A Pumping Lemma for Output Languages of Macro Tree Transducers

Armin Kühnemann

Grundlagen der Programmierung

Institut für Softwaretechnik I, Fakultät Informatik

Technische Universität Dresden

D–01062 Dresden, Germany

e–mail: kuehne@orchid.inf.tu–dresden.de

Abstract

The concept of macro tree transducer is a formal model for studying properties of syntax–directed translations and of functional languages which are based on primitive recursion. In this paper, for output languages of producing and nondeleting macro tree transducers, we introduce and prove a pumping lemma. We apply this pumping lemma to gain two results: (1) there is no producing and nondeleting macro tree transducer which computes the set of all monadic trees with double exponential height as output and (2) there are hierarchies of producing and nondeleting macro tree transducers with respect to their number of functions.

1 Introduction

Pumping lemmata are useful tools to prove that a given language L is not an element of a class $\mathcal{L}$ of formal languages, where $\mathcal{L}$ usually is defined by a class of grammars or translation schemes. Pumping lemmata have been investigated for different kinds of languages, for example string languages, tree languages, and graph languages.

In this introduction we do not give a survey over existing pumping lemmata for several kinds of language classes, as it was done for example in [KV94a]. Rather we want to restrict ourselves to a list of pumping lemmata for tree languages which are generated as output of tree transducers; let us call them tree transducer languages.

In advance, we recall additionally the well known pumping lemma for context–free languages of Bar–Hillel, Perles, and Shamir [BPS61]. The structure of this pumping lemma has served as pattern for most of the existing pumping lemmata in the literature and therefore it seems to be the root of the research about pumping lemmata. Since it also has influenced our pumping lemma, we present here a short version of the lemma's central statement and we recall its proof idea:

For every context–free grammar G there is a natural number pi_G, called the **p**umping **i**ndex of G, such that for every string z which is an element of the language $L(G)$ generated by G and which has at least the length pi_G, the following holds. There is a decomposition $z = uvwxy$, such that v or x is not the empty string and such that for every natural number j, the pumped string uv^jwx^jy is an element of $L(G)$.

The proof can be sketched as follows: We choose a sufficiently long string z of $L(G)$, such that its derivation tree e has the following property: e is high enough, such that it has a path p, on which two different nodes x_1 and x_2 are labeled by the same nonterminal symbol. Assuming that x_1 is closer to the root of e than x_2, we can define the following tree $\bar{e}$: Roughly speaking, the tree $\bar{e}$ is that part of e which has x_1 as root and from which the subtree rooting at x_2 is pruned. Since x_1 and x_2 have the same label, we can construct for every natural number j a new derivation tree, by repeating j times the tree $\bar{e}$. Taking the yield of these derivation trees, we obtain new elements of $L(G)$.

Now we briefly survey the pumping lemmata for tree transducer languages. First Aho and Ullman have inspected pumping lemmata for output languages of translation schemes in [AU71], namely for generalized syntax directed translations. Perrault and Ésik have introduced in [Per76] and [Ési80], respectively, pumping lemmata for (nondeterministic) top–down tree transducers (cf. [Rou70, Tha70, Eng75]). Engelfriet, Rozenberg, and Slutzki have presented in [ERS80] a pumping lemma for output languages of deterministic top–down tree–to–string transducers which has a structure that is closely related to the pumping lemma for context–free string languages. Kühnemann and Vogler have investigated in [KV94a] a pumping lemma for output languages of restricted attributed tree transducers (cf. [Fül81]). The proofs of the latter two lemmata had a big influence on the development of the pumping lemma for output languages of restricted macro tree transducers which we present in this paper.

The concept of macro tree transducer has been introduced by Engelfriet in [Eng80] (cf. also [CF82, EV85]). It is a formal model for studying properties of syntax–directed translations and of functional languages which are based on primitive recursion. A macro tree transducer can be understood as extended scheme of primitive recursion over trees, where additionally simultaneous recursion and nested function calls in parameter positions are allowed. In [EV91] the relation between tree transducers and primitive recursive tree functions is examined in detail (cf. also [FHVV93]).

