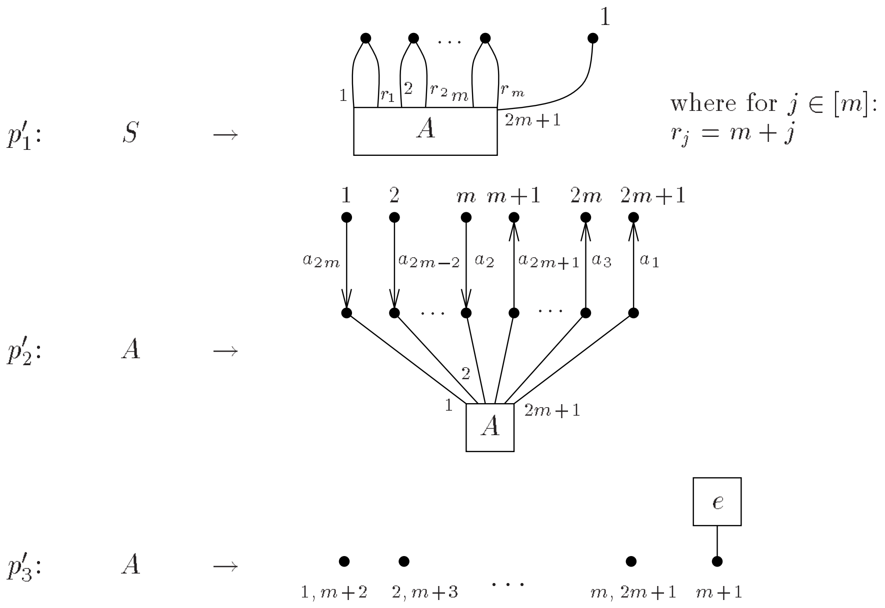


Figure 6: Rules for the grammars G_k for odd k's

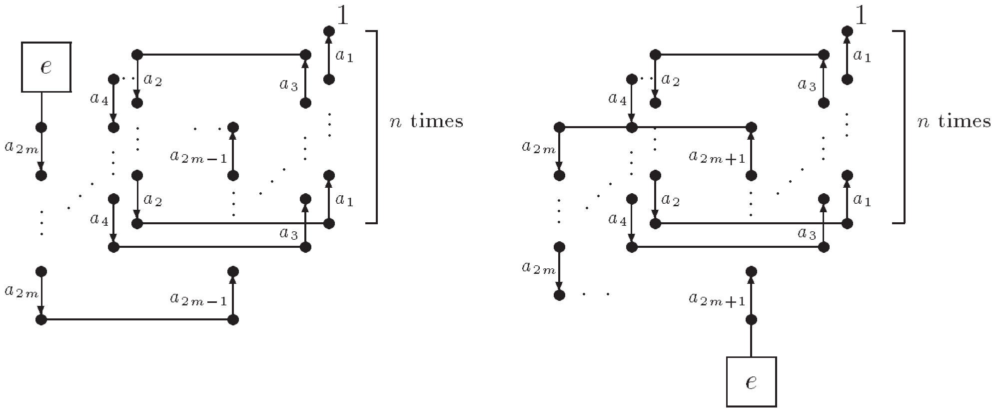


Figure 7: Results of derivations of G_{2m} and G_{2m+1}

Proof:

Recall the pumping lemma for context–free hypergraph languages as stated in Theorem 2.3 of [Hab92]. Now, assume that $L_k \in \mathcal{TREE}_{k-1}$ for a $k \geq 1$. 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* must contain (at least) k hyperedges labeled by $a_1, \ldots, a_k$. This is true, because otherwise pumping would lead to a hypergraph with unequal numbers of occurrences of hyperedges labeled by $a_1, \ldots, a_k$. Also, the hyperedges labeled by $a_1, \ldots, a_k$ must be incident with distinct external nodes, i.e., there must be at least k external nodes. This is true because otherwise the resulting tree would not be monadic (or the numbers of occurrences of hyperedges labeled by $a_1, \ldots, a_k$ would not be equal). However, by assumption, L_k is generated by a cfhg of rank $k-1$, i.e., there may be at most $k-1$ external nodes. This contradicts our assumption and concludes the proof. □

Now, by Lemma 6.2 and Lemma 6.3, the following theorem follows.

