Synthesized and inherited functions – a new computational model for syntax–directed semantics

(revised version, June 11, 1993)

Armin Kühnemann and Heiko Vogler

Abt. Theoretische Informatik, Universität Ulm

Oberer Eselsberg, D–89069 Ulm, Germany

e–mail: {kuehne, vogler} @ informatik.uni–ulm.de

Abstract

In this paper we introduce a new formal model for the concept of syntax–directed semantics, called *macro attributed tree transducer* (for short: mat tree transducer). This model is based on (noncircular) attributed tree transducers and on macro tree transducers. In the first type of transducer, semantic values are computed by means of meaning names called synthesized attributes, and by means of context names called inherited attributes. Both, synthesized and inherited attributes represent basic semantic values. In the second type of transducer, semantic values are computed by meaning names only which are called states. However, in order to have a means of handling context information, states represent functions over semantic values. The new model integrates attributed tree transducers and macro tree transducers by allowing both, meaning names and context names to represent functions over semantic values. In analogy to the terminology of attributed tree transducers, we call such meaning names and context names also synthesized functions and inherited functions, respectively.

We present an inductive characterization of the tree transformation computed by an mat tree transducer. We prove that mat tree transducers are more powerful than both, attributed tree transducers and macro tree transducers. We characterize mat tree transducers by the two–fold composition of attributed tree transducers. This characterization has three consequences: (1) the height of output trees of mat tree transducers is bounded exponentially in the size of the input tree, (2) the composition hierarchy of mat tree transducers is strict, and (3) mat tree transducers are closed under right–composition with top–down tree transducers, but not under left–composition. Moreover, we prove that the addition of inherited attributes does not increase the computational power of macro tree transducers.

1 Introduction

The concept of syntax–directed semantics is a well–accepted principle to describe the meaning of tree–structured objects. Since such objects will control the computation of the meanings, we will call them *control trees*. The concept of syntax–directed semantics was first formalized by E.T. Irons [Iro61] in 1961. In the meantime, many other formalizations of the concept have been studied, e.g., attribute grammars [Knu68], top–down tree transducers [Rou70, Tha70], generalized syntax–directed translation schemes [AU71, AU73], denotational semantics [SS71, Gor79], affix grammars [Kos71], macro tree transducers [Eng80, CF82, EV85], attributed tree transducers [Fül81], attribute coupled grammars [Gie88], context–free hypergraph–based syntax–directed translation schemes [EH91], and top–down tree–to-graph transducers [EV92]. A detailled study of the concept of syntax–directed semantics is given in [Eng81] where special attention is paid to the description and comparison of attribute grammars and macro tree transducers.

In most of the models which are mentioned, handling of context is possible. More precisely, if s is a given control tree and n is a node of s, then the models allow to express that the semantic values of different meaning names at n may depend on information about the context of the subtree $sub(s,n)$ which starts at n (cf. Figure 1).

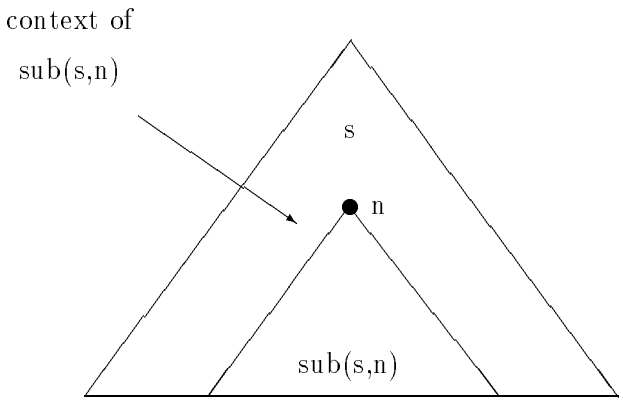

Figure 1: Control tree s, subtree $sub(s,n)$, and its context.

In particular this ability holds for attributed tree transducers and macro tree transducers. However, the context handling is formalized differently:

1. In attributed tree transducers, context names are explicitely associated with n; they carry the semantic values of the context information. Here, meaning names and context names are called synthesized attributes and inherited attributes, respectively. Both types of attributes represent basic semantic values.

2. In macro tree transducers, context information does not appear explicitely. Rather the meaning names of a node n are formalized as functions which map semantic values of context information to the desired semantic values.

In this paper we introduce a new formalization of the concept of syntax–directed semantics, called *macro attributed tree transducer* (for short: mat tree transducer) which integrates the context handling mechanisms of (noncircular) attributed tree transducers and of macro tree transducers. In mat tree transducers, meaning names and context names are available, and both types of names represent functions over basic semantic values. In analogy to attributed tree transducers, we will call meaning names and context names *synthesized functions* and *inherited functions*, respectively.

Since we will compare the expressive power of various subclasses of mat tree transducers, we have to use, for every partner of the comparison, the same semantic domain. Here we fix the free term algebra for the formalizations presented in this paper, i.e., we study the initial algebra semantics of the formal models (cf. [GTWW77] for the concept of initial algebra semantics). On the one hand the free term algebra is a special domain, but on the other hand it is also a general domain in the following sense: every concrete semantics can be obtained from the initial semantics by applying the unique initial homomorphism to the semantic value in the free term algebra. Having this particular semantic domain, it is clear that we study formal models which compute *tree functions*; a tree function takes trees as arguments and delivers trees as result. A tree function of arity one is called *tree transformation*.

Rather than trying to present here in the introduction the general definition of mat tree transducers in an informal style, we illustrate the new model by means of an example. Let us consider the following simple tree transformation for which we will later construct an mat tree transducer:

> Consider the ranked alphabet $\Sigma = \{\sigma, \alpha\}$ where σ and α are symbols of rank 2 and 0, respectively. The tree transformation τ takes trees over Σ as arguments and, for the argument s, it delivers the tree s' which is obtained from s by replacing every leaf node n by the tree–encoding of the path from the root of s to n (cf. Figure 2).

For a control tree s over Σ, the encoding of a path through s is a string over the alphabet $p\Sigma = \{(\sigma,1),(\sigma,2),(\alpha,0)\}$, e.g., for the tree

$$s = \sigma(\alpha, \sigma(\alpha, \alpha)),$$

the encoding of the path leading from the root to the second leaf which is labelled by α, is the string

$$(\sigma,2)(\sigma,1)(\alpha,0).$$

The tree–encoding of a path is the encoding of the path viewed as a tree over the ranked alphabet $pt\Sigma = \{(\sigma,1),(\sigma,2),(\alpha,0)\}$ where (σ,i) has rank 1 and $(\alpha,0)$ has rank 0. Thus,

$$\tau(\sigma(\alpha,\sigma(\alpha,\alpha))) = \sigma((\sigma,1)(\alpha,0),\sigma((\sigma,2)(\sigma,1)(\alpha,0),(\sigma,2)(\sigma,2)(\alpha,0)))$$

where we have dropped the parantheses for the 'path part' of the output tree.

For the tree transformation τ, we construct an mat tree transducer M_τ which computes τ. Before showing its set of (rewrite) rules, we first have to point out how the values of

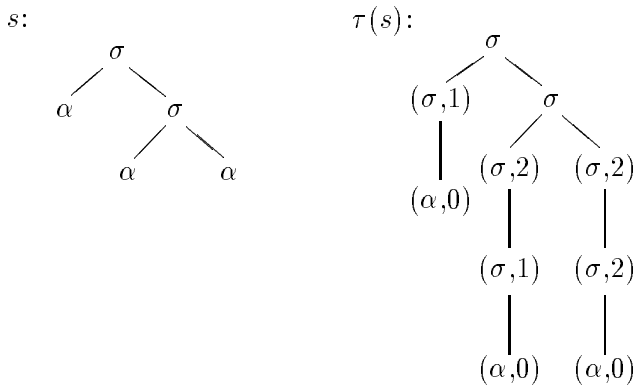

Figure 2: Tree–encoding of the paths from the root of s to the leaves.

inherited functions at the root rt_s of the control tree s are computed. For the computation of values of functions, we intermediately consider the tree $root(s)$ where $root$ is a new symbol of rank 1 which occurs nowhere else in the tree. Then the semantic value of an inherited function b at the root rt_s of s will be described by an ordinary semantic rule for b at the first son of the $root$–labelled node. However, the right–hand sides of these rules may not refer to inherited functions at the father of rt_s. The values of synthesized functions at rt_s can be used to compute both, the values of inherited functions at rt_s and of synthesized functions at the root of the tree $root(s)$ (cf. Figure 3).

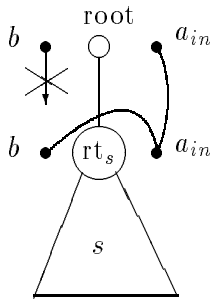

Figure 3: Semantic rules for root.

We note that the tree transformation which is computed by an mat tree transducer, receives s as argument and not $root(s)$. Now we list the rules of our mat tree transducer M_τ; $down$ is a synthesized function with arity one, and up is an inherited function with arity two. We group the rules of M_τ into sets R_{root}, R_σ, and R_α of rules for the symbols

root, σ and α:

$$
\begin{array}{lllll}
R_{root}: & (1) & down(z) & \to & down(z1) \\
 & (2) & up(z1, y_1) & \to & y_1 \\
R_\sigma: & (3) & down(z) & \to & \sigma(down(z1), down(z2)) \\
 & (4) & up(z1, y_1) & \to & up(z, (\sigma, 1)(y_1)) \\
 & (5) & up(z2, y_1) & \to & up(z, (\sigma, 2)(y_1)) \\
R_\alpha: & (6) & down(z) & \to & up(z, (\alpha, 0))
\end{array}
$$

Here, e.g., rule (4) describes the value of the inherited function *up* at every node which is the first son of a σ–labelled node. For every control tree s over Σ, the two types of rewrite variables have the following meaning. We use

- the *path variable* z to represent paths through $root(s)$
- the *parameters* y_j to represent the other arguments of a synthesized or inherited function.

For every control tree s, the set of rules induce a derivation relation $\Rightarrow_{root(s)}$. For the control tree $root(s) = root(\sigma(\alpha, \sigma(\alpha, \alpha)))$, we show the derivation and we discuss in more detail the derivation step which is marked by $(*)$. We expose the applied rule as superscript of the derivation relation:

$$
\begin{array}{ll}
 & down(\varepsilon) \\
 & \Rightarrow^{(1)}_{root(s)} \quad down(1) \\
 & \Rightarrow^{(3)}_{root(s)} \quad \sigma(down(11), down(12)) \\
 & \Rightarrow^{(6)}_{root(s)} \quad \sigma(up(11, (\alpha, 0)), down(12)) \\
 & \Rightarrow^{(4)}_{root(s)} \quad \sigma(up(1, (\sigma, 1)(\alpha, 0)), down(12)) \\
 & \Rightarrow^{(2)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), down(12)) \\
 & \Rightarrow^{(3)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma(down(121), down(122))) \\
 & \Rightarrow^{(6)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma(up(121, (\alpha, 0)), down(122))) \\
(*) & \Rightarrow^{(4)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma(up(12, (\sigma, 1)(\alpha, 0)), down(122))) \\
 & \Rightarrow^{(5)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma(up(1, (\sigma, 2)(\sigma, 1)(\alpha, 0)), down(122))) \\
 & \Rightarrow^{(2)}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma((\sigma, 2)(\sigma, 1)(\alpha, 0), down(122))) \\
 & \Rightarrow^{*}_{root(s)} \quad \sigma((\sigma, 1)(\alpha, 0), \sigma((\sigma, 2)(\sigma, 1)(\alpha, 0), (\sigma, 2)(\sigma, 2)(\alpha, 0)))
\end{array}
$$

The derivation relation $\Rightarrow_{root(s)}$ is defined on trees over the working symbols σ, α, $(\sigma, 1)$, $(\sigma, 2)$, and $(\alpha, 0)$ and constructs ξ of the form $c(p, \ldots)$ where c is a synthesized or inherited function and p is a path through $root(s)$. For instance, the tree

$$\xi = up(121, (\alpha, 0))$$

denotes the value of the inherited function up at the node of $root(s)$ which is reached by the path 121, and with the context information $(\alpha, 0)$. Now let us recall and explain the marked derivation step (cf. Figure 4):

$$\begin{array}{lll} \xi_1 = & \sigma((\sigma,1)(\alpha,0), \sigma(\ up(121,(\alpha,0)) & ,down(122))) \\ \Rightarrow^{(4)}_{root(s)} & \sigma((\sigma,1)(\alpha,0), \sigma(\ up(12,(\sigma,1)(\alpha,0)) & ,down(122))) = \xi_2. \end{array}$$

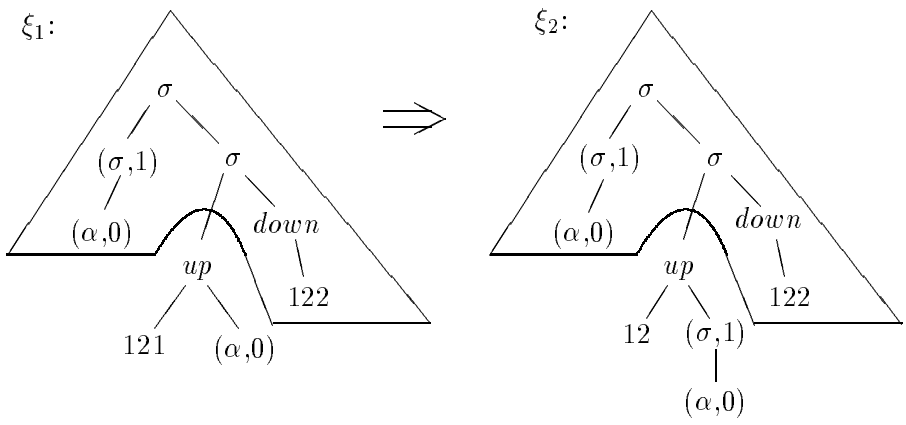

Figure 4: Derivation step.

We observe that ξ is a subtree of ξ_1 and that the left–hand side of rule (4) matches with ξ under the substitution:

- $z = 12$
- $y_1 = (\alpha, 0)$

Hence, ξ can be replaced by ξ' which is obtained from the right–hand side of rule (4) under this substitution; this replacement yields ξ_2.

In general, we designate one synthesized function a_{in} of an mat tree transducer M as the main function; it has the arity one, i.e., it does not have parameter positions. Then the tree transformation $\tau(M)$ which is induced by M, is defined by

$$\tau(M)(s) = t \quad \text{iff} \quad a_{in}(\varepsilon) \Rightarrow^*_{root(s)} t$$

In order to have a uniform denotation of tree transducers and classes of tree transformations computed by tree transducers, we introduce the notion of uv–tree transducer, where u and v are elements of the sets $MOD_s = \{s, s_f\}$ and $MOD_h = \{i, i_f, \varepsilon\}$ of modifiers, respectively (ε denotes the empty word). The modifiers s, s_f, i, i_f mean respectively that the tree transducer may use synthesized attributes, synthesized functions, inherited attributes, and inherited functions, where attributes are functions with arity one. Thus, top–down tree transducers, attributed tree transducers, and macro tree transducers are

closely related to s–tree transducers, si–tree transducers, and s_f–tree transducers, respectively (cf. Observation 3.6). In particular, the example transducer M_τ is an si_f–tree transducer. If it is not necessary to be precise about the combination of modifiers, then we will talk about mat tree transducers (which are $s_f i_f$–tree transducers). The classes of tree functions computed by uv–tree transducers are denoted by the corresponding capitals of the modifiers followed by a T; thus, e.g., the class of tree functions computed by si–tree transducers is denoted by SIT.

In this paper we start the investigation about the relationships between the classes of tree transformations which are computed by uv–tree transducers. We present an inductive definition of the tree transformation computed by an mat tree transducer and we prove the following results:

- Mat tree transducers are strictly more powerful than both, attributed tree transducers and macro tree transducers, i.e., $SIT \cup S_f T \subset S_f I_f T$.
- Mat tree transducers are equivalent to the two–fold composition of si–tree transducers, i.e., $S_f I_f T = SIT \circ SIT$.
- (As corollary) Mat tree transducers form a strict composition hierarchy, i.e., $S_f I_f T^n \subset S_f I_f T^{n+1}$ for every $n \geq 1$ where $S_f I_f T^n$ denotes the n–fold composition of the class $S_f I_f T$.
- (As corollary) Mat tree transducers are closed under right–composition with top–down tree transducers, but not under left–composition, i.e., $S_f I_f T \circ ST \subseteq S_f I_f T$ and $ST \circ S_f I_f T \not\subseteq S_f I_f T$.
- In the environment of synthesized functions, the addition of inherited attributes does not increase the computational power, i.e., $S_f IT = S_f T$.

This paper is divided into five sections. In Section 2 we collect all the prerequisites so that the paper is self contained. In Section 3 we introduce the formal model of mat tree transducer, deal with the problem of circularity, provide an inductive characterization of the tree transformation computed by mat tree transducers, and prove two elementary properties of mat tree transducers. In Section 4 we prove the results concerning the classes of tree transformations. Finally, in Section 5 we provide a summary and a list of further research topics.

In this paper we do not present the details of some proofs in Section 3 and Section 4. These details are carried out in the Appendix of the technical report [KV92].

2 Preliminaries

In this section we collect the notations which are used throughout this paper.

General notations

For every $m \geq 0$, the set $\{1, \ldots, m\}$ is denoted by $[m]$. The empty word is denoted by ε. We will use the sets $Y = \{y_1, y_2, \ldots\}$ and $Z = \{z_1, z_2, \ldots\}$ of *parameters*. For $k \geq 0$, $Y_k = \{y_1, \ldots, y_k\}$ and similarly for Z_k. For an arbitrary set S, the set of all subsets of S is denoted by $\mathcal{P}(S)$.

For a string v and two lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of strings, we abbreviate by $v[u_1/v_1, \ldots, u_n/v_n]$ the string which is obtained from v by replacing u_i by v_i assuming that $u_i \neq u_j$ for every $i \neq j$. The resulting string is also denoted by $v[u_i/v_i\,;\, i \in [n]]$. For a string w and a position ν with $1 \leq \nu \leq length(w)$, $w(\nu)$ denotes the ν'th letter of w.

Let $\Rightarrow$ be a binary relation on some set S. Then, for every $n \geq 0$, $\Rightarrow^n$ denotes the n–fold composition of $\Rightarrow$, and $\Rightarrow^*$ and $\Rightarrow^+$ denote the transitive, reflexive closure of $\Rightarrow$ and the transitive closure of $\Rightarrow$, respectively. The normalform of $s \in S$ with respect to $\Rightarrow$ is denoted by $nf(\Rightarrow, s)$ if it exists. $\Rightarrow$ is *confluent* (or: *locally confluent*), if for every $s, s_1, s_2 \in S$ with $s \Rightarrow^* s_1$ and $s \Rightarrow^* s_2$ ($s \Rightarrow s_1$ and $s \Rightarrow s_2$, respectively), there is an $s' \in S$ such that $s_1 \Rightarrow^* s$ and $s_2 \Rightarrow^* s$. $\Rightarrow$ is *noetherian* or *terminating*, if there is no infinite derivation sequence of $\Rightarrow$. Recall that, if $\Rightarrow$ is noetherian and locally confluent, then $\Rightarrow$ is confluent (cf. Lemma 2.4 of [Hue80]). Moreover, if $\Rightarrow$ is noetherian and confluent, then for every $s \in S$, the normalform of s exists and it is unique.

Ranked alphabets, trees, and tree transformations

A *ranked alphabet* is a pair $(\Sigma, rank_\Sigma)$ where Σ is a final set and $rank_\Sigma : \Sigma \longrightarrow I\!N$ is a mapping which associates with every symbol a nonnegative integer called the rank of the symbol. If $\sigma \in \Sigma$ and $rank_\Sigma(\sigma) = n$ then we also write $\sigma^{(n)}$. If it is clear from the context, then the rank function is dropped from the notation. $\Sigma^{(n)}$ denotes the set of elements with rank n.