A macro tree transducer M takes input trees over a ranked alphabet Σ of input symbols and delivers output trees over a ranked alphabet Δ of output symbols. For this translation process, M uses a ranked alphabet F of functions which have at least rank 1, and a set R of rules which describe the possible derivations. For every function f with rank $n+1$ and for every input symbol σ with rank k there is exactly one rule of the form $f(\sigma(x_1,\ldots,x_k),y_1,\ldots,y_n) \to \rho$, where $x_1,\ldots,x_k$ (and $y_1,\ldots,y_n$) are recursion variables (and context variables, respectively) and ρ is a tree over F, Δ, and $x_1,\ldots,x_k,y_1,\ldots,y_n$, such that for every subtree of ρ of the form $f'(\rho_0,\rho_1,\ldots,\rho_m)$ with $f' \in F$, the tree ρ_0 has to be a recursion variable x_i.

Roughly speaking, the above rule can be applied to a subtree $f(\sigma(e_1,\ldots,e_k),t_1,\ldots,t_n)$ of a tree t, replacing this subtree by ρ', where ρ' results from ρ by substitution of every x_i by e_i and of every y_i by t_i. This rule applications can be repeated to transform an input tree into an output tree.

Every macro tree transducer M computes a total function from input trees to output trees. This function is called the tree transformation of M. The output language of a macro tree transducer M is defined as the range of the tree transformation of M.

The pumping lemma which is presented in this paper, is restricted to producing and nondeleting macro tree transducers: A macro tree transducer is producing, if the right–hand side of every rule contains at least one output symbol. A macro tree transducer is nondeleting, if every variable of the left–hand side of a rule also occurs on the corresponding right–hand side.

The central idea of our pumping lemma for output languages of producing and nondeleting macro tree transducers is adopted from the proof of the pumping lemma for context–free string languages that was outlined at the beginning of this introduction. In the case of context–free string languages we have to inspect a derivation tree of a sufficiently long string to deduce new pumped strings. Here we have to consider an input tree corresponding to a sufficiently large output tree to obtain new pumped output trees. Let us state this more precisely in analogy to the pumping lemma for context–free languages:

For every producing and nondeleting macro tree transducer M, a natural number pi_M, called the **p**umping **i**ndex of M, can be constructed. If we choose an output tree t from the output language of M which has at least pi_M nodes, then every input tree e which can be transformed into t has the following property: e is high enough, such that it has a path p, on which two different nodes x_1 and x_2 can be found, which have the same set of function occurrences that are needed to calculate subtrees of t. Assuming that x_1 is closer to the root of e than x_2, we can define the following tree $\bar{e}$: Roughly speaking, the tree $\bar{e}$ is that part of e which has x_1 as root and from which the subtree rooting at x_2 is pruned. Since the two nodes are compatible with respect to the needed function occurrences, we can construct new input trees by repeating $\bar{e}$ arbitrarily many times. Translating these input trees by M, we obtain new elements of the output language of M.

The proof is based on the observation that the decomposition of the input tree e induces a decomposition of the output tree t into output patterns and that these patterns are used to construct the new output trees. Thus the pumping process itself can be described by using only the output patterns. Therefore the applications of the pumping lemma are completely independend of the underlying input trees.

The intended applications of the pumping lemma induce the necessity to deal with an additional problem in this paper. In general the following statement is not true: If M is a producing and nondeleting macro tree transducer which translates an input tree e_1 (and e_2) into an output tree t_1 (and t_2, respectively), where the size of e_2 is greater than the size of e_1, then the size of t_2 is greater than the size of t_1. Thus, roughly speaking, also the pumping process does not yield output trees which in every case form a strict hierarchy with respect to their sizes; let us call it growth–hierarchy. We shall solve this problem by adjusting the pumping index, such that in an extended pumping lemma additionally a growth–hierarchy of the pumped output trees can be proved. This proof needs the further restriction for macro tree transducers to be noncopying, i.e. every context variable of the left–hand side of a rule occurs at most once on the corresponding right–hand side. We shall see that this is no further restriction in the context of our applications.

In this paper we apply our pumping lemma to prove the following results:

- There is no producing and nondeleting macro tree transducer which computes the set of all monadic trees with double exponential height as output.
- There are several hierarchies of producing and nondeleting macro tree transducers with respect to
  - their number of functions and
  - their number of functions with rank 1 and 2.

This paper is divided into seven sections, from which this one is the first. In Section 2 we fix elementary notions and notations which are necessary for the remaining sections. Section 3 introduces macro tree transducers. Additionally, we prove and recall some fundamental results for macro tree transducers. Section 4 contains the first pumping lemma together with its proof. Section 5 discusses the desired growth–hierarchy which is necessary for our two applications, and presents the extended pumping lemma. The applications themselves are shown in Section 6. Finally, in Section 7 the reader can find a short summary and a presentation of further research topics.