Theorem 6.4 For every $k \in I\!N$, $\mathcal{TREE}_k \subset \mathcal{TREE}_{k+1}$.

Note that this hierarchy of tree languages generated by cfhg's w.r.t. their rank is in contrast to the hierarchy of string languages generated by cfhg's w.r.t. their rank (cf. Theorem 4.5 of Chapter V of [Hab92]), which increases only every second step. This is due to the definition of string–(hyper)graphs. That is, string–graphs have, by definition, two external nodes. Thus, they are different from the tree–generating hypergraphs, representing the monadic trees of the languages L_k. This difference is comparable to the difference between the monoid of words on an alphabet A 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 $\{a_1^n a_2^n \ldots a_k^n \mid n \in I\!N\}$, is very similar to the grammars G_k constructed in Lemma 6.2. The extra external node, which is needed by definition of string–graphs, is needed exclusively to "hold on" to the end of the string–graphs.

We are now ready to prove our second result, i.e., a hierarchy for the output languages of attributed tree transducers w.r.t. their number of attributes.

Corollary 6.5 For every $k \in I\!N$, $OUT(AT_k) \subset OUT(AT_{k+1})$.

Proof:

(1) $L_k \in OUT(AT_k)$ for every $k \in I\!N$: This can be proved analogous to the proof of Lemma 6.2. Let $k \in I\!N$ and let G_k be the cfhg as defined in the proof of Lemma 6.2. By the construction in the proof of Lemma 5.1 of [EH92] there is an attribute grammar G'_k, with k attributes, which generates the tree language $TREE(G_k)$ as output. Let $G_{0,k}$ be the underlying context–free grammar of G'_k. Now, take the alphabet $\Sigma = \{A^{(1)}_{p_2}, A^{(0)}_{p_3}\}$ as input alphabet for an attributed tree transducer A_k. Then, the set of derivation trees $DT(G_{0,k})$ of the grammar $G_{0,k}$ corresponds to the set of control trees $\{root(s) \mid s \in T_\Sigma\}$ of the attributed tree transducer A_k (more precisely, every derivation tree in $DT(G_{0,k})$ is a projection of such a control tree). The rules for $R_{root}, R_{A_{p_2}}$, and $R_{A_{p_3}}$ are taken in correspondence with the semantic rules of G'_k, i.e., in correspondence with the rules p_1, p_2, p_3 and p'_1, p'_2, p'_3, respectively. Then, for every $k \in I\!N$, $out(A_k) = L_k$.

(2) $L_k \notin OUT(AT_{k-1})$ for every $k \geq 1$: Assume that $L_k \in OUT(AT_{k-1})$. Then by Lemma 5.4 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 5.4 preserves the number of attributes of the attributed tree transducer. Then, by the construction of Lemma 4.1 of [EH92] 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. $\square$

7 Further Research Topics

In [KM97] we want to investigate how the two pumping lemmata, the one for context–free hypergraph languages and the one for output languages of (restricted) attributed tree transducers 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.

Note that the technique used in Section 6 to prove a hierarchy for the output languages of attributed tree transducer w.r.t. to their number of attributes does not suffice to prove a hierarchy result which distinguishes between synthesized and inherited attributes. For visiting and producing attributed tree transducers, such a hierarchy result is given in Theorem 4.7 of [KV94]. It seems that $L_{2m} \notin OUT(AT_{m,m-1})$ for $m \geq 1$ and $L_{2m+1} \notin OUT(AT_{m,m})$, for $m \in I\!N$, where $AT_{i,j}$ denotes the class of translations realized by attributed tree transducers with at most i synthesized attributes and at most j inherited attributes. However, this hierarchy is not implied by Corollary 6.5 (cf. Figure 8), because