For a ranked alphabet Σ and an arbitrary set S, the set of *trees over* Σ *indexed by* S, denoted by $T\langle\Sigma\rangle(S)$, is the smallest set T such that (i) $S \subseteq T$ and (ii) for every $\sigma \in \Sigma^{(n)}$ with $n \geq 0$ and $t_1, \ldots, t_n \in T$, $\sigma(t_1, \ldots, t_n) \in T$. For a symbol $\sigma \in \Sigma^{(0)}$ we simply write σ instead of $\sigma()$. $T\langle\Sigma\rangle(\emptyset)$ is abbreviated by $T\langle\Sigma\rangle$.

For a tree $t \in T\langle\Sigma\rangle(S)$, the following six objects are defined inductively on the structure of t: the *height* of t, denoted by $height(t)$; the *size* of t, denoted by $size(t)$, the *set of paths* of t, denoted by $path(t)$; the set of *subtrees* of t, denoted by $sub(t)$; and the *label of the node of* t *which is reached by a path* w, denoted by $label(t, w)$.

1. If $t \in \Sigma^{(0)} \cup S$, then
   - $height(t) = 1$

- $size(t) = 1$
- $path(t) = \{\varepsilon\}$
- $sub(t) = \{t\}$
- $label(t, \varepsilon) = t$.

2. If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$ and $n > 0$, then
   - $height(\sigma(t_1, \ldots, t_n)) = 1 + max\{height(t_i) | i \in [n]\}$
   - $size(\sigma(t_1, \ldots, t_n)) = 1 + \sum_{i \in [n]} size(t_i)$
   - $path(t) = \{\varepsilon\} \cup \{w | w = iv, i \in [n], v \in path(t_i)\}$
   - $sub(t) = \{t\} \cup \bigcup_{i \in [n]} sub(t_i)$
   - if $w = \varepsilon$, then $label(t, w) = \sigma$, and if $w = iv$ for some $i \in [n]$, then $label(t, w) = label(t_i, v)$.

We extend sub in an obvious way to sets $T \subseteq T\langle\Sigma\rangle(S)$ by defining $sub(T) = \bigcup_{t \in T} sub(t)$.

For two ranked alphabets Σ and Δ, let f be a mapping from $\Sigma^{(0)}$ to $T\langle\Delta\rangle(Y)$. Then f is extended to a mapping $YIELD_f : T\langle\Sigma\rangle \longrightarrow T\langle\Delta\rangle(Y)$ by

1. for $\sigma \in \Sigma^{(0)}$, $YIELD_f(\sigma) = f(\sigma)$
2. for $\sigma \in \Sigma^{(k+1)}$ with $k \geq 0$ and $t_0, t_1, \ldots, t_k \in T\langle\Sigma\rangle$, $YIELD_f(\sigma(t_0, t_1, \ldots, t_k)) = YIELD_f(t_0)[y_1/YIELD_f(t_1), \ldots, y_k/YIELD_f(t_k)]$.

Note that, in the second part of the definition, $YIELD_f(t_0)$ may contain a parameter y_n with $n > k$.

For two ranked alphabets Σ and Δ, a function $f : T\langle\Sigma\rangle \longrightarrow T\langle\Delta\rangle$ is called *tree transformation*.

Induction principles

In this paper we use three kinds of induction principles. Besides the well known principles of (mathematical) induction (to define functions on $\mathbb{N}$ and to prove predicates on $\mathbb{N}$) and of structural induction (to define functions on $T\langle\Sigma\rangle$ and to prove predicates on $T\langle\Sigma\rangle$), we introduce a kind of simultaneous induction. Actually, this induction is a special case of the usual induction for trees which are built up over a many–sorted ranked alphabet [GTWW77] (cf. Section 2.5 of [EV85] for an explanation of this connection).

Let Σ be a ranked alphabet, let A be a set, and for every $\sigma \in \Sigma$, let B_σ be a set. The mapping

$$f : T\langle\Sigma\rangle \longrightarrow A$$

and for every $k \geq 0$ and $\sigma \in \Sigma^{(k)}$, the mapping

$$g_\sigma : (T\langle\Sigma\rangle)^k \longrightarrow B_\sigma$$

can be defined by simultaneous induction. The *principle of definition by simultaneous induction* is the following formula:

If

1. for every $\sigma(s_1, \ldots, s_k)$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, if $g_\sigma((s_1, \ldots, s_k))$ is defined, then $f(\sigma(s_1, \ldots, s_k))$ is defined and
2. for every $(s_1, \ldots, s_k) \in (T\langle\Sigma\rangle)^k$ with $k \geq 0$ and $\sigma \in \Sigma^{(k)}$, if $f(s_1), \ldots, f(s_k)$ are defined, then $g_\sigma((s_1, \ldots, s_k))$ is defined,

then for every $s \in T\langle\Sigma\rangle$, $k \geq 0$, $\sigma \in \Sigma^{(k)}$, $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, the values $f(s)$ and $g_\sigma((s_1, \ldots, s_k))$ are defined.

Note, that the induction base is contained in case 2. for $k = 0$.

Let Σ be a ranked alphabet. The predicate

$$P \text{ over } T\langle\Sigma\rangle$$

and for every $k \geq 0$ and $\sigma \in \Sigma^{(k)}$, the predicate

$$Q_\sigma \text{ over } (T\langle\Sigma\rangle)^k$$

can be proved by simultaneous induction. The *principle of proof by simultaneous induction* is the following formula:

If

1. for every $\sigma(s_1, \ldots, s_k)$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, if $Q_\sigma((s_1, \ldots, s_k))$ is true, then $P(\sigma(s_1, \ldots, s_k))$ is true and
2. for every $(s_1, \ldots, s_k) \in (T\langle\Sigma\rangle)^k$ with $k \geq 0$ and $\sigma \in \Sigma^{(k)}$, if $P(s_1), \ldots, P(s_k)$ are true, then $Q_\sigma((s_1, \ldots, s_k))$ is true,

then for every $s \in T\langle\Sigma\rangle$, $k \geq 0$, $\sigma \in \Sigma^{(k)}$, $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, the formulas $P(s)$ and $Q_\sigma((s_1, \ldots, s_k))$ are true.

Again the induction base is contained in case 2. for $k = 0$.

3 Definition and basic properties

In this section we define the syntax of mat tree transducers and the derivation relations which are induced by them. If we restrict the transducers to be noncircular, then their derivation relations are confluent and noetherian, and every noncircular transducer computes a (total) tree transformation. We present a circularity test. For noncircular mat tree transducers we prove an inductive characterization of the computed tree transformations which does not refer to the derivation relation of the transducer. Finally, we prove some basic properties of mat tree transducers.

3.1 The formal model

An essential part of an mat tree transducer M is its function system. This system fixes the set F of function symbols, which may occur in M, and it separates F into the sets of synthesized and inherited function symbols. The set of function symbols is defined as a ranked alphabet; every function symbol has at least rank 1 in order to provide a position for the control tree.

Definition 3.1 A *system of function symbols* is a tuple $\mathcal{F} = (F, F_s, F_h)$ where

- F is a ranked alphabet of *function symbols*; for every $c \in F$, $rank_F(c) \geq 1$.
- $F_s \subseteq F$ and $F_h \subseteq F$ are the disjoint sets of *synthesized function symbols* and *inherited function symbols*, respectively, with $F = F_s \cup F_h$. □

We call a synthesized (or inherited) function symbol with rank 1 a *synthesized attribute* (and *inherited attribute*, respectively). In the sequel, we shall use the slightly inaccurate notions of 'synthesized functions' and 'inherited functions' rather than 'synthesized function symbols' and 'inherited function symbols', respectively. In analogy to macro tree transducers, we designate the first argument of every function symbol to contain its *control argument* s and a path through s, the other arguments are called *context parameters*. Actually, in the first argument of a function, only a path through s will occur, the control tree itself will parameterize the derivation relation (cf. Definition 3.8).

A system of function symbols is the first component in the definition of an mat tree transducer M. Additionally, we have to specify a ranked input alphabet Σ. Then, intuitively, M takes an argument s where s is the control tree over Σ, on which the evaluation of function values is performed. An output tree is built up over a ranked alphabet Δ of working symbols; we require that $\Sigma \subseteq \Delta$. The derivations of M will start with a designated synthesized function a_{in} of rank 1 and with an extra marker *root* on top of the control tree where *root* is a new symbol of rank 1 (cf. Figure 3 in the introduction). Of course, the kernel of the definition of an mat tree transducer is the finite set of rewrite rules; for a few moments, we postpone the definition of the possible right–hand sides of rules.

Definition 3.2 A *macro attributed tree transducer* (for short: mat tree transducer) is a tuple $(\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ where

- $\mathcal{F} = (F, F_s, F_h)$ is a system of function symbols
- Δ is the ranked alphabet of *working symbols* with $F \cap \Delta = \emptyset$
- Σ is the ranked alphabet of *input symbols* with $\Sigma \subseteq \Delta$
- $a_{in} \in F_s^{(1)}$ is the *initial synthesized function*
- $root$ is a symbol of rank 1, called the *root marker*, where $root \notin F \cup \Delta$
- $R = \bigcup_{\sigma \in \Sigma \cup \{root\}} R_\sigma$ is a finite set of *rules*, defined by Condition 1. and 2.

  1. For every $a \in F_s^{(p+1)}$ with $p \geq 0$, the set R_{root} contains exactly one rule of the form
  $$a(z, y_1, \ldots, y_p) \to \zeta$$
  with $\zeta \in RHS(F_s, \emptyset, \Delta, Y_p, root)$.
  For every $b \in F_h^{(p+1)}$ with $p \geq 0$, the set R_{root} contains exactly one rule of the form
  $$b(z1, y_1, \ldots, y_p) \to \zeta$$
  with $\zeta \in RHS(F_s, \emptyset, \Delta, Y_p, root)$.
  2. For every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$ and for every $a \in F_s^{(p+1)}$ with $p \geq 0$, the set R_σ contains exactly one rule of the form
  $$a(z, y_1, \ldots, y_p) \to \zeta$$
  with $\zeta \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$.
  For every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, for every $b \in F_h^{(p+1)}$ with $p \geq 0$ and for every $j \in [k]$, the set R_σ contains exactly one rule of the form
  $$b(zj, y_1, \ldots, y_p) \to \zeta$$
  with $\zeta \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$. □

For an mat tree transducer $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$, we fix the following notions and notations.

- The set $\Sigma \cup \{root\}$ is denoted by Σ_+.
- In the rules of R, the symbol z is called *path variable*.
- For every $\sigma \in \Sigma_+^{(k)}$, the set of *inside function occurrences* of σ, denoted by $in(\sigma)$, is the set $\{(b, zj) | b \in F_h, j \in [k]\} \cup \{(a, z) | a \in F_s\}$. The set of *outside function occurrences* of σ, denoted by $out(\sigma)$, is the set $\{(b, z) | b \in F_h\} \cup \{(a, zj) | a \in F_s, j \in [k]\}$. The set of *function occurrences* of σ, denoted by $fun(\sigma)$, is the set $in(\sigma) \cup out(\sigma)$.

- For $c \in F^{(p+1)}$, $\sigma \in \Sigma_+^{(k)}$ and $\eta \in \{zj | j \in [k] \cup \{\varepsilon\}\}$, we call a rule of R_σ with the left–hand side $c(\eta, y_1, \ldots, y_p)$ a (c, η, σ)–rule or just c–rule. The right–hand side of this rule is denoted by $rhs(c, \eta, \sigma)$ and we abbreviate $rhs(\sigma) = \{rhs(c, \eta, \sigma) | (c, \eta) \in in(\sigma)\}$.

Thus, for every $(c, \eta) \in in(\sigma)$, there is exactly one (c, η, σ)–rule in R. Next we define the set $RHS(G_s, G_h, \Delta, Y_p, \sigma)$ of possible right–hand sides for (c, η, σ)–rules where $c \in F^{(p+1)}$, $\sigma \in \Sigma_+$, $G_s \subseteq F_s$ and $G_h \subseteq F_h$.

Definition 3.3 Let $\mathcal{F} = (F, F_s, F_h)$ be a system of function symbols and let $G_s \subseteq F_s$ and $G_h \subseteq F_h$, Δ a ranked alphabet, $p \geq 0$, and $\sigma \in \Sigma_+^{(k)}$ with $k \geq 0$. The set of (Y_p, σ)–*right–hand sides over* G_s, G_h *and* Δ, denoted by $RHS(G_s, G_h, \Delta, Y_p, \sigma)$, is the smallest subset RHS of $(G_s \cup G_h \cup \Delta \cup Y_p \cup [k] \cup \{z, (,), ,\})^*$ such that the following four conditions hold:

(i) $Y_p \subseteq RHS$.

(ii) For every $\delta \in \Delta^{(r)}$ with $r \geq 0$, and $\zeta_1, \ldots, \zeta_r \in RHS$, $\delta(\zeta_1, \ldots, \zeta_r) \in RHS$.

(iii) For every $a \in G_s^{(r+1)}$ with $r \geq 0$, $i \in [k]$, and $\zeta_1, \ldots, \zeta_r \in RHS$, $a(zi, \zeta_1, \ldots, \zeta_r) \in RHS$.

(iv) For every $b \in G_h^{(r+1)}$ with $r \geq 0$, and $\zeta_1, \ldots, \zeta_r \in RHS$, $b(z, \zeta_1, \ldots, \zeta_r) \in RHS$. □

We note that in $rhs(c, \eta, \sigma)$ only outside function occurrences of σ appear. We also note that in the right–hand side of a $(b, z1, root)$–rule where b is an inherited function, no inherited function at z may occur, because we have chosen $G_h = \emptyset$ in this case. Let us illustrate the definitions by means of an example taken from [Ems91].

Example 3.4 Let $\Delta = \{\sigma^{(2)}, \alpha^{(0)}, \gamma^{(1)}\}$ be a ranked alphabet of working symbols. Let $\Sigma = \{\sigma, \alpha\}$ be a subset of Δ. The system of function symbols $\mathcal{F} = (F, F_s, F_h)$ is defined by

- $F = \{a_{in}^{(1)}, a^{(2)}, b^{(2)}\}$
- $F_s = \{a_{in}, a\}$
- $F_h = \{b\}$.

Let $R = R_{root} \cup R_\sigma \cup R_\alpha$ be the following set of rules:

$$
\begin{array}{lllll}
R_{root}: & (1) & a_{in}(z) & \to & a(z1,\alpha) \\
 & (2) & a(z,y_1) & \to & \alpha \\
 & (3) & b(z1,y_1) & \to & \gamma(y_1) \\
R_{\sigma}: & (4) & a_{in}(z) & \to & \alpha \\
 & (5) & a(z,y_1) & \to & a(z2,a(z2,y_1)) \\
 & (6) & b(z1,y_1) & \to & b(z,b(z,y_1)) \\
 & (7) & b(z2,y_1) & \to & a(z1,a(z1,y_1)) \\
R_{\alpha}: & (8) & a_{in}(z) & \to & \alpha \\
 & (9) & a(z,y_1) & \to & b(z,b(z,y_1))
\end{array}
$$

Then $M_{doubleexp} = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ is an mat tree transducer. We note without further explanation at this point that, given a tree $s \in T\langle\Sigma\rangle$, the result of the derivation starting from $a_{in}(\varepsilon)$ is the tree $\gamma^m(\alpha)$ where $m = 2^{2 \cdot size(s)-1}$ (cf. Theorem 4.2). □

We note that rules (2), (4), and (8) are only mentioned, because by definition the set R has to be complete (for every $(c,\eta) \in in(\sigma)$ there is exactly one (c,η,σ)–rule in R). It is not so difficult to see already here that the rules (2), (4) and (8) are never used in a computation.

Before working out the definition of the derivation relation, we first introduce a uniform classification scheme for subclasses of mat tree transducers. Recall that a synthesized (or inherited) function of arity one is called synthesized (and inherited, respectively) attribute.

Definition 3.5 Let $MOD_s = \{s_f, s\}$ and $MOD_h = \{\varepsilon, i_f, i\}$ be two sets of modifiers, where intuitively, s_f, s, i_f, and i abbreviate synthesized function, synthesized attribute, inherited function, and inherited attribute, respectively. Let $u \in MOD_s$ and $v \in MOD_h$.

A *uv–tree transducer* M is an mat tree transducer for which its system $\mathcal{F} = (F, F_s, F_h)$ of functions is restricted as follows:

- If $uv = s_f i_f$, then there are no restrictions.
- If $uv = s_f i$, then $F_h = F_h^{(1)}$, i.e., M may only use synthesized functions and inherited attributes.
- If $uv = s i_f$, then $F_s = F_s^{(1)}$, i.e., M may only use synthesized attributes and inherited functions.
- If $uv = si$, then $F_s = F_s^{(1)}$ and $F_h = F_h^{(1)}$, i.e., M may only use synthesized attributes and inherited attributes.
- If $uv = s_f$, then $F = F_s$, i.e., M may only use synthesized functions and no inherited functions.
- If $uv = s$, then $F = F_s$ and $F_s = F_s^{(1)}$, i.e., M may only use synthesized attributes and no inherited functions. □

Some of the classes of tree transducers which are known from the literature, fit into this classification scheme apart from different syntactic sugaring. Moreover, in our classification, we always assume that for every synthesized function $a \in F_s^{(p+1)}$, the right–hand side of the $(a, z, root)$–rule has the form $a(z1, y_1, \ldots, y_p)$ (cf. the notion of *mat tree transducer with initial chain rules* of Definition 3.29 for this restriction and Theorem 3.30). In the following observation we always refer to the total deterministic versions of the mentioned tree transducers.

Observation 3.6

1. Top–down tree transducers [Rou70, Tha70] are s–tree transducers.

2. Macro tree transducers [Eng80, CF82, EV85] are s_f–tree transducers.

3. Attributed tree transducers [Fül81] are si–tree transducers in which, for every inherited attribute b, the right–hand side of $(b, z1, root)$–rules may not contain synthesized attributes. In Theorem 3.2 of [Fül81], Fülöp gave an example τ of a tree transformation which cannot be computed by an attributed tree transducer. However, dropping the mentioned restriction on right–hand sides of $(b, z1, root)$–rules, Giegerich proved in [Gie88] that τ can be computed by attributed tree transducers. (There, attributed tree transducers are special instances of attribute coupled grammars). In fact, we follow the suggestion in [Gie88] to generalize the concept of attributed tree transducer by dropping the restriction on initialization rules for inherited attributes (for a further discussion the reader is referred to the appendix of [Gie88]). In accordance to [Gie88] we also call si–tree transducers *full attributed tree transducers*. However, we note that in [FV91] Fülöp and Vagvölgyi gave an example τ' of a tree transformation which even cannot be computed by full attributed tree transducers.

4. By definition, macro attributed tree transducers are $s_f i_f$–tree transducers. Macro attributed tree transducers are closely related to total deterministic context–free tree grammars with tree–walking storage type, i.e. $CFT(TW)$–transducers (cf. [Eng86], [EV86]) for the concept of grammar with storage). The nesting of synthesized and inherited functions is imitated by the nesting of nonterminals in right–hand sides of rules of a context–free tree grammar. The control tree with a path is represented as configuration of the storage type TW.

5. In [KSV89] higher order attribute grammars are defined. Roughly speaking, in such grammars the control tree can be expanded during attribute evaluation by an attribute value (assuming that it has the form of a tree). If this additional feature is used restrictedly in such a way that a higher order attribute grammar M can be simulated by the two–fold composition of attribute grammars, then (as we will show later) M can also be simulated by an mat tree transducer.

In Figure 5 we arrange the classes of tree transducers in a diagramm, where we abbreviate tree transducer by tr^2. If one moves from class A to class B via an arc which is labelled by lab, then B is obtained from A by:

- adding inherited attributes, if $lab = +i$,
- allowing inherited functions with arbitrary rank (greater 0), if $lab = i \to i_f$,
- allowing synthesized functions with arbitrary rank (greater 0), if $lab = s \to s_f$.