2 Preliminaries

Here we collect the basic notations, notions, and definitions which are used in this paper. After introducing the general notations we deal with ranked alphabets and trees.

2.1 General Notations

We denote the set of natural numbers including 0 by $I\!N$ and the set of natural numbers without 0 by $I\!N_+$. For every $m \in I\!N$, the set $\{1, \ldots, m\}$ is denoted by $[m]$, thus $[0]$ denotes the empty set $\emptyset$. We shall use the sets $X = \{x_1, x_2, x_3, \ldots\}$ and $Y = \{y_1, y_2, y_3, \ldots\}$ of *variables*. For every $k \in I\!N$, we use the abbreviations $X_k = \{x_1, \ldots, x_k\}$ and $Y_k = \{y_1, \ldots, y_k\}$, thus X_0 and Y_0 are empty sets.

For an arbitrary set S, the cardinality of S is denoted by $card(S)$ and the set of all subsets of S is denoted by $\mathcal{P}(S)$. If S is a nonempty subset of $I\!N$, then $max(S)$ denotes the maximum of S. Additionally, we define $max(\emptyset) = 0$. Let S and T be two sets. A relation $f \subseteq S \times T$ is a *partial function*, if for every $(s, t_1) \in f$ and $(s, t_2) \in f$, the elements t_1 and t_2 are equal. Such a partial function is denoted by $f : S - \rightarrow T$. If additionally for every $s \in S$ there is a $t \in T$ such that $(s, t) \in f$, then f is a *total function*; this will be denoted by $f : S \longrightarrow T$. If $f : S \longrightarrow S$ is a total function, then, for every $i \in I\!N$, the function $f^i : S \longrightarrow S$ is defined as follows: for every $s \in S$, $f^0(s) = s$ and if $i > 0$, then $f^i(s) = f(f^{i-1}(s))$.

If A is an alphabet, then A^* denotes the set of *strings* over A. The empty string is denoted by ε. For a string v and two lists $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ of strings such that $u_i \neq u_j$ for every i, j with $i \neq j$ and such that no pair u_i and u_j overlaps in v, we abbreviate by $v[u_1/v_1, \ldots, u_n/v_n]$ the string which is obtained from v by replacing for every $i \in [n]$ every occurrence of u_i in v by v_i. The resulting string is also denoted by $v[u_i/v_i\,;\, i \in [n]]$ or by $v[u_i/v_i\,;\, u_i \in U]$, where $U = \{u_1, \ldots, u_n\}$. $|v|$ denotes the *length* of a string v over an alphabet which should be known from the context. If v_1 and v_2 are two strings, then we denote the concatenation of v_1 and v_2 by $v_1 \circ v_2$ or simply by $v_1 v_2$. If v is a string, then, for every $i \in I\!N$, the string v^i is defined by $v^0 = \varepsilon$ and $v^i = vv^{i-1}$ for every $i > 0$. If V_1 and V_2 are two sets of strings, then $V_1 \cdot V_2 = \{v_1 v_2 \mid v_1 \in V_1,\ v_2 \in V_2\}$.

Let $\Rightarrow$ be a binary relation on some set T. Then, $\Rightarrow^*$ and $\Rightarrow^+$ denote the transitive, reflexive closure of $\Rightarrow$ and the transitive closure of $\Rightarrow$, respectively. Let $n \in I\!N_+$. If $t_j \in T$ for every $j \in [n+1]$ and if $t_j \Rightarrow t_{j+1}$ for every $j \in [n]$, then the sequence $t_1 \Rightarrow t_2 \Rightarrow \ldots \Rightarrow t_{n+1}$ is called a *derivation*. If only the first element t_1 and the last element t_{n+1} of a derivation are important, we also use the notation $t_1 \Rightarrow^+ t_{n+1}$. Note that in general, this notation denotes more than one derivation. If $t \Rightarrow^* t'$ for $t, t' \in T$ and if there is no $t'' \in T$ such that $t' \Rightarrow t''$, then t' is called a *normal form of t with respect to* $\Rightarrow$. In general t can have either no or one or more than one normal form. If the normal form of t with respect to $\Rightarrow$ exists and if it is unique, then it is denoted by $nf(\Rightarrow, t)$. The relation $\Rightarrow$ is *confluent*, if for every $t, t_1, t_2 \in T$ with $t \Rightarrow^* t_1$ and $t \Rightarrow^* t_2$, there is a $t' \in T$ such that $t_1 \Rightarrow^* t'$ and $t_2 \Rightarrow^* t'$. It is *noetherian* or *terminating*, if there is no infinite derivation of $\Rightarrow$. If $\Rightarrow$ is noetherian and confluent, then for every $t \in T$, the normal form of t exists and it is unique.