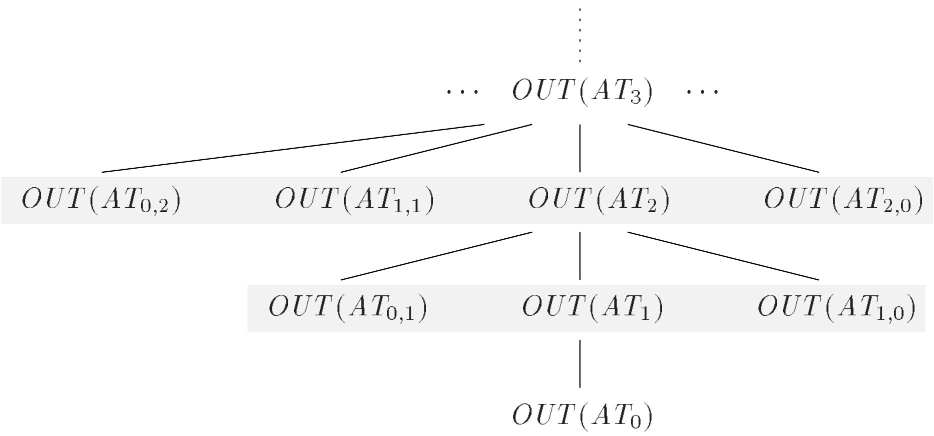


Figure 8: Consequences of Corollary 6.5

there is no distinction between the different classes of output languages of attribute tree transducers with equal (total) number of attributes (viz., the classes depicted in the grey boxes in Figure 8). This hierarchy can not be proved using the pumping lemma for context–free hypergraph languages and is subject to further research. Maybe it could be proved by a more “fine–grained” pumping lemma for context–free hypergraph languages, which distinguishes between nodes corresponding to synthesized and inherited attributes, respectively.

Acknowledgment

I would like to thank Zoltan Fülöp, Armin Kühnemann and Heiko Vogler for fruitful discussions on the contents of this paper.

References

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

[DJL88] P. Deransart, M. Jourdan, and B. Lorho. *Attribute grammars, definitions and bibliography.* Number 323 in LNCS. Springer–Verlag, 1988.

[DKH96] F. Drewes, H.-J. Kreowski, and A. Habel. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Transformations, Volume I: Foundations.* World Scientific, 1996. (to appear).

[Ech90] R. Echahed. On completeness of narrowing strategies. *Theoret. Comput. Sci.*, 72:133–146, 1990.

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

[Eng75a] J. Engelfriet. Bottom–up and top–down tree transformations — a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[Eng75b] J. Engelfriet. Tree automata and tree grammars. Lecture Notes, DAIMI FN-10, University of Aarhus, 1975.

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

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

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

[Eng96] J. Engelfriet. Context–free graph grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3.* Springer-Verlag, 1996. (to appear).

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

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

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

[Gan83] H. Ganzinger. Increasing modularity and language–independency in automatically generated compilers. *Science of Computer Programming*, 3:223–278, 1983.

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

[GR62] S. Ginsburg and H. G. Rice. Two families of languages related to ALGOL. *J. Assoc. Comput. Mach*, 9(3):350–371, 1962.

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

[Hab92] A. Habel. *Hyperedge–replacement: Grammars and Languages*, volume 643 of *LNCS*. Springer–Verlag, 1992.

[HK87] A. Habel and H.-J. Kreowski. Some structural aspects of hypergraph languages generated by hyperedge replacement. In F. Brandenburg, editor, *Proceedings of the 4th annual symposium on theoretical aspects of computer science, STACS 87*, number 247 in LNCS, pages 207–219. Springer–Verlag, 1987.

[KM97] A. Kühnemann and S. Maneth. Relations and applications of pumping lemmata for attributed tree transducers and hyperedge–replacement grammars. (in preparation), 1997.

[Knu68] D.E. Knuth. Semantics of context–free languages. *Math. Systems Theory*, 2:127–145, 1968.

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

[MW67] J. Mezei and J. B. Wright. Algebraic automata and context–free sets. *Information and Control*, 11:3–29, 1967.

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

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

[Tha69] J.W. Thatcher. Generalized^2 sequential machine maps. IBM Res. Report RC 2466, 1969.

[Tha70] J.W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. Syst. Sci.*, 4:339–367, 1970.

[You89] J.H. You. Enumerating outer narrowing derivations for constructor–based term rewriting systems. *J. Symbolic Computation*, 7:319–341, 1989.