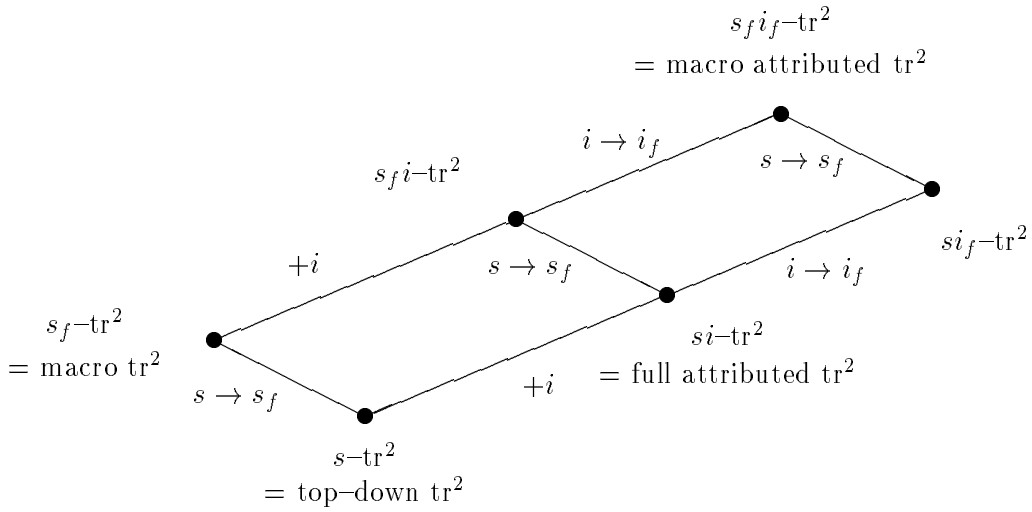

Figure 5: Classes of tree transducers.

In the next definition we inductively describe the set of all sentential forms of mat tree transducers. For a given control tree s, a sentential form is a tree over function symbols, working symbols, and paths through s. Moreover, the first argument of a function always contains a path through s and vice versa a path may only occur in the first argument of a function. For later purposes, we immediately define the sentential forms with respect to trees $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$ where $root$ is the root–marker of M, and not only for trees over Σ. Moreover, the paths which occur in the forms, are restricted to be elements of an arbitrary subset $P \subseteq path(\tilde{s})$.

Note that we also incorporate the parameters of Y in the sentential forms, because this will turn out to be more appropriate when characterizing the tree transformation induced by an mat tree transducer (see Definition 3.25). Roughly speaking, a parameter which occurs in a sentential form can be viewed as an additional working symbol; in particular, it is not used as substitution symbol.

Definition 3.7 Let $\mathcal{F} = (F, F_s, F_h)$ be a system of function symbols and let $G_s \subseteq F_s$ and $G_h \subseteq F_h$, and Δ and Σ be two ranked alphabets. Moreover, let $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$ and $P \subseteq path(\tilde{s})$. The set of (G_s, G_h, P, Δ)–*sentential forms*, denoted by $SF(G_s, G_h, P, \Delta)$, is defined inductively as follows where we abbreviate $SF(G_s, G_h, P, \Delta)$ by SF.

(i) $Y \subseteq SF$.

(ii) For every $\delta \in \Delta^{(k)}$ with $k \geq 0$ and $\xi_1, \ldots, \xi_k \in SF$, $\delta(\xi_1, \ldots, \xi_k) \in SF$.

(iii) For every $a \in F_s^{(p+1)}$ with $p \geq 0$, $w \in P$, and $\xi_1, \ldots, \xi_p \in SF$, $a(w, \xi_1, \ldots, \xi_p) \in SF$.

(iv) For every $b \in F_h^{(p+1)}$ with $p \geq 0$, $w \in P$ with $w \neq \varepsilon$, and $\xi_1, \ldots, \xi_p \in SF$, $b(w, \xi_1, \ldots, \xi_p) \in SF$. □

Notice that the control tree $\tilde{s}$ does not occur in sentential forms. It is only needed to define the subset P of the paths of $\tilde{s}$.

Now we describe the derivation relation of an mat tree transducer M with respect to a tree $\tilde{s}$. We note that the substitution function ϕ which is used in the following definition, will be defined at the end of the definition. Recall that Σ_+ abbreviates the set $\Sigma \cup \{root\}$.

Definition 3.8 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer with system $\mathcal{F} = (F, F_s, F_h)$ of functions. Let $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$ and $P \subseteq path(\tilde{s})$. The *derivation relation of M with respect to $\tilde{s}$ and P*, denoted by $\Rightarrow_{M,\tilde{s},P}$, is a binary relation on $SF(F_s, F_h, path(\tilde{s}), \Delta)$ defined as follows:

For $\Phi, \Psi \in SF(F_s, F_h, path(\tilde{s}), \Delta)$, $\quad \Phi \Rightarrow_{M,\tilde{s},P} \Psi$ if

- there is a $\beta \in SF(F_s, F_h, path(\tilde{s}), \Delta \cup \{u\})$ in which the 0–ary symbol $u \notin F \cup \Delta$ occurs exactly once,
- there is a function $c \in F^{(p+1)}$ with $p \geq 0$,
- there is a path $w \in P$,
- there are $\xi_1, \ldots, \xi_p \in SF(F_s, F_h, path(\tilde{s}), \Delta)$,

such that $\Phi = \beta[u/c(w, \xi_1, \ldots, \xi_p)]$ and if one of the following two conditions holds:

1. - c is a synthesized function
   - $label(\tilde{s}, w) = \sigma$ for some $\sigma \in \Sigma_+^{(k)}$ with $k \geq 0$
   - there is a rule $c(z, y_1, \ldots, y_p) \to \zeta$ in R_σ and
   - $\Psi = \beta[u/\phi_{w,\tilde{\xi}}(\zeta)]$ with $\tilde{\xi} = (\xi_1, \ldots, \xi_p)$.
2. - c is an inherited function
   - $w = vj$ for some $v \in path(\tilde{s})$, $label(\tilde{s}, v) = \sigma$ for some $\sigma \in \Sigma_+^{(k)}$ with $k \geq 1$, and $j \in [k]$
   - there is a rule $c(zj, y_1, \ldots, y_p) \to \zeta$ in R_σ and
   - $\Psi = \beta[u/\phi_{v,\tilde{\xi}}(\zeta)]$ with $\tilde{\xi} = (\xi_1, \ldots, \xi_p)$.

For every $p \geq 0$, $\sigma \in \Sigma_+^{(k)}$ with $k \geq 0$, $\theta \in path(\tilde{s})$, and $\tilde{\xi} = (\xi_1, \ldots, \xi_p) \in (SF(F_s, F_h, path(\tilde{s}), \Delta))^p$, the function

$$\phi_{\theta,\tilde{\xi}} : RHS(F_s, F_h, \Delta, Y_p, \sigma) \longrightarrow SF(F_s, F_h, path(\tilde{s}), \Delta)$$

transforms every local path into a global path of $\tilde{s}$ and it replaces every y_j by ξ_j:

$$\phi_{\theta,\tilde{\xi}}(\zeta) = \zeta[z/\theta][y_j/\xi_j; j \in [p]].$$

□

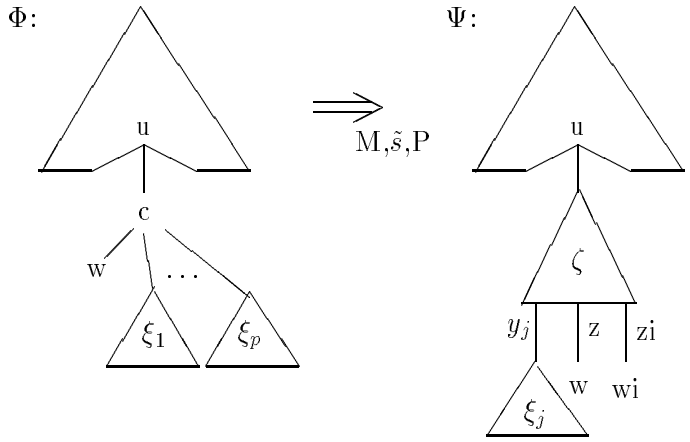

Figure 6: Derivation step.

In Figure 6 a derivation step is shown for which the first condition holds. If $\Phi \Rightarrow_{M,\tilde{s},P} \Psi$, then we also say that Φ is *rewritten* to Ψ and that Ψ is the *result* of the application of the rule. If M or $\tilde{s}$ are known from the context, we drop the corresponding indices from $\Rightarrow$. If $P = path(\tilde{s})$, then we drop P. The tree $\tilde{s}$ is also called the *control tree* of $\Rightarrow_{M,\tilde{s},P}$, because it controls the derivation of the transducer.

If $c(w, t_1, \ldots, t_n)$ with $c \in F$, $w \in path(\tilde{s})$ and $t_1, \ldots, t_n \in SF(F_s, F_h, path(\tilde{s}), \Delta)$ is a subtree of a tree $\Phi \in SF(F_s, F_h, path(\tilde{s}), \Delta)$, we call (c, w) a function occurrence in Φ. Notice that for given c and w, there can be more than one function occurrence (c, w) in a sentential form (cf. Example 3.9).

Example 3.9 Consider the mat tree transducer $M_{doubleexp}$ of Example 3.4. Intuitively, this transducer performs a depth–first right–to–left tree traversal over its control tree and, in every step, the rest of the tree–walk is nested in the parameter position. Let $s = \sigma(\sigma(\alpha, \alpha), \alpha) \in T\langle\Sigma\rangle$ be the control tree. We abbreviate $\Rightarrow_{M_{doubleexp}, root(s), path(root(s))}$ and $root(s)$ by $\Rightarrow$ and $\tilde{s}$, respectively. The index of the applied rule is indicated as a superscript. After having shown some derivation steps, we will explain one of them in some more detail.

$$
\begin{aligned}
& & a_{in}(\varepsilon) \\
\Rightarrow^{(1)} & & a(1,\alpha) \\
\Rightarrow^{(5)} & & a(12,a(12,\alpha)) \\
\Rightarrow^{(9)} & & b(12,b(12,a(12,\alpha))) \\
\Rightarrow^{(7)} & & a(11,a(11,b(12,a(12,\alpha)))) \\
\Rightarrow^{(5)} & & a(112,a(112,a(11,b(12,a(12,\alpha))))) \\
\Rightarrow^{(9)} & & b(112,b(112,a(112,a(11,b(12,a(12,\alpha)))))) \\
\Rightarrow^{(7)} & (*1) & a(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha))))))) \\
\Rightarrow^{(9)} & & b(111,b(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha)))))))) \\
\Rightarrow^{(6)} & & b(11,b(11,b(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha))))))))) \\
\Rightarrow^{(6)} & & b(1,b(1,b(11,b(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha)))))))))) \\
\Rightarrow^{(3)} & & \gamma(b(1,b(11,b(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha)))))))))) \\
\Rightarrow^{(3)} & & \gamma(\gamma(b(11,b(111,a(111,b(112,a(112,a(11,b(12,a(12,\alpha)))))))))) \\
\Rightarrow & & \ldots
\end{aligned}
$$

In the derivation step which is marked by $(*1)$, we consider the function occurrence $(b, 112)$ at the root of the sentential form which we want to rewrite. Since $w = 112$, we have to apply the $(b, z2, label(\tilde{s}, 11))$–rule with $label(\tilde{s}, 11) = \sigma$ and $z = 11$, i.e. the rule

$$(7) \quad b(z2, y_1) \to a(z1, a(z1, y_1))$$

in R_σ of $M_{doubleexp}$. Then, in the right–hand side of the rule,

- the two occurrences of the path variable z are replaced by the path 11 of $\tilde{s}$ and
- y_1 is replaced by $b(112, a(112, a(11, b(12, a(12, \alpha)))))$. □

3.2 Confluence, termination, and computed tree function

Here we show that the derivation relations of mat tree transducers are locally confluent. We define the noncircularity property of mat tree transducers and prove that, if the transducers are restricted to be noncircular, then their derivation relations are noetherian and hence, by a result of Newman [New42], they are also confluent.

Now let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$ be a control tree. The proof that $\Rightarrow_{M,\tilde{s}}$ is locally confluent, is straightforward. Consider two function occurrences (c, w) and (c', w') of a sentential form $\Phi \in SF(F_s, F_h, path(\tilde{s}), \Delta)$ and let Ψ and Ψ' be the results of the application of rules r and r', respectively, to (c, w) and (c', w'). There are essentially two different cases. First, (c, w) and (c', w') are independent which means that none of these function occurrences is nested in the parameter position of the other function occurrence. Then, by applying rule r' to Ψ and rule r to Ψ', the same tree is obtained. Second, one function occurrence is nested in a parameter position of the other where we assume without loss of generality that (c', w') is nested in the j–th parameter position of (c, w). Assume that the right–hand

side of rule r contains k occurrences of y_j. Then, by applying rule r' k times to Ψ, the same tree is obtained as applying rule r to Ψ'. Since a similar proof is worked out in [FHVV93] (in the proof of local confluence of the derivation relations of macro tree transducers with external functions) we drop the formal proof here and just state the result.

Lemma 3.10 For every mat tree transducer $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ and every $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$, the derivation relation $\Rightarrow_{M,\tilde{s},path(\tilde{s})}$ is locally confluent. □

Since the class of mat tree transducers contains the class of attributed tree transducers as subclass, and since an attributed tree transducer can be circular (in the same sense as an attribute grammar), we can conclude that, in general, the derivation relations of mat tree transducers are not noetherian (cf., e.g., [Ems91] for an example of a circular attributed tree transducer.) However, noncircular mat tree transducers induce noetherian derivation relations. In order to show this property, we first have to define the notion of circularity which is a straightforward generalization of the concept of circularity of attributed tree transducers [Fül81].

Definition 3.11 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer.

1. M is *circular* if
   - there is an $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$
   - there is a sentential form $\Phi = c(w, t_1, \ldots, t_p) \in SF(F_s, F_h, path(\tilde{s}), \Delta)$ with $t_1, \ldots, t_p \in T\langle\Delta\rangle$
   - there is a $\beta \in SF(F_s, F_h, path(\tilde{s}), \Delta \cup \{u\})$ in which the 0–ary symbol u occurs exactly once
   - there are $\xi_1, \ldots, \xi_p \in SF(F_s, F_h, path(\tilde{s}), \Delta)$

   such that
   $$c(w, t_1, \ldots, t_p) \Rightarrow^+_{M,\tilde{s}} \Psi = \beta[u/c(w, \xi_1, \ldots, \xi_p)].$$
2. M is *noncircular* if it is not circular. □

On purpose we have considered only those sentential forms Φ in which the subtrees $t_1, \ldots, t_p$ are already in normalform. Then we can be sure that the occurrence of $c(w, \xi_1, \ldots, \xi_p)$ in Ψ is generated by the evaluation of the function occurrence (c, w) at the root of Φ and not by the parameter terms $t_1, \ldots, t_p$. We note that, in the previous definition, the node of β which is labeled by u, may occur nested in a parameter position of a function; even then we call the mat tree transducer circular (also cf. Example 3.31).

Clearly, if an mat tree transducer M is circular, then there is an infinite derivation. Now we have to show that, if M is noncircular, then for every control tree $\tilde{s}$, the derivation relation $\Rightarrow_{M,\tilde{s}}$ is noetherian.

First we make some observations about infinite derivations and we coin some useful notions (cf. [Ems91]). In the following let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$ be a control tree.

Definition 3.12 A tree θ from which an infinite derivation $\theta = \theta_0 \Rightarrow_{M,\tilde{s}} \theta_1 \Rightarrow_{M,\tilde{s}} \theta_2 \ldots$ starts, is called an *ω–reducible tree*.

A *minimal ω–reducible tree* θ is an ω–reducible tree such that no subtree of θ is an ω–reducible tree. □

Observation 3.13 Every ω–reducible tree has a minimal ω–reducible subtree. □

Observation 3.14 Every minimal ω–reducible tree has the form $c(w, \xi_1, \ldots, \xi_p)$ where the ξ_i's are not ω–reducible. □

Observation 3.15 If θ is a minimal ω–reducible tree, then there is an infinite derivation which starts from θ by the application of a rule to the root of θ. □

Now we can prove that noncircularity is a sufficient condition for the termination of the derivation relations of mat tree transducers.

Lemma 3.16 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer. If M is noncircular, then for every $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$, $\Rightarrow_{M,\tilde{s}}$ is noetherian.

Proof: We prove the implication by contradiction. Let M be noncircular and assume that, for some $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$, $\Rightarrow_{M,\tilde{s}}$ is not noetherian.

Then there is an ω–reducible tree Φ. By Observation 3.13, Φ contains a minimal ω–reducible subtree $\theta_{0,0}$. By Observation 3.14, $\theta_{0,0}$ has the form $c(w, \xi_1, \ldots, \xi_p)$ where the ξ_i's are not ω–reducible. By Observation 3.15, there is an infinite derivation

$$\theta_{0,0} \Rightarrow_{M,\tilde{s}} \theta_{0,1} \Rightarrow_{M,\tilde{s}} \theta_{0,2} \ldots$$

such that the first rule is applied to the root of $\theta_{0,0}$.

Clearly, $\theta_{0,1}$ is also ω–reducible and it contains a minimal ω–reducible subtree, say $\theta_{1,0}$. And there is an infinite derivation

$$\theta_{1,0} \Rightarrow_{M,\tilde{s}} \theta_{1,1} \Rightarrow_{M,\tilde{s}} \theta_{1,2} \ldots$$

such that the first rule is applied to the root of $\theta_{1,0}$.

Again, $\theta_{1,1}$ is ω–reducible and it contains a minimal ω–reducible subtree, say $\theta_{2,0}$. Obviously, this construction of minimal subtrees can be repeated and thereby we obtain two infinite sequences $(\theta_{i,0})_{i\geq 0}$ and $(\theta_{i,1})_{i\geq 0}$ such that for every $i \geq 0$

- $\theta_{i,0}$ has the form $c_i(w_i, \xi_{i,1}, \ldots, \xi_{i,p_i})$ and the $\xi_{i,j}$'s are not ω–reducible
- $\theta_{i,0} \Rightarrow_{M,s} \theta_{i,1}$ and the rule is applied to the root of $\theta_{i,0}$
- $\theta_{i+1,0}$ is a minimal ω–reducible subtree of $\theta_{i,1}$.

This means that, for every i, j with $j > i$ there is a $\beta_{i,j} \in SF(F_s, F_h, path(\tilde{s}), \Delta \cup \{u\})$ in which u occurs exactly once such that

$$\theta_{i,0} \Rightarrow^+_{M,\tilde{s}} \beta_{i,j}[u/\theta_{j,0}]$$

and in this derivation no rule is applied to the parameter positions of $\theta_{i,0}$.

Since $\theta_{i,0}$ has for every i the form $c_i(w_i, \xi_{i,1}, \ldots, \xi_{i,p_i})$ and since there are only finitely many functions in M and paths in $\tilde{s}$, there must be indices i_0, j_0 with $j_0 > i_0$ such that $c_{i_0} = c_{j_0}$ and $w_{i_0} = w_{j_0}$. Hence, there is a β such that

$$\begin{aligned}\theta_{i_0,0} = c_{i_0}(w_{i_0}, \xi_{i_0,1}, \ldots, \xi_{i_0,p_{i_0}}) &\Rightarrow^+_{M,\tilde{s}} \beta[u/c_{j_0}(w_{j_0}, \xi_{j_0,1}, \ldots, \xi_{j_0,p_{j_0}})] \\ &= \beta[u/c_{i_0}(w_{i_0}, \xi_{j_0,1}, \ldots, \xi_{j_0,p_{i_0}})]\end{aligned}$$

Since in this derivation no rule is applied to the trees in the parameter positions of $\theta_{i_0,0}$ it follows that even for every $t_1, \ldots, t_{p_{i_0}} \in T\langle\Delta\rangle$ there are $\xi_1, \ldots, \xi_{p_{i_0}}$ such that

$$c_{i_0}(w_{i_0}, t_1, \ldots, t_{p_{i_0}}) \Rightarrow^+_{M,\tilde{s}} \beta[u/c_{i_0}(w_{i_0}, \xi_1, \ldots, \xi_{p_{i_0}})].$$

Thus M is circular which is a contradiction, and hence M induces noetherian derivation relations only. □

Theorem 3.17 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer. If M is noncircular, then for every $\tilde{s} \in T\langle\Sigma\rangle \cup \{root(s) | s \in T\langle\Sigma\rangle\}$, $\Rightarrow_{M,\tilde{s}}$ is confluent and noetherian.

Proof: This follows immediately from the facts that $\Rightarrow_{M,\tilde{s}}$ is locally confluent and noetherian, and from a theorem of [New42]. □

Since the derivation relations of noncircular mat tree transducers are confluent and noetherian, every sentential form has a unique normalform. This is the basis for the definition of the tree transformation which is computed by an mat tree transducer. (Recall the meaning of the notation $nf(\Rightarrow, \xi)$ from the preliminaries.)

Definition 3.18 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be a noncircular mat tree transducer.

The *tree transformation computed by* M, denoted by $\tau(M)$ is the total function of type $T\langle\Sigma\rangle \longrightarrow T\langle\Delta\rangle$ defined as follows. For every $s \in T\langle\Sigma\rangle$,

$$\tau(M)(s) = nf(\Rightarrow_{M,root(s)}, a_{in}(\varepsilon)).$$

□

We denote the *classes of tree transformations computed* by noncircular $s_f i_f$–tree transducers, s_f–tree transducers, noncircular si–tree transducers, and s–tree transducers by $S_f I_f T$, $S_f T$, SIT, and ST, respectively. In the rest of this paper, we always mean noncircular $s_f i_f$–tree transducers when we talk about mat tree transducers.

3.3 Dependency graphs and circularity test

For mat tree transducers we can construct a circularity test which is very similar to the ordinary circularity test for attribute grammars, as it is described for example in [ASU86]. In fact, we will also count those circularities which result from nested function calls. This point of view is taken care of in the definition of dependency graph where we treat all the function occurrences in right–hand sides of rules in the same way, whether they occur nested in function parameters or not.

Now we define the concepts of dependency graph, extended dependency graph, and is–graph for mat tree transducers. These concepts are defined similarly to the corresponding concepts for attribute grammars (cf. e.g., [Cou84, Eng84]). Recall that, for an input symbol σ, $fun(\sigma)$ denotes the set of function occurrences of σ (cf. the list of notions behind Definition 3.2).

Definition 3.19 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $\sigma \in \Sigma_+^{(k)}$ with $k \geq 0$. The *dependency graph* of σ for M, denoted by $D_M(\sigma)$, is the directed graph $(fun(\sigma), E_{D(\sigma)})$ with $E_{D(\sigma)} = \{((c,\eta),(c',\eta')) \in out(\sigma) \times in(\sigma) \mid c(\eta,\ldots) \text{ occurs in } rhs(c',\eta',\sigma)\}$ as set of arcs. □

As an example, consider the synthesized functions $a_{in}^{(1)}$ and $a^{(1)}$ and the inherited function $b^{(2)}$. For the binary input symbol σ, let R_σ contain the following rules:

$$
\begin{array}{lcl}
a_{in}(z) & \rightarrow & b(z, a(z1)) \\
a(z) & \rightarrow & a(z2) \\
b(z1, y_1) & \rightarrow & a_{in}(z2) \\
b(z2, y_1) & \rightarrow & \alpha
\end{array}
$$

The dependency graph of σ is shown in Figure 7.

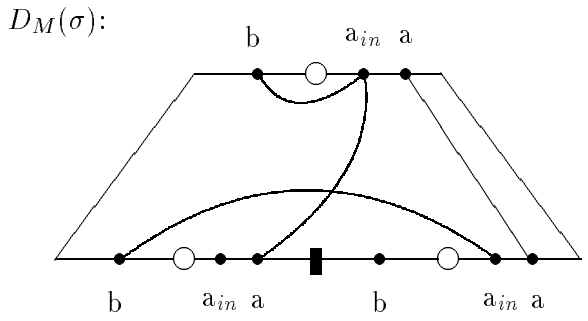

Figure 7: Dependency graph of σ.

It should be clear that, although a function may occur in the right–hand side of a rule, it is possible that this function is not needed to compute the initial synthesized function. For

instance in Example 3.31, the function a occurs in the right–hand side of the (a_{in}, z, σ)–rule although the function occurrence (a, z) of α is not needed to compute the value of a_{in} at the root of the control tree. Thus, roughly speaking, the definition of the dependency graph covers the worst case.

Next, for an mat tree transducer M, an input symbol $\sigma \in \Sigma_+^{(k)}$, and control trees $s_1, \ldots, s_k$, we define simultaneously the *extended dependency graph* of $(\sigma, s_1, \ldots, s_k)$ and the *is–graph* of $\sigma(s_1, \ldots, s_k)$. Intuitively, the is–graph of a tree s shows the dependencies between inherited function occurrences and synthesized function occurrences at the root of s, and the extended dependency graph of $(\sigma, s_1, \ldots, s_k)$ is the dependency graph of σ in which the is–graphs of $s_1, \ldots, s_k$ are pasted at the function occurrences of the sons of σ. In the sequel we also need the set of all is–graphs of M which we call the *collection of is–graphs* for M.

Definition 3.20 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ and $\mu = (\sigma, s_1, \ldots, s_k)$.

The *extended dependency graph* of μ for M, denoted by $D_{M,\mu}$, is the directed graph $(fun(\sigma), E_\mu)$, and the *is–graph* of s for M, denoted by $is_M(s)$, is the directed graph (F, E_s) where E_μ and E_s are defined simultaneously by structural induction on s as follows.

- For every $s \in \Sigma^{(0)}$, define $E_{(s)} = E_{D(s)}$ and $E_s = \{(c, c') \in F_h \times F_s \mid c \text{ occurs in } rhs(c', z, s)\}$.
- For every $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ with $k \geq 1$, define $E_{(\sigma,s_1,\ldots,s_k)} = E_{D(\sigma)} \cup \bigcup_{i\in[k]}\{((c, zi), (c', zi)) \mid (c, c') \in E_{s_i}\}$ and $E_s = \{(c, c') \mid$ there is a path in $D_{M,(\sigma,s_1,\ldots,s_k)}$ from (c, z) to $(c', z)\}$.

Define the *collection of is–graphs* of M, denoted by $is\text{–}set_M$, as follows:
$is\text{–}set_M := \{is_M(s) \mid s \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}\}$ □

We note that $is\text{–}set_M$ is a finite set, because there are only finitely many graphs with F as set of nodes. We also note that, e.g., in [Eng84], $D_{M,(\sigma,s_1,\ldots,s_k)}$ is denoted by $D(\sigma)[D_{s_1}, \ldots, D_{s_k}]$. To give an example, let us assume that there are trees s_1 and s_2 over the input alphabet of which the is–graphs are shown in Figure 8.

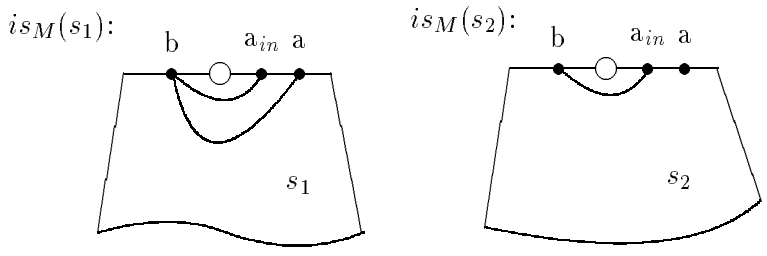

Figure 8: is–graphs of s_1 and s_2.

In Figure 9 the extended dependency graph of (σ, s_1, s_2) is shown.

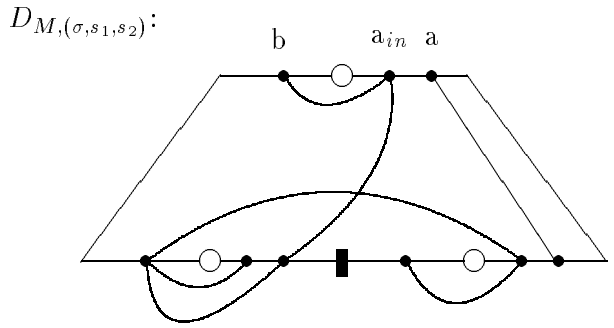

Figure 9: Extended dependency graph of (σ, s_1, s_2).

Now we can use the notion of collection of is–graphs to construct a circularity test for mat tree transducers that is based on the following lemma. We formulate the lemma in such a way that it yields directly the idea of the algorithm; more precisely, every universal quantification is taken over a finite set; in particular, the lemma does not refer to the infinite set of control trees (as Definition 3.20 does).

Lemma 3.21 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer. M is circular iff there is a $\sigma \in \Sigma_+^{(k)}$ and there are $is_1, \ldots, is_k \in is\text{–}set_M$, such that the graph $D_M(\sigma, is_1, \ldots, is_k)$ has a cycle where $D_M(\sigma, is_1, \ldots, is_k)$ is the graph, which results from $D_M(\sigma)$ by adding an arc $((c, zi), (c', zi))$ for every arc (c, c') in is_i (with $i \in [k]$).

Proof: If M is circular, then there is a minimal tree $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ with $k \geq 1$ (minimal in the sense of minimal height), a function $c \in F$, and an $i \in [k]$, such that the function occurrence (c, i) is located on a cycle in the extended dependency graph $D_{M,(\sigma,s_1,\ldots,s_k)}$. Since $is\text{–}set_M$ is the collection of the is–graphs of all control trees, $is\text{–}set_M$ also contains $is_M(s_1), \ldots, is_M(s_k)$. Clearly, $D_{M,(\sigma,s_1,\ldots,s_k)} = D_M(\sigma, is_M(s_1), \ldots, is_M(s_k))$. The other direction of the equivalence is trivially true, because the is–graphs describe the dependencies between functions at the root of subgraphs of σ. □

Thus we can apply an algorithm, which constructs $is\text{–}set_M$ for an mat tree transducer M step by step and which checks simultaneously the condition of the previous lemma. For this purpose, we modify the circularity test for attribute grammars in [ASU86] (cf. page 335) and we use a Modula2–like notation for its presentation. For every grammar symbol X, the set $\mathcal{F}(X)$ in [ASU86] corresponds to the collection of all is–graphs for X. In contrast to attribute grammars, we have no distinction between trees of different types, namely trees with different nonterminal symbols at the root. Thus, in our case it is possible to use only one collection $is\text{–}set_M$ of is–graphs for M.

Algorithm 3.22 (Circularity test for mat tree transducers)
Input: Mat tree transducer $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ with $\mathcal{F} = (F, F_s, F_h)$ and dependency graphs $D_M(\sigma)$ for every $\sigma \in \Sigma_+$.

Output: The answer "M is circular" or "M is noncircular".
Method: The algorithm needs the following variables:

$is\text{–}set_M$	collection of is–graphs of M
is	new constructed is–graph
D	new constructed extended dependency graph
$change$	information, whether $is\text{–}set_M$ has changed during the last run through the *repeat*–loop

Then the algorithm works as follows:

$is\text{–}set_M := \emptyset$;
repeat
  $change := false$;
  for every $\sigma \in \Sigma_+^{(k)}$ *do*
    for every k–tuple $(is_1, \ldots, is_k)$ with $is_j \in is\text{–}set_M$ *do*
      $D := D_M(\sigma)$;
      for every $i \in [k]$ *do*
        for every arc (c, c') in is_i *do*
          add an arc $((c, zi), (c', zi))$ to D
        end
      end;
      if there is a cycle in D *then*
        write("M is circular");
        stop
      else (* there is no cycle in D *)
        $is := (F, \emptyset)$;
        for every path in D from (c, z) to (c', z) *do*
          add an arc (c, c') to is
        end; (* for *)
        if $is \notin is\text{–}set_M$ *then*
          $is\text{–}set_M := is\text{–}set_M \cup \{is\}$;
          $change := true$
        end (* if *)
      end (* if *)
    end (* for *)
  end (* for *)
until $change = false$;
write("M is noncircular")

□

Clearly, the algorithm terminates, since $is\text{–}set_M$ is a finite set. The algorithm discovers every is–graph of M, because for every $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ the graph $is_M(s)$ with $height(s) = l$ is calculated in the l–th run through the repeat–loop out of $D_M(\sigma)$ and $is_M(s_1), \ldots, is_M(s_k)$, or it is even calculated earlier. Thus we can conclude with the following observation concerning the correctness of the circularity test.

Observation 3.23 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer. Algorithm 3.22 with input M and dependency graphs $D_M(\sigma)$ for every $\sigma \in \Sigma_+$ yields the answer

- "M is circular", iff M is circular,
- "M is noncircular", iff M is noncircular. □

3.4 Inductive characterization of computed functions

For every mat tree transducer M, we inductively define a function which characterizes the tree transformation computed by M. (Note that we only consider noncircular mat tree transducers.) More precisely, without using the derivation relation, we will construct functions, denoted by M_s, such that for every synthesized function a of rank $p+1$ and every control tree s the following equation holds:

$$M_s(a(\varepsilon, y_1, \ldots, y_p)) = nf(\Rightarrow_{M,s}, a(\varepsilon, y_1, \ldots, y_p)).$$

In the derivations based on $\Rightarrow_{M,s}$, the parameters $y_1, \ldots, y_p$ are considered as additional working symbols of rank 0; thus, we call the functions M_s a *symbolic inductive characterization* of the derivation relation of M. Clearly, the tree $M_s(a(\varepsilon, y_1, \ldots, y_p)) \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$. (Recall from Definition 3.7 that Y is a subset of $SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$.) The definition of M_s and the inductive characterization of the tree transformation computed by an mat tree transducer are generalizations of the corresponding notions for macro tree transducers [EV85].

For the definition of the functions M_s we need an order in which, for an extended dependency graph $D_{M,(\sigma,s_1,\ldots,s_k)}$, the function occurrences of $in(\sigma)$ can be evaluated. Clearly, since the mat tree transducer is noncircular, such an order exists, and it can be constructed by topologically sorting the function occurrences by obeying the dependencies given in $D_{M,(\sigma,s_1,\ldots,s_k)}$. But this ordering is not unique, because independent occurrences of inherited functions may be ordered arbitrarily. In order to be able to refer in the definition of the functions M_s to one particular ordering of $in(\sigma)$, we assume that there is a total ordering $<$ of F which has to be used in the case of independent function occurrences. Although the definition of the functions M_s depends on one particular ordering, one can observe that the functions M_s themselves do not depend on $<$.

Definition 3.24 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ and $\mu = (\sigma, s_1, \ldots, s_k)$. Moreover, let $<$ be a total ordering of F.

The *topological sorting of the inside function occurrences of* $D_{M,\mu}$ *with respect to* $<$, denoted by $topsort(M, <, \mu)$, is the string over $in(\sigma)$ obtained by the following two steps:

- First, topologically sort the vertices of $D_{M,\mu}$ and if there is a choice in order, then sort depth–first left to right (concerning the tree $\sigma(s_1,\ldots,s_k)$), and if there is still a choice, then obey the total order $<$. This yields a unique string w over $fun(\sigma)$.
- Second, drop all elements of $out(\sigma)$ from w. This yields $topsort(M,<,\mu)$. □

Now we proceed with the definition of M_s with respect to a particular total ordering $<$ of functions. We use in this definition the principle of definition by simultaneous induction, as it is stated in Section 2. Here we instantiate this principle as follows:

f corresponds to the function

$$M_< : T\langle\Sigma\rangle \longrightarrow A$$

where $A = \{\phi | \phi : \tilde{F}_s \longrightarrow SF(\emptyset, F_h, \{\varepsilon\}, \Delta)\}$ and $\tilde{F}_s = \{a(\varepsilon, y_1, \ldots, y_p) | a \in F_s^{(p+1)}, p \geq 0\}$. Instead of $M_<(s)(a(\varepsilon, y_1, \ldots, y_p))$ we write $M_{<,s}(a(\varepsilon, y_1, \ldots, y_p))$.

For $k \geq 0$ and $\sigma \in \Sigma^{(k)}$, g_σ corresponds to the function

$$M_{<,\sigma} : (T\langle\Sigma\rangle)^k \longrightarrow B_\sigma$$

where $B_\sigma = \{\phi | \phi : sub(rhs(\sigma)) \longrightarrow SF(\emptyset, F_h, \{\varepsilon\}, \Delta)\}$.
Instead of $M_{<,\sigma}((s_1,\ldots,s_k))(t)$ we write $M_{<,(\sigma,s_1,\ldots,s_k)}(t)$.

Definition 3.25 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer and let $<$ be a total ordering on F.

1. For every $s \in T\langle\Sigma\rangle \cup \{root(s') \,|\, s' \in T\langle\Sigma\rangle\}$ with $s = \sigma(s_1,\ldots,s_k)$ and $k \geq 0$ and for every $a \in F_s^{(p+1)}$ with $p \geq 0$, define the tree $M_{<,s}(a(\varepsilon, y_1, \ldots, y_p)) \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$ by

$$M_{<,s}(a(\varepsilon, y_1, \ldots, y_p)) = M_{<,(\sigma,s_1,\ldots,s_k)}(rhs(a, z, \sigma)).$$

2. For every $s \in T\langle\Sigma\rangle \cup \{root(s') \,|\, s' \in T\langle\Sigma\rangle\}$ with $s = \sigma(s_1,\ldots,s_k)$, $k \geq 0$ and $\mu = (\sigma, s_1, \ldots, s_k)$, define the set $\{M_{<,\mu}(rhs(w(\nu), \sigma)) \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta) \mid \nu$ is a position in $w = topsort(M, <, \mu)\}$ by (mathematical) induction on ν.

   The tree $M_{<,\mu}(rhs(w(\nu), \sigma)) \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$ itself is defined by structural induction on $rhs(w(\nu), \sigma)$ where we abbreviate $sub(rhs(w(\nu), \sigma))$ by sub.

   (a) If $y_j \in sub$, then $M_{<,\mu}(y_j) = y_j$.

   (b) If $\delta(\zeta_1,\ldots,\zeta_m) \in sub$ with $\delta \in \Delta^{(m)}$ and $m \geq 0$, then $M_{<,\mu}(\delta(\zeta_1,\ldots,\zeta_m)) = \delta(M_{<,\mu}(\zeta_1),\ldots,M_{<,\mu}(\zeta_m))$.

   (c) If $d(zi, \zeta_1,\ldots,\zeta_m) \in sub$ with $d \in F_s^{(m+1)}$, $m \geq 0$, and $i \in [k]$, then $M_{<,\mu}(d(zi, \tilde{\zeta})) = \phi_{\mu,i,(\tilde{\zeta})}(M_{<,s_i}(d(\varepsilon, y_1, \ldots, y_m)))$
   where $(\tilde{\zeta})$ abbreviates $(\zeta_1,\ldots,\zeta_m)$ and $\phi_{\mu,i,(\tilde{\zeta})}$ is defined by structural induction on trees in $SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$ as follows where $SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$ is abbreviated by SF:

i. For every $j \in [m]$, $\phi_{\mu,i,(\tilde{\zeta})}(y_j) = M_{<,\mu}(\zeta_j)$.

ii. For every $\delta \in \Delta^{(q)}$ and $\psi_1, \ldots, \psi_q \in SF$,
$\phi_{\mu,i,(\tilde{\zeta})}(\delta(\psi_1, \ldots, \psi_q)) = \delta(\phi_{\mu,i,(\tilde{\zeta})}(\psi_1), \ldots, \phi_{\mu,i,(\tilde{\zeta})}(\psi_q))$.

iii. For every $b \in F_h^{(q+1)}$ with $q \geq 0$ and $\psi_1, \ldots, \psi_q \in SF$,
$\phi_{\mu,i,(\tilde{\zeta})}(b(\varepsilon, \psi_1, \ldots, \psi_q)) = M_{<,\mu}(rhs(b, zi, \sigma))[y_j/\ \phi_{\mu,i,(\tilde{\zeta})}(\psi_j); j \in [q]]$.

(d) If $b(z, \zeta_1, \ldots, \zeta_m) \in sub$ with $b \in F_h^{(m+1)}$ and $m \geq 0$, then
$M_{<,\mu}(b(z, \zeta_1, \ldots, \zeta_m)) = b(\varepsilon, M_{<,\mu}(\zeta_1), \ldots, M_{<,\mu}(\zeta_m))$. □

Since M is noncircular, $M_{<,\mu}(\zeta)$ is well–defined. In particular, one should observe that the evaluation of $M_{<,\mu}(d(zi, \tilde{\zeta}))$ never refers to itself. Let us give an example to illustrate the definition.

Example 3.26 Consider the mat tree transducer of Example 3.4 and consider the control tree $s = \sigma(\alpha, \alpha)$. In this example, a total ordering of the functions is superfluous, because the topological sorting of the inside function occurrences of $D_{M,(\sigma,\alpha,\alpha)}$ and $D_{M,(\alpha)}$ yields the strings $(b, z1)(b, z2)(a, z)$ and (a, z), respectively, without any choice. Thus, we drop the denotation $<$ from $M_{<,s}$ and $M_{<,\mu}$.

First we want to compute the tree $M_\alpha(a(\varepsilon, y_1))$. According to part 1 of the definition,

$$\begin{aligned} & M_\alpha(a(\varepsilon, y_1)) \\ = \ & M_{(\alpha)}(rhs(a, z, \alpha)) \\ = \ & M_{(\alpha)}(b(z, (b(z, y_1))). \end{aligned}$$

Since $topsort(M, <, (\alpha)) = (a, z)$, we can calculate the tree $M_{(\alpha)}(b(z, (b(z, y_1)))$ immediately as follows:

$$\begin{aligned} & M_{(\alpha)}(b(z, (b(z, y_1))) \\ = \ & b(\varepsilon, M_{(\alpha)}(b(z, y_1))) \\ = \ & b(\varepsilon, b(\varepsilon, M_{(\alpha)}(y_1))) \\ = \ & b(\varepsilon, b(\varepsilon, y_1)) \end{aligned}$$

Now let us compute $M_{\sigma(\alpha,\alpha)}(a(\varepsilon, y_1))$:

$$\begin{aligned} & M_{\sigma(\alpha,\alpha)}(a(\varepsilon, y_1)) \\ = \ & M_{(\sigma,\alpha,\alpha)}(rhs(a, z, \sigma)) \\ = \ & M_{(\sigma,\alpha,\alpha)}(a(z2, a(z2, y_1))) \end{aligned}$$

Since $topsort(M, <, (\sigma, \alpha, \alpha)) = (b, z1)(b, z2)(a, z)$, we compute three trees in the following order:

1. $M_{(\sigma,\alpha,\alpha)}(rhs(b, z1, \sigma))$

2. $M_{(\sigma,\alpha,\alpha)}(rhs(b,z2,\sigma))$

3. $M_{(\sigma,\alpha,\alpha)}(rhs(a,z,\sigma))$

$$
\begin{aligned}
& M_{(\sigma,\alpha,\alpha)}(rhs(b,z1,\sigma)) \\
= \; & M_{(\sigma,\alpha,\alpha)}(b(z,b(z,y_1))) \\
= \; & b(\varepsilon, M_{(\sigma,\alpha,\alpha)}(b(z,y_1))) \\
= \; & b(\varepsilon,b(\varepsilon, M_{(\sigma,\alpha,\alpha)}(y_1))) \\
= \; & b(\varepsilon,b(\varepsilon,y_1))
\end{aligned}
$$

In the following we will abbreviate a tree of the form $b(\varepsilon,\ldots b(\varepsilon,t)\ldots)$ with k–times $b(\varepsilon,$ and for some arbitrary t by $[b(\varepsilon,]^k \;\; t \;\; [)]^k$

$$
\begin{aligned}
& M_{(\sigma,\alpha,\alpha)}(rhs(b,z2,\sigma)) \\
= \; & M_{(\sigma,\alpha,\alpha)}(a(z1,a(z1,y_1))) \\
= \; & \phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(M_\alpha(a(\varepsilon,y_1))) \\
= \; & \phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(b(\varepsilon,b(\varepsilon,y_1))) \\
= \; & M_{(\sigma,\alpha,\alpha)}(rhs(b,z1,\sigma))[y_1/\phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(b(\varepsilon,y_1))] \\
= \; & b(\varepsilon,b(\varepsilon,y_1))[y_1/\phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(b(\varepsilon,y_1))] \\
= \; & b(\varepsilon,b(\varepsilon,\phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(b(\varepsilon,y_1)))) \\
= \; & b(\varepsilon,b(\varepsilon,b(\varepsilon,b(\varepsilon,\phi_{(\sigma,\alpha,\alpha),1,(a(z1,y_1))}(y_1))))) \\
= \; & b(\varepsilon,b(\varepsilon,b(\varepsilon,b(\varepsilon,M_{(\sigma,\alpha,\alpha)}(a(z1,y_1)))))) \\
= \; & b(\varepsilon,b(\varepsilon,b(\varepsilon,b(\varepsilon,\phi_{(\sigma,\alpha,\alpha),1,(y_1)}(M_\alpha(a(\varepsilon,y_1))))))) \\
= \; & b(\varepsilon,b(\varepsilon,b(\varepsilon,b(\varepsilon,\phi_{(\sigma,\alpha,\alpha),1,(y_1)}(b(\varepsilon,b(\varepsilon,y_1))))))) \\
= \; & [b(\varepsilon,]^8 \;\; y_1 \;\; [)]^8
\end{aligned}
$$

It is not very difficult, but tedious to see that $M_{(\sigma,\alpha,\alpha)}(rhs(a,z,\sigma))$ is the tree

$$[b(\varepsilon,]^{32} \;\; y_1 \;\; [)]^{32}.$$

This corresponds to the statement of Theorem 4.2 in the next section. □

In general, one can observe that the value of $M_{<,s}(a(\varepsilon,y_1,\ldots,y_p))$ does not depend on the ordering $<$ of the set of functions:

Observation 3.27 Let $M=(\mathcal{F},\Delta,\Sigma,a_{in},root,R)$ be an mat tree transducer and let $<_1$ and $<_2$ be total orderings on F.

Then, for every $a \in F_s^{(p+1)}$ with $p \geq 0$ and every $s \in T\langle\Sigma\rangle$,

$$M_{<_1,s}(a(\varepsilon,y_1,\ldots,y_p)) = M_{<_2,s}(a(\varepsilon,y_1,\ldots,y_p)).$$

□

Thus in the rest of this paper, we drop the index $<$ from the denotations $M_{<,s}$ and $M_{<,\mu}$, and use an explicit total ordering only when this is necessary. We also use the notation $topsort(M, \mu)$ instead of $topsort(M, <, \mu)$, assuming, that one arbitrary, but fixed ordering $<$ of F is used, whenever it is needed.

We note that, if M is a macro tree transducer, i.e., an mat tree transducer without inherited functions, then this definition coincides with the symbolic inductive characterization provided in Definition 3.18 of [EV85] for macro tree transducers. If M is a full attributed tree transducer, i.e., an mat tree transducer, in which every function has rank 1, then the definition is closely related to the evaluation of noncircular attribute grammars by means of macro tree transducers [Fra82] (cf. P7 in Section 4.2 of [Eng84]). There the value of every synthesized attribute at the root rt of a subtree t is described as a function in terms of the inherited attributes at rt. The problem of constructing the corresponding right–hand sides of rewrite rules was, roughly speaking, solved by pasting together alternatingly right–hand sides of semantic rules of inherited attributes of immediate sons of rt and recursive calls $a(zi)$ of a synthesized attribute a to some i–th subtree of the control tree. By remembering the set of pairs (a, i) which have already occurred during this construction, one can stop building up the right–hand side whenever one pair occurred already before, because this situation can never appear in a noncircular attribute grammar. Here we do not have to remember the set of pairs (a, i) as a termination criterion: since we consider some arbitrary but fixed control tree s, we know the dependencies between function occurrences at the root of every subtree $\sigma(s_1, \ldots, s_k)$ of s. Thus we can construct the tree $M_{\sigma(s_1,\ldots,s_k)}(a(\varepsilon, y_1, \ldots, y_p))$ on the basis of the finite string which results from topological sorting of $D_{M,(\sigma,s_1,\ldots,s_k)}$.

Next we prove that the symbolic inductive characterization of an mat tree transducer correctly represents its derivation relation.

Theorem 3.28 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer. Then, for every $s \in T\langle\Sigma\rangle$,

$$\tau(M)(s) = M_{root(s)}(a_{in}(\varepsilon)).$$

Proof: To prove the statement of the theorem it suffices to prove the following predicate P and, for every $k \geq 0$ and $\sigma \in \Sigma_+^{(k)}$, the predicate Q_σ by the principle of proof by simultaneous induction as it is stated in Section 2:

(P) For every $s \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$, $P(s)$ is true, iff for every $a \in F_s^{(p+1)}$ with $p \geq 0$:

$$M_s(a(\varepsilon, y_1, \ldots, y_p)) = nf(\Rightarrow_s, a(\varepsilon, y_1, \ldots, y_p)).$$

(Q_σ) For every $k \geq 0$, $\sigma \in \Sigma_+^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, $Q_\sigma((s_1, \ldots, s_k))$ is true, iff for every position ν of $w = topsort(M, (\sigma, s_1, \ldots, s_k))$:

$$M_{(\sigma,s_1,\ldots,s_k)}(rhs(w(\nu), \sigma)) = nf(\Rightarrow_{\sigma(s_1,\ldots,s_k)}, rhs(w(\nu), \sigma)[z/\varepsilon]).$$

We do not work out the details of the proof here (see [KV92]), but we mention that the following statement has to be proved by structural induction, in order to show Condition 2. of the principle of proof by simultaneous induction (cf. Section 2):

For every $sub \in sub(rhs(w(\nu), \sigma))$,

$$M_{(\sigma,s_1,\ldots,s_k)}(sub) = nf(\Rightarrow_{\sigma(s_1,\ldots,s_k)}, sub[z/\varepsilon]).$$

In the case $sub = d(zi, \zeta_1, \ldots, \zeta_m)$ for some synthesized function $d \in F_s^{(m+1)}$ of this structural induction, we need a further statement that also has to be proved by structural induction:

For every $t \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta \cup Y_m)$,

$$\phi_{(\sigma,s_1,\ldots,s_k),i,(\zeta_1,\ldots,\zeta_m)}(t) = nf(\Rightarrow_{\sigma(s_1,\ldots,s_k)}, \theta(t))[y_\alpha / nf(\Rightarrow_{\sigma(s_1,\ldots,s_k)}, \zeta_\alpha[z/\varepsilon]); \alpha \in [m]],$$

where $\theta(t)$ is obtained from t by replacing the local path ε by the local path i. □

3.5 Elementary properties

In this subsection we want to prove two properties concerning mat tree transducers. First, we show that obeying a particular restriction of the $(a, z, root)$–rules for any synthesized function a does not decrease the computational power. Second, we will see that some circular mat tree transducers can be transformed 'equivalently' into noncircular ones using a pruning mechanism on right–hand sides of rules.

3.5.1 Mat tree transducers with initial chain rules

Here we present a slight syntactic restriction of mat tree transducers which does not decrease their transformational power. Using this restricted form, some of the constructions in later sections become easier; moreover, we can get closer to other tree transducers which are known from the literature, by considering the restricted version. The restriction concerns the $(a, z, root)$–rules for every synthesized function $a \in F_s^{(p+1)}$: they must have the form $a(z, y_1, \ldots, y_p) \to a(z1, y_1, \ldots, y_p)$. We call such a tree transducer an *mat tree transducer with initial chain rules*.

Definition 3.29 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer with $\mathcal{F} = (F, F_s, F_h)$. M is called *mat tree transducer with initial chain rules* if for every $a \in F_s^{(p+1)}$, the right–hand side of the $(a, z, root)$–rule has the form $a(z1, y_1, \ldots, y_p)$. □

In a similar way we define the notion of *with initial chain rules* also for s_f–tree transducers, si–tree transducers, and s–tree transducers. We denote the classes of tree transformations computed by mat tree transducers (and s_f–tree transducers, si–tree transducers, and s–tree transducers) with initial chain rules by $S_f I_f T_{icr}$ (and $S_f T_{icr}$, SIT_{icr} and ST_{icr}, respectively).

We show that this restriction does not decrease the computational power of mat tree transducers. Roughly speaking, if we want to construct an mat tree transducer with

initial chain rules for a usual mat tree transducer, then we have to postpone one step of the calculation that would have been done by M at the root. Therefore we have to enrich other rules.

Theorem 3.30 $S_fI_fT \subseteq S_fI_fT_{icr}$, $S_fT \subseteq S_fT_{icr}$, $SIT \subseteq SIT_{icr}$, and $ST \subseteq ST_{icr}$

Proof: For a given mat tree transducer M we construct an mat tree transducer with initial chain rules M' which computes the same tree transformation as M. The construction for the other three statements of the theorem can be easily derived from the construction for $S_fI_fT \subseteq S_fI_fT_{icr}$, and hence they are not shown here.

Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer with $\mathcal{F} = (F, F_s, F_h)$. We construct $M' = (\mathcal{F}', \Delta, \Sigma, a'_{in}, root, R')$ as follows, where, roughly speaking, a new (a'_{in}, z, σ)–rule combines an old $(a_{in}, z, root)$–rule and those (a, z, σ)–rules, for which $a(z1, \ldots)$ appears in $rhs_M(a_{in}, z, root)$ (cf. Figure 10 for an example).

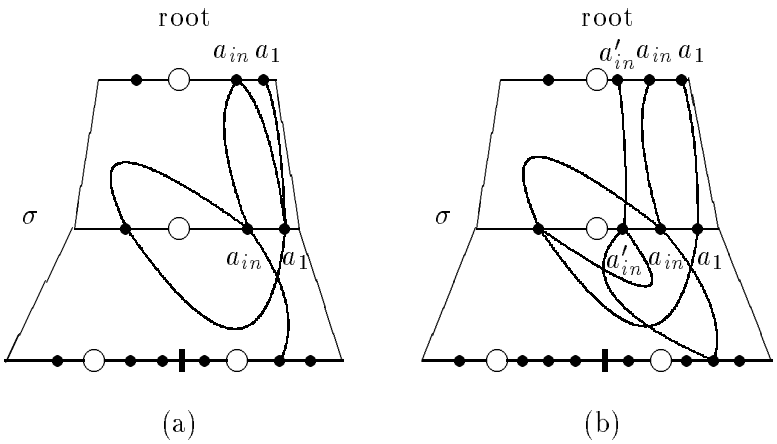

Figure 10: (a) Dependency graphs of M, (b) Dependency graphs of M'.

$\mathcal{F}' = (F', F'_s, F'_h)$ with $F'_s = F_s \cup \{a_{in}^{'(1)}\}$ and $F'_h = F_h$.

$R' = \bigcup_{\sigma \in \Sigma_+} R'_\sigma$ where

1. For every $a \in F_s^{'(p+1)}$, the set R'_{root} contains the rule

   $$a(z, y_1, \ldots, y_p) \to a(z1, y_1, \ldots, y_p)$$

   and for every $b \in F'_h$ the set R'_{root} contains the $(b, z1, root)$–rule of R_{root}.

2. For every $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, the set R'_σ contains the rules of R_σ and a new rule

   $$a'_{in}(z) \to \rho_\sigma(rhs_M(a_{in}, z, root))$$

   where $\rho_\sigma : RHS(F_s, \emptyset, \Delta, Y_0, root) \longrightarrow RHS(F_s, F_h, \Delta, Y_0, \sigma)$ is defined as follows:

$$\begin{aligned}
\rho_\sigma(a(z1,\zeta_1,\ldots,\zeta_p)) &:= rhs_M(a,z,\sigma)[y_i/\rho_\sigma(\zeta_i)\,;\, i\in[p]] \\
&\qquad \text{for every } a\in F_s^{(p+1)} \\
&\qquad \text{and } \zeta_1,\ldots,\zeta_p \in RHS(F_s,\emptyset,\Delta,Y_0,root) \\
\rho_\sigma(\delta(\zeta_1,\ldots,\zeta_p)) &:= \delta(\rho_\sigma(\zeta_1),\ldots,\rho_\sigma(\zeta_p)) \\
&\qquad \text{for every } \delta\in\Delta^{(p)} \\
&\qquad \text{and } \zeta_1,\ldots,\zeta_p \in RHS(F_s,\emptyset,\Delta,Y_0,root)
\end{aligned}$$

Now we prove that for every control tree $s\in T\langle\Sigma\rangle$, the following equation holds:

$$M'_{root(s)}(a'_{in}(\varepsilon)) = M_{root(s)}(a_{in}(\varepsilon)).$$

Note that

$$(*)\qquad M'_s(a(\varepsilon,y_1,\ldots,y_p)) = M_s(a(\varepsilon,y_1,\ldots,y_p))$$

holds for every $a\in F_s^{(p+1)}$, because in the construction of M' only the $(a,z,root)$–rules have changed and the function a'_{in} cannot be called in both expressions.

Let $s=\sigma(s_1,\ldots,s_k)$ with $\sigma\in\Sigma^{(k)}$, $k\geq 0$ and $s_i\in T\langle\Sigma\rangle$ for every $i\in[k]$:

$$\begin{aligned}
& M'_{root(s)}(a'_{in}(\varepsilon)) & \\
= \;& M'_{(root,s)}(rhs_{M'}(a'_{in},z,root)) & \text{(Def. of } M'_{root(s)}) \\
= \;& M'_{(root,s)}(a'_{in}(z1)) & \text{(initial chain rule)} \\
= \;& \phi'_{(root,s),1,()}(M'_s(a'_{in}(\varepsilon))) & \text{(Def. of } M'_{(root,s)}) \\
= \;& \phi'_{(root,s),1,()}(M'_{(\sigma,s_1,\ldots,s_k)}(rhs_{M'}(a'_{in},z,\sigma))) & \text{(Def. of } M'_s) \\
= \;& \phi'_{(root,s),1,()}(M'_{(\sigma,s_1,\ldots,s_k)}(\rho_\sigma(rhs_M(a_{in},z,root)))) & \text{(Def. of } rhs_{M'}(a'_{in},z,\sigma)) \\
= \;& M_{(root,s)}(rhs_M(a_{in},z,root)) & \text{(to prove (see below))} \\
= \;& M_{root(s)}(a_{in}(\varepsilon)) & \text{(Def. of } M_{root(s)})
\end{aligned}$$

Now it is left to prove that

$$\phi'_{(root,s),1,()}(M'_{(\sigma,s_1,\ldots,s_k)}(\rho_\sigma(t))) = M_{(root,s)}(t)$$

for every $t\in sub(rhs_M(a_{in},z,root))\subseteq RHS(F_s,\emptyset,\Delta,Y_0,root)$ by structural induction. This induction is not shown here. □

This theorem shows that s_f–tree transducers, si–tree transducers, and s–tree transducers really compute the same class of tree transformations as macro tree transducers, full attributed tree transducers (see the discussion in Observation 3.6 (3.)), and top–down tree transducers. Thus, in the sequel, we will also use the notions which are known from the literature.

3.5.2 Pruned tree transducers

We have proved that noncircularity is a sufficient condition for the uniqueness of normal-forms. But clearly, there are mat tree transducers which are circular but still compute a

total tree function. This is due to the possibility that, roughly speaking, the circularity can occur in the parameter position of a function and eventually, this position is deleted.

Example 3.31 Consider the mat tree transducer M with synthesized function symbols a_{in} and a each of rank 1, and with one inherited function symbol b of rank 2. The ranked alphabet Σ of input symbols is the set $\{\sigma^{(1)}, \alpha^{(0)}\}$ which is also the set Δ of working symbols. M contains the following set $R = R_{root} \cup R_\sigma \cup R_\alpha$ of rules:

$$
\begin{array}{lllll}
R_{root}: & a_{in}(z) & \rightarrow & a_{in}(z1) \\
 & a(z) & \rightarrow & a(z1) \\
 & b(z1, y_1) & \rightarrow & \alpha \\
R_\sigma: & a_{in}(z) & \rightarrow & b(z, a(z1)) \\
 & a(z) & \rightarrow & a(z1) \\
 & b(z1, y_1) & \rightarrow & a(z1) \\
R_\alpha: & a_{in}(z) & \rightarrow & \alpha \\
 & a(z) & \rightarrow & b(z, \alpha)
\end{array}
$$

Then M can derive as follows on the control tree $root(\sigma(\alpha))$:

$$
\begin{array}{ll}
 & a_{in}(\varepsilon) \\
\Rightarrow & a_{in}(1) \\
\Rightarrow & b(1, a(11)) \\
\Rightarrow & b(1, b(11, \alpha)) \\
\Rightarrow & b(1, a(11)) \\
\Rightarrow & \ldots
\end{array}
$$

Thus, M is circular, but for every $\tilde{s} \in \{root(s) | s \in T\langle\Sigma\rangle\}$, $nf(\Rightarrow, a_{in}(\varepsilon)) = \alpha$. To see this one first has to observe that either $nf(\Rightarrow, a_{in}(\varepsilon))$ is undefined or $nf(\Rightarrow, a_{in}(\varepsilon)) = \alpha$, because the rules cannot construct any other output tree than α. But also, for every $n \geq 0$,

$$a_{in}(\varepsilon) \Rightarrow_{M, root(\sigma^n(\alpha))} a_{in}(1) \Rightarrow_{M, root(\sigma^n(\alpha))} b(1, a(11)) \Rightarrow_{M, root(\sigma^n(\alpha))} \alpha$$

by rewriting the function occurrence $(b, 1)$ with the rule from R_{root}. Hence, $nf(\Rightarrow, a_{in}(\varepsilon)) = \alpha$ for every control tree s. □

These circularities in mat tree transducers which result from the parameters of some functions, can sometimes be eliminated by pruning the argument trees which are not used in a derivation like in our example above. This method is also interesting concerning mat tree transducers which are not circular, because they can become easier by pruning arguments. Roughly speaking, a tree in a parameter position j of a function c is pruned, if y_j does not occur in any right–hand side of c.

Definition 3.32 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer with $\mathcal{F} = (F, F_s, F_h)$ and let $\alpha \in \Delta^{(0)}$.

The *pruned mat tree transducer* $M' = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R')$ *of* M is defined in two steps as follows:

1.) We define a function $notused : F \longrightarrow \mathcal{P}(Y)$ which delivers, for every function $f \in F^{(p+1)}$, the set of those variables of Y_p which occur in no f–rule:

$$\begin{array}{lcll} notused(a) & := & \{y_i \in Y_p \,|\, y_i \notin \bigcup_{\sigma\in\Sigma_+} sub(rhs(a,z,\sigma))\} & \text{if } a \in F_s^{(p+1)} \\ notused(b) & := & \{y_i \in Y_p \,|\, y_i \notin \bigcup_{\sigma\in\Sigma_+} \bigcup_{j\in[rank(\sigma)]} sub(rhs(b,zj,\sigma))\} & \text{if } b \in F_h^{(p+1)} \end{array}$$

2.) We construct the set R' by transforming the rules of R_σ into rules of R'_σ for every $\sigma \in \Sigma_+$ using the simultaneously defined functions $\underline{trans}_\sigma$ and $\underline{prune}_\sigma$. For every rule

$$f(\eta, y_1, \ldots, y_p) \to \zeta$$

in R_σ with $\sigma \in \Sigma_+$, $f \in F^{(p+1)}$, $\eta \in \{z\} \cup \{zj \,|\, j \in [rank(\sigma)]\}$ and $\zeta \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$ there is a rule

$$f(\eta, y_1, \ldots, y_p) \to \underline{trans}_\sigma(\zeta, notused)$$

in R'_σ, where the function

$$\underline{trans}_\sigma : RHS(F_s, F_h, \Delta, Y_p, \sigma) \times \{nu \,|\, nu : F \longrightarrow \mathcal{P}(Y)\} \longrightarrow RHS(F_s, F_h, \Delta, Y_p, \sigma)$$

is defined as follows:

$$\begin{array}{lcl} \underline{trans}_\sigma(a(zj, t_1, \ldots, t_n), nu) & := & a(zj, \underline{prune}_\sigma(t_1, a, 1, nu), \ldots, \underline{prune}_\sigma(t_n, a, p, nu)) \\ & & \text{if } a \in F_s^{(n+1)} \text{ and } t_1, \ldots, t_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma) \\ \underline{trans}_\sigma(b(z, t_1, \ldots, t_n), nu) & := & b(z, \underline{prune}_\sigma(t_1, b, 1, nu), \ldots, \underline{prune}_\sigma(t_n, b, p, nu)) \\ & & \text{if } b \in F_h^{(n+1)} \text{ and } t_1, \ldots, t_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma) \\ \underline{trans}_\sigma(\delta(t_1, \ldots, t_n), nu) & := & \delta(\underline{trans}_\sigma(t_1, nu), \ldots, \underline{trans}_\sigma(t_n, nu)) \\ & & \text{if } \delta \in \Delta^{(n)} \text{ and } t_1, \ldots, t_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma) \\ \underline{trans}_\sigma(y_i, nu) & := & y_i \quad \text{if } y_i \in Y_p \end{array}$$

The function

$$\begin{array}{l} \underline{prune}_\sigma : RHS(F_s, F_h, \Delta, Y_p, \sigma) \times F \times I\!N \times \{nu \,|\, nu : F \longrightarrow \mathcal{P}(Y)\} \\ \qquad \longrightarrow RHS(F_s, F_h, \Delta, Y_p, \sigma) \end{array}$$

does the pruning of subtrees as follows depending on the information of the function nu:

$$\begin{array}{lcll} \underline{prune}_\sigma(t, f, i, nu) & := & \underline{trans}_\sigma(t, nu) & \text{if } y_i \notin nu(f) \\ \underline{prune}_\sigma(t, f, i, nu) & := & \alpha & \text{if } y_i \in nu(f) \end{array}$$

□

We will not give a proof for the fact that an mat tree transducer M computes the same tree transformation as the pruned mat tree transducer of M, and conclude this section with an example:

Example 3.33 Consider the mat tree transducer M of Example 3.31 on page 35. We calculate the pruned mat tree transducer M' of M in two steps:

1.) Because a_{in} and a have no context parameters and because the variable y_1 does not occur in a b–rule, the function *notused* is defined as follows:

$$\begin{aligned} notused(a_{in}) &:= \emptyset \\ notused(a) &:= \emptyset \\ notused(b) &:= \{y_1\} \end{aligned}$$

2.) With the help of *notused* we can transform the rules of M into the rules of M'. In fact only the (a_{in}, z, σ)–rule changes and we show a transformation only for this rule:

$a_{in}(z) \to \underline{trans}_\sigma(b(z, a(z1)), notused)$ is in R_σ, where

$$\begin{aligned} \underline{trans}_\sigma(b(z, a(z1)), notused) &= b(z, \underline{prune}_\sigma(a(z1), b, 1, notused)) \\ &= b(z, \alpha) \qquad (y_1 \in notused(b)) \end{aligned}$$

Then M' can only derive as follows on the control tree $root(\sigma(\alpha))$:

$$a_{in}(\varepsilon) \Rightarrow_{M', root(\sigma(\alpha))} a_{in}(1) \Rightarrow_{M', root(\sigma(\alpha))} b(1, \alpha) \Rightarrow_{M', root(\sigma(\alpha))} \alpha$$

□

4 Formal power of mat tree transducers

In the first subsection of this section we show that mat tree transducers are strictly more powerful than both, full attributed tree transducers and macro tree transducers. In the second subsection we will prove that the computational power of mat tree transducers is the same as the computational power of the twofold composition of full attributed tree transducers, i.e., $S_f I_f T = SIT \circ SIT$ (cf. Theorem 4.8). This characterization leads to an upper bound of the height of output trees of mat tree transducers. In connection with the fact that the composition hierarchy of full attributed tree transducers is strict, the characterization also yields the strictness of the composition hierachy of mat tree transducers. As another corollary of the characterization it follows that mat tree transducers are closed under right composition with top–down tree transducers but not under left composition, i.e., $S_f I_f T \circ ST \subseteq S_f I_f T$ and $ST \circ S_f I_f T \not\subseteq S_f I_f T$ (cf. Corollary 4.15 and Corollary 4.16). In the third subsection we prove that s_f–tree transducers are as powerful as $s_f i$–tree transducers. In words, adding inherited attributes to s_f–tree transducers does not increase their transformational power.

4.1 $S_f I_f T$ strictly contains SIT and $S_f T$

First we recall the lemmata concerning, for macro tree transducers and attributed tree transducers, the upper bound of the height of output trees. Furthermore, we extend the latter one to the corresponding property of full attributed tree transducers.

Lemma 4.1

1. (Lemma 3.3 of [Eng81], Theorem 3.24 of [EV85]) Let M be a macro tree transducer and let $\tau(M)(s) = t$. Then $height(t) \leq 2^{c \cdot height(s)}$ for some constant $c > 0$.
2. (Lemma 4.1 of [Fül81]) Let M be an attributed tree transducer and let $\tau(M)(s) = t$. Then $height(t) \leq c \cdot size(s)$ for some constant $c > 0$.
3. The second statement of this lemma also holds for full attributed tree transducers.

Proof: The proof of the third statement uses the same argumentation as the proof of Lemma 4.1 of [Fül81]. □

Now we construct an mat tree transducer for which the upper bounds of output trees as they are mentioned in the previous lemma, do not hold.

Theorem 4.2 Let $\Delta = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and $\Sigma = \{\sigma, \alpha\}$. There is an mat tree transducer M, such that for every $s \in T\langle\Sigma\rangle$ the property

$$\tau(M)(s) = \gamma^{2^{2size(s)-1}}(\alpha)$$

holds.

Proof: Clearly, the desired tree transducer M is the mat tree transducer $M_{doubleexp}$ of Example 3.4 on page 13.

In order to prove the theorem, we show that, for every $s \in T\langle\Sigma\rangle$, the statement

$$M_{root(s)}(a_{in}(\varepsilon)) = \gamma^{2^{2size(s)-1}}(\alpha)$$

holds.

$$\begin{aligned}
& M_{root(s)}(a_{in}(\varepsilon)) & \\
= \; & M_{(root,s)}(rhs(a_{in}, z, root)) & \text{(Def. of } M_{root(s)}) \\
= \; & M_{(root,s)}(a(z1, \alpha)) & \text{(Def. of } rhs(\ldots)) \\
= \; & \phi_{(root,s),1,(\alpha)}(M_s(a(\varepsilon, y_1))) & \text{(Def. of } M_{(root,s)}) \\
= \; & \phi_{(root,s),1,(\alpha)}(\left[b(\varepsilon,\right]^{2^{2size(s)-1}} y_1 \left[)\right]^{2^{2size(s)-1}}) & \text{(to prove (see below))} \\
= \; & \left[M_{(root,s)}(rhs(b, z1, root))[y_1/\right]^{2^{2size(s)-1}} M_{(root,s)}(\alpha) \left[\right]^{2^{2size(s)-1}} & \\
& & \text{(Def. of } \phi_{(root,s),1,(\alpha)}) \\
= \; & \left[M_{(root,s)}(rhs(b, z1, root))[y_1/\right]^{2^{2size(s)-1}} \alpha \left[\right]^{2^{2size(s)-1}} & \text{(Def. of } M_{(root,s)}) \\
= \; & \left[M_{(root,s)}(\gamma(y_1))[y_1/\right]^{2^{2size(s)-1}} \alpha \left[\right]^{2^{2size(s)-1}} & \text{(Def. of } rhs(\ldots)) \\
= \; & \left[\gamma(y_1)[y_1/\right]^{2^{2size(s)-1}} \alpha \left[\right]^{2^{2size(s)-1}} & \text{(Def. of } M_{(root,s)}) \\
= \; & \gamma^{2^{2size(s)-1}}(\alpha) & ([\ldots])
\end{aligned}$$

Now it is left to prove the following statement

$$M_s(a(\varepsilon, y_1)) = \left[b(\varepsilon,\right]^{2^{2size(s)-1}} y_1 \left[)\right]^{2^{2size(s)-1}}$$

for every $s \in T\langle\Sigma\rangle$ by structural induction on s. We do not show this proof here.

□

In particular, if s is a fully binary tree, i.e., a tree in which every path has the same length, then $size(s) = 2^{height(s)} - 1$ and $\tau(M_{doubleexp})(s) = \gamma^{2^{2^{height(s)+1}-3}}$.

Actually, the growing of heights of output trees as it is shown in Theorem 4.2, is already the upper bound. This is shown in Corollary 4.12.

Theorem 4.3 $SIT \subset S_f T \subset S_f I_f T$.

Proof: By Lemma 4.10, we obtain $SIT \subseteq S_f T$. By Lemma 4.1(3) and Example 4.3 of [EV85], we obtain the strictness of the inclusion $SIT \subset S_f T$. The strict inclusion $S_f T \subset S_f I_f T$ follows from Theorem 4.2 and Lemma 4.1(1). □

4.2 Characterization of mat tree transducers

The characterization of mat tree transducers by the twofold composition of full attributed tree transducers is based on the decomposition $S_f I_f T = SIT \circ YIELD$ which is a gener-

alization of the decomposition result $S_f T = ST \circ YIELD$ for macro tree transducers (cf., e.g., Theorem 3 [Eng80], Proposition 4.19 [CF82], or Corollary 5.9 [EV85]).

Lemma 4.4 $S_f I_f T \subseteq SIT \circ YIELD$ and $S_f T \subseteq ST \circ YIELD$

Proof: For a given mat tree transducer M, we construct a full attributed tree transducer N and a mapping f such that $\tau(M) = \tau(N) \circ YIELD_f$. The second equation follows immediately from the first one, if the inherited functions are dropped from M.

Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an mat tree transducer with system $\mathcal{F} = (F, F_s, F_h)$ of function symbols.
Define $\Delta' = \{\delta'^{(0)} \mid \delta \in \Delta\} \cup \{\pi_j^{(0)} \mid j \in [n]\} \cup \{sub_j^{(j)} \mid j \in [m]\}$
where $n+1 = max\{j \mid F^{(j)} \neq \emptyset\}$ and $m = max\{j \mid F^{(j)} \cup \Delta^{(j-1)} \neq \emptyset\}$.
Define $f : \Delta'^{(0)} \longrightarrow T\langle \Delta \cup Y \rangle$ by
$f(\delta') = \delta(y_1, \ldots, y_k)$ if $\delta \in \Delta^{(k)}$, and by
$f(\pi_j) = y_j$ for every $j \in [n]$.
Moreover, define $F_s' = \{a'^{(1)} \mid a \in F_s\}$ and $F_h' = \{b'^{(1)} \mid b \in F_h\}$.

For the construction of the full attributed tree transducer N, we define the family $\{COMB_{p,\sigma}\}_{p \geq 0, \sigma \in \Sigma_+}$ of mappings

$$COMB_{p,\sigma} : RHS(F_s, F_h, \Delta, Y_p, \sigma) \longrightarrow RHS(F_s', F_h', \Delta', Y_0, \sigma)$$

by structural induction on the set of right–hand sides:

(i) For every $j \in [p]$, $COMB_{p,\sigma}(y_j) = \pi_j$.

(ii) For every $\delta \in \Delta^{(r)}$ with $r \geq 0$ and $\zeta_1, \ldots, \zeta_r \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$,
$COMB_{p,\sigma}(\delta(\zeta_1, \ldots, \zeta_r)) = sub_{r+1}(\delta', COMB_{p,\sigma}(\zeta_1), \ldots, COMB_{p,\sigma}(\zeta_r))$.

(iii) For every $a \in F_s^{(r+1)}$ with $r \geq 0$ and $j \in [rank_\Delta(\sigma)]$, and $\zeta_1, \ldots, \zeta_r \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$,
$COMB_{p,\sigma}(a(zj, \zeta_1, \ldots, \zeta_r)) = sub_{r+1}(a'(zj), COMB_{p,\sigma}(\zeta_1), \ldots, COMB_{p,\sigma}(\zeta_r))$.

(iv) For every $b \in F_h^{(r+1)}$ with $r \geq 0$ and $\zeta_1, \ldots, \zeta_r \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$,
$COMB_{p,\sigma}(b(z, \zeta_1, \ldots, \zeta_r)) = sub_{r+1}(b'(z), COMB_{p,\sigma}(\zeta_1), \ldots, COMB_{p,\sigma}(\zeta_r))$.

Then we define the full attributed tree transducer $N = (\mathcal{F}', \Sigma \cup \Delta', \Sigma, a_{in}', root, R')$ by:

- $\mathcal{F}' = (F', F_s', F_h')$ with $F' = F_s' \cup F_h'$.
- If $c(\eta, y_1, \ldots, y_p) \to \zeta$ is in R_σ, then $c'(\eta) \to COMB_{p,\sigma}(\zeta)$ is in R_σ'.

If $YIELD_f$ is extended in the obvious way to right–hand sides of rules by defining $YIELD_f(c'(\eta)) = c(\eta, y_1, \ldots, y_p)$ for a rule in R_σ', if $c \in F^{(p+1)}$ and $\eta \in \{zj \mid j \in [rank(\sigma)] \cup \{\varepsilon\}\}$, then we can prove

$$(\#) \qquad YIELD_f(COMB_{p,\sigma}(t)) = t$$

for every $t \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$ by structural induction on t. This part of the proof is not worked out here.

In the sequel we use a further extension of $YIELD_f$ to particular sentential forms by defining $YIELD_f(c'(\varepsilon)) = c(\varepsilon, y_1, \ldots, y_p)$ if $c \in F^{(p+1)}$.

Observe that, for every $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ and $\mu = (\sigma, s_1, \ldots, s_k)$, $topsort(M, \mu) = topsort(N, \mu)$ without consideration of the renaming of functions, because the dependency graphs of M cover the worst case. To prove the construction, we show the following predicate P and, for every $k \geq 0$ and $\sigma \in \Sigma_+^{(k)}$, the predicate Q_σ by the principle of proof by simultaneous induction:

(P) For every $s \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$, $P(s)$ is true, iff for every $a \in F_s^{(p+1)}$ with $p \geq 0$:

$$M_s(a(\varepsilon, y_1, \ldots, y_p)) = YIELD_f(N_s(a'(\varepsilon)))$$

(Q_σ) For every $k \geq 0$, $\sigma \in \Sigma_+^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, $Q_\sigma((s_1, \ldots, s_k))$ is true, iff for every position ν of $w = topsort(N, (\sigma, s_1, \ldots, s_k))$:

$$M_{(\sigma,s_1,\ldots,s_k)}(YIELD_f(rhs_N(w(\nu), \sigma))) = YIELD_f(N_{(\sigma,s_1,\ldots,s_k)}(rhs_N(w(\nu), \sigma)))$$

We do not present this proof here, but we mention the statements which are used during this proof and which have to be proved on their part. To show Condition 2. of the principle of proof by simultaneous induction (cf. Section 2), the following statement must be proved by structural induction:

For every $sub \in sub(rhs_N(w(\nu), \sigma))$,

$$M_{(\sigma,s_1,\ldots,s_k)}(YIELD_f(sub)) = YIELD_f(N_{(\sigma,s_1,\ldots,s_k)}(sub)).$$

In the case of this induction, in which $sub = d'(zi)$ for some $d' \in F'_s$ with $rank_F(d) = p+1$, a further statement has to be proved by structural induction:

For every $t \in SF(\emptyset, F'_h, \{\varepsilon\}, \Delta')$,

$$\phi^M_{(\sigma,s_1,\ldots,s_k),i,(y_1,\ldots,y_p)}(YIELD_f(t)) = YIELD_f(\phi^N_{(\sigma,s_1,\ldots,s_k),i,()}(t))$$

□

Example 4.5 Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be the mat tree transducer of Example 3.4 with $F_s = \{a_{in}^{(1)}, a^{(2)}\}$, $F_h = \{b^{(2)}\}$, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ and $\Delta = \Sigma \cup \{\gamma^{(1)}\}$. Here we want to illustrate the transformation of the rule

$$a(z, y_1) \to a(z2, a(z2, y_1))$$

in R_σ of M. We first apply $COMB_{1,\sigma}$ to the right–hand side t of this rule:

$$\begin{aligned} COMB_{1,\sigma}(t) &= COMB_{1,\sigma}(a(z2, a(z2, y_1))) \\ &= sub_2(a'(z2), COMB_{1,\sigma}(a(z2, y_1))) \\ &= sub_2(a'(z2), sub_2(a'(z2), COMB_{1,\sigma}(y_1))) \\ &= sub_2(a'(z2), sub_2(a'(z2), \pi_1)). \end{aligned}$$

Thus, the full attributed tree transducer N has the following rule:

$$a'(z) \to sub_2(a'(z2), sub_2(a'(z2), \pi_1)).$$

The mapping $f : \Delta'^{(0)} \longrightarrow T\langle \Delta \cup Y \rangle$ is defined by

$$\begin{aligned} f(\sigma') &:= \sigma(y_1, y_2) \\ f(\gamma') &:= \gamma(y_1) \\ f(\alpha') &:= \alpha \\ f(\pi_1) &:= y_1. \end{aligned}$$

Finally, we want to check property (#) in the proof of the previous lemma by applying $YIELD_f$ to the tree $COMB_{1,\sigma}(t)$:

$$\begin{aligned} YIELD_f(COMB_{1,\sigma}(t)) &= YIELD_f(sub_2(a'(z2), sub_2(a'(z2), \pi_1))) \\ &= YIELD_f(a'(z2))[y_1/YIELD_f(sub_2(a'(z2), \pi_1))] \\ &= a(z2, y_1)[y_1/YIELD_f(a'(z2))[y_1/YIELD_f(\pi_1)]] \\ &= a(z2, y_1)[y_1/a(z2, y_1)[y_1/y_1]] \\ &= a(z2, a(z2, y_1)) \\ &= t. \end{aligned}$$

□

Lemma 4.6 $SIT \circ YIELD \subseteq S_f I_f T$ and $ST \circ YIELD \subseteq S_f T$

Proof: Let $N = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be a full attributed tree transducer and let $f : \Delta^{(0)} \longrightarrow T\langle \Delta' \cup Z \rangle$ be a mapping for some ranked alphabet Δ'. Here we use variables from Z to define $YIELD_f$ rather than variables from Y, because the latter ones are used as substitution variables in rules. The variables from Z will be treated as additional working symbols in the mat tree transducer, which we have to construct. Then there is an $n \geq 0$ such that $range(f) \subseteq T\langle \Delta' \cup Z_n \rangle$. Note that $\tau(M) \circ YIELD_f$ is a mapping of type $T\langle \Sigma \rangle \longrightarrow T\langle \Delta' \cup Z_n \rangle$.

Construct the mat tree transducer $M = (\mathcal{F}', \Delta' \cup Z_n \cup \Sigma, \Sigma, \tilde{a}_{in}, root, R')$ as follows:

- If $\mathcal{F} = (F, F_s, F_h)$, then $\mathcal{F}' = (F', F'_s, F'_h)$ with $F' = F'_s \cup F'_h$, $F'_s = \{a'^{(n+1)} \mid a \in F_s\} \cup \{\tilde{a}_{in}^{(1)}\}$ and $F'_h = \{b'^{(n+1)} \mid b \in F_h\}$.
- If $c(\eta) \to \zeta$ is a rule in R, then $c'(\eta, y_1, \ldots, y_n) \to YIELD_f(\zeta)[z_i/y_i \,;\, i \in [n]]$ is in R' where $YIELD_f$ is extended by $YIELD_f(c(\eta)) = c'(\eta, z_1, \ldots, z_n)$.

  Moreover, if $a_{in}(z) \to \zeta$ is a rule in R, then also $\tilde{a}_{in}(z) \to YIELD_f(\zeta)$ is in R'.

For every $s = \sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$ and $\mu = (\sigma, s_1, \ldots, s_k)$, $topsort(M, \mu) = topsort(N, \mu)$, because there are no new dependencies between functions after the construction of M. We will use a further extension of $YIELD_f$ to particular sentential forms by defining $YIELD_f(c(\varepsilon)) = c'(\varepsilon, z_1, \ldots, z_n)$ if $c \in F$.

We can show $M_{root(s)}(\tilde{a}_{in}(\varepsilon)) = YIELD_f(N_{root(s)}(a_{in}(\varepsilon)))$, if we prove the following predicate P and, for every $k \geq 0$ and $\sigma \in \Sigma_+^{(k)}$, the predicate Q_σ by the principle of proof by simultaneous induction:

(P) For every $s \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$, $P(s)$ is true, iff
for every $a \in F_s$:

$$M_s(a'(\varepsilon, y_1, \ldots, y_n))[y_i/z_i \,;\, i \in [n]] = YIELD_f(N_s(a(\varepsilon)))$$

(Q_σ) For every $k \geq 0$, $\sigma \in \Sigma_+^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, $Q_\sigma((s_1, \ldots, s_k))$ is true, iff
for every position ν of $w = topsort(N, (\sigma, s_1, \ldots, s_k))$:

$$M_{(\sigma,s_1,\ldots,s_k)}(YIELD_f(rhs_N(w(\nu), \sigma))) = YIELD_f(N_{(\sigma,s_1,\ldots,s_k)}(rhs_N(w(\nu), \sigma)))$$

Again, this part of the proof is not presented here. We only mention the additional statements, which have to be proved in order to realize this part of the proof. To prove Condition 2. of the principle of proof by simultaneous induction, the following statement must be proved by structural induction:

For every $sub \in sub(rhs_N(w(\nu), \sigma))$,

$$M_{(\sigma,s_1,\ldots,s_k)}(YIELD_f(sub)) = YIELD_f(N_{(\sigma,s_1,\ldots,s_k)}(sub)).$$

The case $sub = d(zi)$ where $d \in F_s$ needs the proof of the following additional statement by structural induction:

For every $t \in SF(\emptyset, F_h, \{\varepsilon\}, \Delta)$,

$$\phi^M_{(\sigma,s_1,\ldots,s_k),i,(z_1,\ldots,z_n)}(YIELD_f(t)[z_i/y_i \,;\, i \in [n]]) = YIELD_f(\phi^N_{(\sigma,s_1,\ldots,s_k),i,()}(t)).$$

Now we can prove $M_{root(s)}(\tilde{a}_{in}(\varepsilon)) = YIELD_f(N_{root(s)}(a_{in}(\varepsilon))$:

	$M_{root(s)}(\tilde{a}_{in}(\varepsilon))$	
$=$	$M_{(root,s)}(rhs_M(\tilde{a}_{in}, z, root))$	(Def. of $M_{root(s)}$)
$=$	$M_{(root,s)}(rhs_M(a'_{in}, z, root)[y_i/z_i \,;\, i \in [n]])$	(Construction of M)
$=$	$M_{(root,s)}(rhs_M(a'_{in}, z, root))[y_i/z_i \,;\, i \in [n]]$	(#) (see below)
$=$	$M_{root(s)}(a'_{in}(\varepsilon, y_1, \ldots, y_n))[y_i/z_i \,;\, i \in [n]]$	(Def. of $M_{root(s)}$)
$=$	$YIELD_f(N_{root(s)}(a_{in}(\varepsilon)))$	(Def. of $P(root(s))$)

(#) is true, because $M_{(root,s)}$ treats variables $y_i \in Y_n$ and working symbols $z_i \in Z_n$ in the same way: it leaves them unchanged (cf. Definition 3.25). So the substitution can take place before or after the application of $M_{(root,s)}$.

The second equation of Theorem 4.6 follows immediately from the first one, if the inherited functions are left out in the proof. □

Theorem 4.7 $S_f I_f T = SIT \circ YIELD$ and $S_f T = ST \circ YIELD$

Proof: The equations follow immediately from the two Lemmata 4.4 and 4.6. □

In order to prove the desired characterization result $S_f I_f T = SIT \circ SIT$ we need three more lemmata:

1. $YIELD \subseteq SIT$ (cf. Lemma 4.9)
2. $SIT \subseteq S_f T$ (cf. Lemma 4.10)
3. $SIT \circ ST \subseteq SIT$ (cf. Lemma 4.11)

Then we can prove the characterization result as follows:

Theorem 4.8 $S_f I_f T = SIT \circ SIT$.

Proof:

$$\begin{array}{lcll} S_f I_f T & = & SIT \circ YIELD & \text{(Theorem 4.7)} \\ & \subseteq & SIT \circ SIT & \text{(Lemma 4.9)} \\ & \subseteq & SIT \circ S_f T & \text{(Lemma 4.10)} \\ & \subseteq & SIT \circ ST \circ YIELD & \text{(Theorem 4.7)} \\ & \subseteq & SIT \circ YIELD & \text{(Lemma 4.11).} \end{array}$$

□

The inclusion $YIELD \subseteq SIT$ follows from $YIELD \subseteq L1S\text{–}AG$ (cf. Theorem 1.3 of [Eng81]) where $L1S\text{–}AG$ denotes the class of tree functions computed by left–to–right evaluatable attribute grammars with only one synthesized attribute. We briefly recall the construction in our terminology:

Lemma 4.9 (Theorem 1.3 of [Eng81]) $YIELD \subseteq SIT$.

Proof: Let Σ and Δ be two ranked alphabets and let $f : \Sigma^{(0)} \longrightarrow T\langle \Delta \cup Y_n \rangle$ be a mapping for some $n \geq 0$. Construct the full attributed tree transducer $N = (\mathcal{F}, \Delta, \Sigma, a, root, R)$ as follows:

- $\mathcal{F} = (F, F_s, F_h)$ with $F_s = \{a\}$ and $F_h = \{b_1, \ldots, b_n\}$
- – For every $\sigma \in \Sigma^{(k+1)}$ with $k \geq 0$, R_σ contains the rule

  $$a(z) \to a(z1)$$

  and for every $j \in [k]$ and $m \in [n]$ the rule

  $$b_m(z(j+1)) \to b_m(z);$$

  moreover, if $k \geq n$, then for every $m \in [n]$ the rule

  $$b_m(z1) \to a(z(m+1))$$

is in R_σ, if $k < n$, then for every $m \in [k]$ the rule

$$b_m(z1) \to a(z(m+1))$$

is in R_σ and for every $m \in \{k+1, \ldots, n\}$ the rule

$$b_m(z1) \to b_m(z)$$

is in R_σ.

- For every $\sigma \in \Sigma^{(0)}$, R_σ contains the rule

$$a(z) \to f(\sigma)[y_m/b_m(z); m \in [n]].$$

- R_{root} contains the rule

$$a(z) \to a(z1)$$

and for every $m \in [n]$ the rule

$$b_m(z1) \to y_m$$

Then it is easy to see that $YIELD_f = \tau(N)$. □

The second inclusion $SIT \subseteq S_fT$ has been proved in [Fra82] for attribute grammars rather than full attributed tree transducers (also cf. Section 5.1 of [CF82] and Section 4.2 of [Eng84]). In Lemma 5.5 of [FHVV93], the inclusion has been proved for a slight variation of attributed tree transducers. At the end of this section we will prove the result $S_fIT \subseteq S_fT$ (cf. Theorem 4.17), which is a generalization of $SIT \subseteq S_fT$ that allows also synthesized functions. Hence, we only cite the result here without proving it.

Lemma 4.10 (e.g. Lemma 5.5 of [FHVV93]) $SIT \subseteq S_fT$.

In Theorem 4.3 of [Fül81] it is shown that attributed tree transducers are closed under right composition with top–down tree transducers. This property also holds for full attributed tree transducers, i.e., *si*–tree transducers.

Lemma 4.11 (Theorem 4.3 of [Fül81]) $SIT \circ ST \subseteq SIT$.

Proof: Let N and M be a full attributed tree transducer and a top–down tree transducer, respectively. The idea of the construction of a full attributed tree transducers N' such that $\tau(N) \circ \tau(M) = \tau(N')$ is to take the cartesian product of the functions of N and M, translate the right–hand sides of rules of N by M, and use these new trees as right–hand sides of rules of N'. The formal construction can be taken over exactly as it is presented in [Fül81] for attributed tree transducers. Clearly, since we deal here with a full attributed tree transducer, the right–hand sides of $(b, z1, root)$–rules (for any inherited attribute b) have to be translated by M too. □

This completes the series of lemmata which were needed to prove the characterization $S_fI_fT = SIT \circ SIT$. We derive three corollaries from the characterization result:

1. we describe an upper bound on the height of output trees computed by mat tree transducers,

2. we prove the strictness of the composition hierarchy of mat tree transducers, and

3. we show that mat tree transducers are closed under right composition with top–down tree transducers but not under left composition with top–down tree transducers.

Corollary 4.12 Let M be an mat tree transducer and let $\tau(M)(s) = t$. Then $height(t) \leq 2^{c \cdot size(s)}$ for some constant $c > 0$.

Proof: By Theorem 4.8 there are two full attributed tree transducers N_1 and N_2 such that $\tau(M) = \tau(N_1) \circ \tau(N_2)$. Thus there is a tree t' such that $\tau(N_1)(s) = t'$ and $\tau(N_2)(t') = t$. By Lemma 4.1(3), there are constants d and e such that $height(t') \leq d \cdot size(s)$ and $height(t) \leq e \cdot size(t')$. Thus $height(t) \leq e \cdot size(t') \leq e \cdot 2^{b \cdot height(t')} \leq e \cdot 2^{b \cdot d \cdot size(s)} \leq 2^{c \cdot size(s)}$ for some constant $b, c > 0$. □

In Corollary 4.1 of [Fül81] it has been shown that the composition hierarchy of attributed tree transducers is strict. Because of Lemma 4.1(3), the proof of this property carries over immediately to full attributed tree transducers.

Lemma 4.13 For every $n \geq 1$, $SIT^n \subset SIT^{n+1}$.

It is clear that the strictness of the composition hierarchy of mat tree transducers follows immediately from this lemma and from the characterization result $S_f I_f T = SIT \circ SIT$.

Theorem 4.14 For every $n \geq 1$, $S_f I_f T^n \subset S_f I_f T^{n+1}$.

Figure 11 shows an overview of the composition hierarchies of full attributed tree transducers, macro tree transducers, and mat tree transducers. Note that the strictness of the composition hierarchy of macro tree transducers has been shown in Theorem 4.16 of [EV86].

By using the fact that full attributed tree transducers are closed under right composition with top–down tree transducers, and the characterization result of mat tree transducers, we can prove that mat tree transducers are closed under right composition with top–down tree transducers.

Corollary 4.15 $S_f I_f T \circ ST \subseteq S_f I_f T$.

Proof:

$$\begin{aligned} S_f I_f T \circ ST &= SIT \circ SIT \circ ST && \text{(by Theorem 4.8)} \\ &\subseteq SIT \circ SIT && \text{(by Lemma 4.11)} \\ &= S_f I_f T && \text{(by Theorem 4.8).} \end{aligned}$$

□

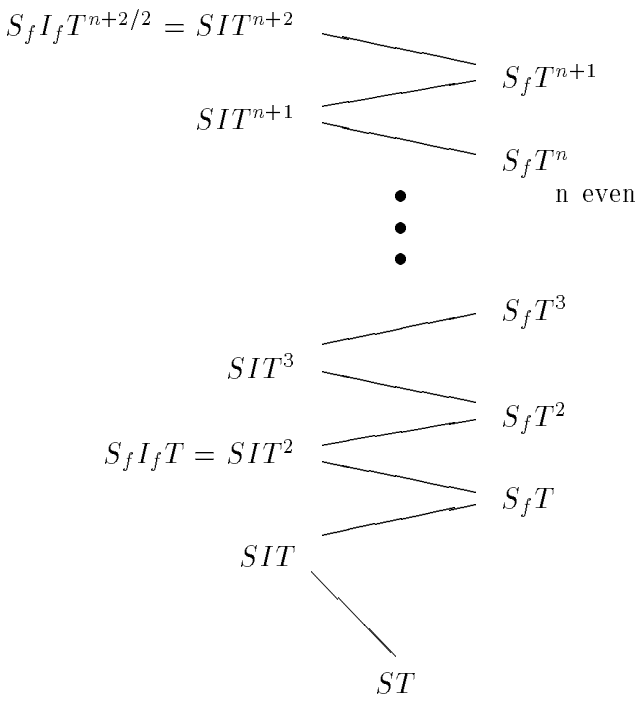

Figure 11: Composition hierarchies.

Note that it would also have been possible, to prove Corollary 4.15 directly in analogy to the proof of $S_f T \circ ST \subseteq S_f T$ in Theorem 4.12 of [EV85]. However, mat tree transducers are not closed under left composition with top–down tree transducers.

Corollary 4.16 $ST \circ S_f I_f T \not\subseteq S_f I_f T$.

Proof: Assume that $ST \circ S_f I_f T \subseteq S_f I_f T$. Then $ST \circ S_f I_f T = S_f I_f T$, because ST contains all identity tree transformations $id_\Sigma : T\langle\Sigma\rangle \longrightarrow T\langle\Sigma\rangle$. Then:

$$\begin{array}{lcll} SIT \circ SIT & = & S_f I_f T & \text{(by Theorem 4.8)} \\ & = & ST \circ S_f I_f T & \text{(by assumption)} \\ & = & ST \circ SIT \circ SIT & \text{(by Theorem 4.8)} \\ & \subseteq & ST \circ S_f T \circ SIT & \text{(by Lemma 4.10)} \\ & \subseteq & S_f T \circ SIT & \text{(by Corollary 4.10 of [EV85])} \\ & = & ST \circ YIELD \circ SIT & \text{(by Theorem 4.7)} \\ & \subseteq & ST \circ SIT \circ SIT & \text{(by Lemma 4.9).} \end{array}$$

In particular, we have $(*)$ $SIT \circ SIT = S_f T \circ SIT$. Since $SIT \subset S_f T$ (by Lemma 4.3), we also obtain $SIT \circ SIT \subset S_f T \circ SIT$ which is a contradiction to $(*)$. Thus, $ST \circ S_f I_f T \not\subseteq S_f I_f T$. □

4.3 Elimination of inherited attributes

We want to conclude this section with the result that the addition of inherited attributes does not increase the computational power of macro tree transducers.

The following theorem $S_fIT \subseteq S_fT$ is a generalization of the result $SIT \subseteq S_fT$, which has been proved in [Fra82] (also cf. [Eng84] and [FHVV93]).

Theorem 4.17 $S_fIT \subseteq S_fT$.

Proof: Let $M = (\mathcal{F}, \Delta, \Sigma, a_{in}, root, R)$ be an s_fi–tree transducer with initial chain rules (see Definition 3.30) with $\mathcal{F} = (F, F_s, F_h)$, $F_h = \{b_1, \ldots, b_q\}$ and $rank_F(b_i) = 1$ for every $b_i \in F_h$. We can assume, that an element $\alpha \in \Delta^{(0)}$ exists, which we will need in the construction to stop the process of building up right–hand sides. We construct an s_f–tree transducer M' which is equivalent to M.

Let $M' = (\mathcal{F}', \Delta, \Sigma, \hat{a}_{in}, root, R')$ be defined as follows:

$\mathcal{F}' = (F', F'_s, F'_h)$ with $F'_s = \{a'^{(p+q+1)} \mid a \in F_s^{(p+1)}\} \cup \{\hat{a}_{in}^{(1)}\}$ and $F'_h = \emptyset$.

The new set $R' = \bigcup_{\sigma \in \Sigma_+} R'_\sigma$ of rules is built up from the set $R = \bigcup_{\sigma \in \Sigma_+} R_\sigma$:

For every rule $a(z, y_1, \ldots, y_p) \to \zeta$ in R_σ with $a \in F_s^{(p+1)}$, $\sigma \in \Sigma_+^{(k)}$ and $\zeta \in RHS(F_s, F_h, \Delta, Y_p, \sigma)$ there is a rule

$$a'(z, y_1, \ldots, y_p, y_{p+1}, \ldots, y_{p+q}) \to \psi_{\sigma,p,\emptyset}(\zeta)$$

in R'_σ.
The functions $\psi_{\sigma,p,Z} : RHS(F_s, F_h, \Delta, Y_p, \sigma) \longrightarrow RHS(F'_s, \emptyset, \Delta, Y_{p+q}, \sigma)$ with $\sigma \in \Sigma_+^{(k)}$, $p \in I\!N$ and $Z \subseteq F_s \times [k]$ are defined as follows:

$$
\begin{array}{lcll}
\psi_{\sigma,p,Z}(y_j) & := & y_j & \text{for every } j \in [p], \\
\psi_{\sigma,p,Z}(\delta(\zeta_1, \ldots, \zeta_n)) & := & \delta(\psi_{\sigma,p,Z}(\zeta_1), \ldots, \psi_{\sigma,p,Z}(\zeta_n)) & \\
 & & & \text{for every } \delta \in \Delta^{(n)} \\
 & & & \text{and } \zeta_1, \ldots, \zeta_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma), \\
\psi_{\sigma,p,Z}(b_i(z)) & := & y_{p+i} & \text{for every } b_i \in F_h^{(1)}, \\
\psi_{\sigma,p,Z}(d(zj, \zeta_1, \ldots, \zeta_n)) & := & d'(zj, \psi_{\sigma,p,Z}(\zeta_1), \ldots, \psi_{\sigma,p,Z}(\zeta_n), & \\
 & & \quad \psi_{\sigma,p,Z\cup\{(d,j)\}}(rhs(b_1, zj, \sigma)), \ldots, \psi_{\sigma,p,Z\cup\{(d,j)\}}(rhs(b_q, zj, \sigma))) & \\
 & & & \text{for every } d \in F_s^{(n+1)} \\
 & & & \text{and } \zeta_1, \ldots, \zeta_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma), \\
 & & & \text{if } (d, j) \notin Z, \\
 & := & \alpha & \text{for every } d \in F_s^{(n+1)}, \\
 & & & \zeta_1, \ldots, \zeta_n \in RHS(F_s, F_h, \Delta, Y_p, \sigma), \\
 & & & \text{and for any } \alpha \in \Delta^{(0)}, \\
 & & & \text{if } (d, j) \in Z
\end{array}
$$

Additionally, we have a rule

$\hat{a}_{in}(z) \to \psi_{root,0,\emptyset}(rhs(a_{in}, z, root))$ $\quad (= \hat{a}_{in}(z) \to \psi_{root,0,\emptyset}(a_{in}(z1)) \quad)$

in R_{root} and for every $\sigma \in \Sigma$ we have a dummy rule

$\hat{a}_{in}(z) \to \alpha$

in R_σ.

The $(\hat{a}_{in}, z, root)$–rule switches to the other functions of M', which have q more arguments for the inherited functions of M. It has the same right–hand side as the $(a'_{in}, z, root)$–rule that is never used in a derivation like all the other $(a', z, root)$–rules for $a \in F_s$. Note, that now it becomes unimportant that the new parameters of the functions a' are undefined at the root.

Now we prove, that the construction of an s_f–tree transducer M' for a given $s_f i$–tree transducer M is correct. In particular we show for every control tree $root(s)$ with $s \in T\langle\Sigma\rangle$, that

$(\#)$ $\quad M_{root(s)}(a_{in}(\varepsilon)) = M'_{root(s)}(\hat{a}_{in}(\varepsilon))$

Therefore we prove the following predicate P and, for every $k \geq 0$ and $\sigma \in \Sigma_+^{(k)}$, the predicate Q_σ by the principle of proof by simultaneous induction:

(P) For every $s \in T\langle\Sigma\rangle \cup \{root(s') \mid s' \in T\langle\Sigma\rangle\}$, $P(s)$ is true, iff
for every $a \in F_s^{(p+1)}$ with $p \geq 0$:

$$M_s(a(\varepsilon, y_1, \ldots, y_p)) = M'_s(a'(\varepsilon, y_1, \ldots, y_p, y_{p+1}, \ldots, y_{p+q}))[y_{p+i}/b_i(\varepsilon) \ ; \ i \in [q]]$$

(Q_σ) For every $k \geq 0$, $\sigma \in \Sigma_+^{(k)}$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, $Q_\sigma((s_1, \ldots, s_k))$ is true, iff
for every position ν of $w = topsort(M, (\sigma, s_1, \ldots, s_k))$ and

- if $w(\nu) = (a, z)$ with $a \in F_s$, then for $p + 1 = rank(a)$,
- if $w(\nu) = (b_i, zj)$ with $b_i \in F_h$ and $j \in [k]$, then for every $p \in I\!N$, if there exists a synthesized function $a \in F_s^{(p+1)}$:

$$M_{(\sigma,s_1,\ldots,s_k)}(rhs(w(\nu), \sigma)) = M'_{(\sigma,s_1,\ldots,s_k)}(\psi_{\sigma,p,\emptyset}(rhs(w(\nu), \sigma)))[y_{p+i}/b_i(\varepsilon) \ ; \ i \in [q]]$$

Then we can prove $(\#)$ as follows:

$$\begin{array}{lll}
& M_{root(s)}(a_{in}(\varepsilon)) & \\
= & M'_{root(s)}(a'_{in}(\varepsilon, y_1, \ldots, y_q))[y_i/b_i(\varepsilon) \ ; \ i \in [q]] & \text{(Def. of } P(root(s))) \\
= & M'_{(root,s)}(rhs(a'_{in}, z, root))[y_i/b_i(\varepsilon) \ ; \ i \in [q]] & \text{(Def. of } M'_{root(s)}) \\
= & M'_{(root,s)}(rhs(\hat{a}_{in}, z, root))[y_i/b_i(\varepsilon) \ ; \ i \in [q]] & \text{(Construction of } M') \\
= & M'_{(root,s)}(rhs(\hat{a}_{in}, z, root)) & (rhs(\ldots) \text{ without any } y_i) \\
= & M'_{root(s)}(\hat{a}_{in}(\varepsilon)) & \text{(Def. of } M'_{root(s)})
\end{array}$$

The proof of the predicate P and, for every $k \geq 0$ and $\sigma \in \Sigma_+^{(k)}$, of the predicate Q_σ by the principle of proof by simultaneous induction is not presented here. In order to show Condition 2. (cf. Section 2) of this principle, the following statement has to be proved by structural induction:

For every $t \in sub(rhs(w(\nu), \sigma))$,

$$M_{(\sigma,s_1,\ldots,s_k)}(t) = M'_{(\sigma,s_1,\ldots,s_k)}(\psi_{\sigma,p,\emptyset}(t))[y_{p+i}/b_i(\varepsilon)\ ;\ i \in [q]]$$

In the case $t = d(zj, t_1, \ldots, t_r)$ for some $d \in F_s^{(r+1)}$ we have to show one further statement by structural induction:

For every $t \in sub(M'_{s_j}(d(\varepsilon, y_1, \ldots, y_{r+q}))) \subseteq T\langle \Delta \cup Y_{r+q} \rangle$,

$$\begin{array}{ll} & \phi^M_{(\sigma,s_1,\ldots,s_k),j,(t_1,\ldots,t_r)}(t[y_{r+i}/b_i(\varepsilon)\ ;\ i \in [q]]) \\ = & (\phi^{M'}_{(\sigma,s_1,\ldots,s_k),j,(\psi_{\sigma,p,\emptyset}(t_1),\ldots,\psi_{\sigma,p,\emptyset}(t_r),\psi_{\sigma,p,\{(d,j)\}}(rhs(b_1,zj,\sigma)),\ldots,\psi_{\sigma,p,\{(d,j)\}}(rhs(b_q,zj,\sigma)))}(t)) \\ & \quad [y_{p+i}/b_i(\varepsilon)\ ;\ i \in [q]] \end{array}$$

This completes the proof of our theorem. □

Corollary 4.18 $S_f IT = S_f T$

Example 4.19 Let $M = (\mathcal{F}, \Delta, \Sigma, s, root, R)$ be an $s_f i$–tree transducer with initial chain rules with $\mathcal{F} = (F, F_s, F_h)$, $F_s = \{s^{(1)}, e^{(2)}\}$, $F_h = \{i^{(1)}\}$, $\Delta = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$, $\Sigma = \{\gamma, \alpha\}$ and $R = R_{root} \cup R_\gamma \cup R_\alpha$ with:

$$\begin{array}{lll} R_{root}: & s(z) & \to\ s(z1) \\ & e(z, y_1) & \to\ e(z1, y_1) \\ & i(z1) & \to\ e(z1, \alpha) \\ R_\gamma: & s(z) & \to\ \sigma(i(z), s(z1)) \\ & e(z, y_1) & \to\ e(z1, e(z1, y_1)) \\ & i(z1) & \to\ e(z1, i(z)) \\ R_\alpha: & s(z) & \to\ i(z) \\ & e(z, y_1) & \to\ \gamma(y_1) \end{array}$$

The construction of Theorem 4.17 delivers the following s_f–tree transducer $M' = (\mathcal{F}', \Delta, \Sigma, \hat{s}, root, R')$ with $\mathcal{F}' = (F', F'_s, \emptyset)$, $F_s = \{s'^{(2)}, e'^{(3)}, \hat{s}^{(1)}\}$ and $R' = R'_{root} \cup R'_\gamma \cup R'_\alpha$ with:

$$
\begin{array}{lll}
R'_{root}: \quad \hat{s}(z) & \rightarrow & \psi_{root,0,\emptyset}(s(z1)) \quad \text{where} \\
\quad \psi_{root,0,\emptyset}(s(z1)) & = & s'(z1, \psi_{root,0,\{(s,1)\}}(e(z1,\alpha))) \\
 & = & s'(z1, e'(z1, \psi_{root,0,\{(s,1)\}}(\alpha), \\
 & & \qquad \psi_{root,0,\{(s,1),(e,1)\}}(e(z1,\alpha)))) \\
 & = & s'(z1, e'(z1,\alpha,\alpha)) \\
s'(z,y_1) & \rightarrow & \psi_{root,0,\emptyset}(s(z1)) \quad \text{where} \\
\quad \psi_{root,0,\emptyset}(s(z1)) & = & s'(z1, e'(z1,\alpha,\alpha)) \\
e'(z,y_1,y_2) & \rightarrow & \psi_{root,1,\emptyset}(e(z1,y_1)) \quad \text{where} \\
\quad \psi_{root,1,\emptyset}(e(z1,y_1)) & = & e'(z1, \psi_{root,1,\emptyset}(y_1), \psi_{root,1,\{(e,1)\}}(e(z1,\alpha))) \\
 & = & e'(z1, y_1, \alpha) \\
R'_{\gamma}: \quad \hat{s}(z) & \rightarrow & \alpha \\
s'(z,y_1) & \rightarrow & \psi_{\gamma,0,\emptyset}(\sigma(i(z), s(z1))) \quad \text{where} \\
\quad \psi_{\gamma,0,\emptyset}(\sigma(i(z), s(z1))) & = & \sigma(\psi_{\gamma,0,\emptyset}(i(z)), \psi_{\gamma,0,\emptyset}(s(z1))) \\
 & = & \sigma(y_1, s'(z1, \psi_{\gamma,0,\{(s,1)\}}(e(z1,i(z))))) \\
 & = & \sigma(y_1, s'(z1, e'(z1, \psi_{\gamma,0,\{(s,1)\}}(i(z)), \\
 & & \qquad \psi_{\gamma,0,\{(s,1),(e,1)\}}(e(z1,i(z)))))) \\
 & = & \sigma(y_1, s'(z1, e'(z1, y_1, \alpha))) \\
e'(z,y_1,y_2) & \rightarrow & \psi_{\gamma,1,\emptyset}(e(z1,e(z1,y_1))) \quad \text{where} \\
\quad \psi_{\gamma,1,\emptyset}(e(z1,e(z1,y_1))) & = & e'(z1, \psi_{\gamma,1,\emptyset}(e(z1,y_1)), \psi_{\gamma,1,\{(e,1)\}}(e(z1,i(z)))) \\
 & = & e'(z1, e'(z1, \psi_{\gamma,1,\emptyset}(y_1), \psi_{\gamma,1,\{(e,1)\}}(e(z1,i(z)))), \alpha) \\
 & = & e'(z1, e'(z1, y_1, \alpha), \alpha) \\
R'_{\alpha}: \quad \hat{s}(z) & \rightarrow & \alpha \\
s'(z,y_1) & \rightarrow & \psi_{\alpha,0,\emptyset}(i(z)) \quad \text{where} \\
\quad \psi_{\alpha,0,\emptyset}(i(z)) & = & y_1 \\
e'(z,y_1,y_2) & \rightarrow & \psi_{\alpha,1,\emptyset}(\gamma(y_1)) \quad \text{where} \\
\quad \psi_{\alpha,1,\emptyset}(\gamma(y_1)) & = & \gamma(\psi_{\alpha,1,\emptyset}(y_1)) \\
 & = & \gamma(y_1)
\end{array}
$$

□

5 Summary and further research topics

In this paper we have introduced a new formal model for the concept of syntax–directed translation with context handling: the macro attributed tree transducer. This model integrates the features of attributed tree transducers and of macro tree transducers in the sense that synthesized and inherited attributes are allowed to have additional parameters to store context information. We have started to investigate theoretical questions about mat tree transducers. In particular,

- we have developed an inductive characterization of the tree transformation which is computed by an mat tree transducer,
- we have proved that mat tree transducers are equivalent to the twofold composition of full attributed tree transducers, and
- we have shown that the addition of inherited attributes does not increase the power of macro tree transducers.

Clearly, there are some questions left open which may be the topic of further research.

1. We conjecture that the difference classes $S_fIT - SI_fT$ and $SI_fT - S_fIT$ are not empty. Intuitively, this would mean that it makes a difference whether parameters (and hence, storing of context information) are allowed only during information transfer from the leaves to the root (i.e., synthesized functions and inherited attributes) or whether they are allowed only during information transfer from the root to the leaves (i.e., synthesized attributes and inherited functions).
2. In [EH92] it has been proved that attribute grammars are equivalent to a syntax–directed translation device which is obtained by starting from a context–free grammar G and attaching with every production of G a production of some term–generating context–free hypergraph grammar in a restricted form (for context–free hypergraph grammars cf., e.g., [BC87, Cou87, HK87, EH91]). In [EV92] it is shown that macro tree transducers are equivalent to top–down tree–to–graph transducers which are term–generating context–free hypergraph grammars of which the derivations are controlled by trees. How does the extension of context–free hypergraph grammars look like such that it is equivalent to mat tree transducers?
3. For a small imperative programming language (similar to PL/0), macro tree transducer like formalisms have been used in [Vog91] and in [Ind79] to specify in a succinct form parts of the static semantics and the generation of code, respectively. On the other hand, the concept of attribute grammars has proved its appropriateness in the area of compiler theory. Does the concept of mat tree transducer contribute to the elegancy and succinctness of specifying compilers?
4. In [Gie88] the *single syntactic used restriction* (for short: ssur) has been introduced for attribute coupled grammars. There it was argued that a great deal of isolated phases of a compiler can be specified by ssur attribute coupled grammars. It was

proved in [Gie88] that ssur attribute coupled grammars are closed under composition. This leads to a mathematical foundation for the modular specification of the compiler. How does the ssur restriction look like for mat tree transducers? Are ssur mat tree transducers closed under composition?

5. For the development of programs it is sometimes desirable to have a syntax–directed editor available. Roughly speaking, such an editor can be viewed as an attribute grammar in which all the features of the programming language which the designer of the programming language wants to be checked, are specified by semantic rules. The execution of a syntax–directed editor by the user of the system is based on the theory of incremental evaluation of attribute grammars [DRT81, TR81, RT89]. Is it fruitful to extend the specification language of such features from attribute grammars to mat tree transducers? How is the incremental evaluation of mat tree transducers formalized?

References

[ASU86] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers – Principles, Techniques, and Tools*. Addison Wesley, 1986.

[AU71] A.V. Aho and J.D. Ullman. Translations on a context free grammar. *Inform. and Control*, 19:439–475, 1971.

[AU73] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I and Vol. II*. Prentice-Hall, 1973.

[BC87] M. Bauderon and B. Courcelle. Graph expressions and graph rewritings. *Math. Syst. Theory*, 20:83–127, 1987.

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

[Cou84] B. Courcelle. Attribute grammars: definitions, analysis of dependencies, proof methods. In B. Lorho, editor, *Methods and tools for compiler construction*, pages 81–102. Cambridge University Press, 1984.

[Cou87] B. Courcelle. An axiomatic definition of context-free rewriting and its application to nlc graph grammars. *Theoret. Comput. Sci.*, 55:141–181, 1987.

[DRT81] A. Demers, T. Reps, and T. Teitelbaum. Incremental evaluation for attribute grammars with application to syntax-directed editors. *8th Assoc. Comput. Sci Princ. of Progr. Lang. 1981, 105-116*, 1981.

[EH91] J. Engelfriet and L. Heyker. The string generation power of of context-free hypergraph grammars. *J. Comput. Syst. Sci.*, 43:328–360, 1991.

[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.

[Ems91] K. Emser-Loock. Integration von attributierten Grammatiken und primitivrekursiven Programmschemata. Master Thesis, RWTH Aachen, 1991.

[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.

[Eng81] J. Engelfriet. Tree transducers and syntax directed semantics. Technical Report Memorandum 363, Technische Hogeschool Twente, 1981.

[Eng84] J. Engelfriet. Attribute grammars: attribute evaluation methods. In B. Lorho, editor, *Methods and tools for compiler construction*, pages 103–138. Cambridge University Press, 1984.

[Eng86] J. Engelfriet. Context-free grammars with storage. Technical Report 86-11, University of Leiden, 1986.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31:71–145, 1985.

[EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoret. Comput. Sci.*, 42:251–368, 1986.

[EV92] J. Engelfriet and H. Vogler. The translation power of top-down tree-to-graph transducers. Technical Report 92–14, University of Ulm, 1992. to appear in J. Comput. Syst. Sci.

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

[Fra82] P. Franchi-Zannettacci. *Attributs semantiques et schemas de programmes.* PhD thesis, Universite de Bordeaux I, 1982.

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

[FV91] Z. Fülöp and S. Vagvölgyi. Attributed tree transducers cannot induce all deterministic bottom-up tree transformations. manuscript, submitted for publication, 1991.

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

[Gor79] M.J.C. Gordon. *The denotational description of programming languages; an introduction.* Springer-Verlag, 1979.

[GTWW77] J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. Initial algebra semantics and continuous algebras. *J. Assoc. Comput. Mach.*, 24:68–95, 1977.

[HK87] A. Habel and H.-J. Kreowski. May we introduce to you: hyperedge replacement. In *Graph grammars and their application to computer science, LNCS 291*, pages 15–26. Springer-Verlag, 1987.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *J. Assoc. Comput. Mach.*, 27:797–821, 1980.

[Ind79] K. Indermark. Functional compiler description. *Banach Center Publications*, 21:223–232, 1979.

[Iro61] E.T. Irons. A syntax directed compiler for ALGOL 60. *Comm. Assoc. Comput. Mach.*, 4:51–55, 1961.

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

[Kos71] C.H.A. Koster. Affix grammars. In *Proc. of the IFIP working conf. on ALGOL68 implementation.* North-Holland, Amsterdam, 1971.

[KSV89] M.F. Kuiper, S.D. Swierstra, and H.H. Vogt. Higher order attribute grammars. In *Proc. of SIGPLAN '89, Symp. Progr. Lang. Design and Impl.*, pages 131–145. ACM Press, 1989.

[KV92] A. Kühnemann and H. Vogler. Synthesized and inherited functions – a new computational model for syntax–directed semantics. Technical Report 92–06, University of Ulm, 1992.

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

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

[RT89] Th.W. Reps and T. Teitelbaum. *The synthesizer generator - a system for constructing language-based editors*. Springer-Verlag, 1989.

[SS71] D. Scott and C. Strachey. Toward a mathematical semantics for computer languages. In J. Fox, editor, *Computers and automata*, pages 19–46. Wiley, New York, 1971.

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

[TR81] T. Teitelbaum and T. Reps. Cornell program synthesizer: syntax-directed programming env. *CACM 24 (1981), 563-573*, 1981.

[Vog91] H. Vogler. Functional description of the contextual analysis in block-structured programming languages: a case study of tree transducers. *Science of Computer Programming*, 16:251–275, 1991.