2.2 Ranked Alphabets, Trees, and Tree Transformations

A *ranked alphabet* is a pair $(\Sigma, rank_\Sigma)$ where Σ is a finite set and $rank_\Sigma : \Sigma \longrightarrow I\!N$ is a mapping which associates with every symbol $\sigma \in \Sigma$ a natural number $rank_\Sigma(\sigma)$ called the rank of the symbol. If $\sigma \in \Sigma$ with $rank_\Sigma(\sigma) = n$, and Σ is clear from the context, then we also write $\sigma^{(n)}$ or $rank(\sigma) = n$. If the rank function is clear from the context, then it is dropped from the notation. The set of elements with rank n is denoted by $\Sigma^{(n)}$. A ranked alphabet Σ is called *monadic*, if $\Sigma = \Sigma^{(0)} \cup \Sigma^{(1)}$.

For a ranked alphabet Σ and a set S, the set of *trees over* Σ *indexed by* S, denoted by $T\langle\Sigma\rangle(S)$, is the smallest subset $T \subseteq (\Sigma \cup S \cup \{(,),,\})^*$ such that

1. $S \subseteq T$ and
2. for every $\sigma \in \Sigma^{(n)}$ with $n \geq 0$ and $t_1, \ldots, t_n \in T$, the string $\sigma(t_1, \ldots, t_n) \in T$.

$T\langle\Sigma\rangle(\emptyset)$ is abbreviated by $T\langle\Sigma\rangle$. A subset $L \subseteq T\langle\Sigma\rangle$ is called a *tree language*. If Σ is monadic, then we call L a *monadic (tree) language*. For a symbol $\sigma \in \Sigma^{(0)}$ we simply write σ instead of $\sigma()$.

A *tree transformation* is a total function $\tau : T\langle\Sigma\rangle \longrightarrow T\langle\Delta\rangle$ where Σ and Δ are ranked alphabets.

The following functions are defined inductively on the structure of trees in $T\langle\Sigma\rangle(S)$:

- $height : T\langle\Sigma\rangle(S) \longrightarrow I\!N$ delivers the *height of a tree* $t \in T\langle\Sigma\rangle(S)$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $height(\sigma(t_1, \ldots, t_n)) = 1 + max(\{height(t_i) \mid i \in [n]\})$.
  If $t \in S$, then $height(t) = 1$.
- $size_{S'} : T\langle\Sigma\rangle(S) \longrightarrow I\!N$ delivers the *size of a tree* $t \in T\langle\Sigma\rangle(S)$ *with respect to a subset* $S' \subseteq \Sigma \cup S$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $size_{S'}(\sigma(t_1, \ldots, t_n)) = 1 + \sum_{i \in [n]}(size_{S'}(t_i))$, if $\sigma \in S'$,
  $size_{S'}(\sigma(t_1, \ldots, t_n)) = \sum_{i \in [n]}(size_{S'}(t_i))$, if $\sigma \notin S'$.
  If $t \in S \cap S'$, then $size_{S'}(t) = 1$.
  If $t \in S - S'$, then $size_{S'}(t) = 0$.
  If $S' = \Sigma \cup S$, then we abbreviate $size_{S'}$ by $size$.
- $paths : T\langle\Sigma\rangle(S) \longrightarrow \mathcal{P}(I\!N^*)$ delivers the *set of paths of a tree* $t \in T\langle\Sigma\rangle(S)$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $paths(\sigma(t_1, \ldots, t_n)) = \{\varepsilon\} \cup \{p \mid p = ip', i \in [n], p' \in paths(t_i)\}$.
  If $t \in S$, then $paths(t) = \{\varepsilon\}$.
- $label : T\langle\Sigma\rangle(S) \times I\!N^* - \rightarrow \Sigma \cup S$ delivers the *label of the node of a tree* $t \in T\langle\Sigma\rangle(S)$ *reached by a path* $p \in paths(t)$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $label(\sigma(t_1, \ldots, t_n), p) = \sigma$, if $p = \varepsilon$,
  $label(\sigma(t_1, \ldots, t_n), p) = label(t_i, p')$, if $p = ip'$ for some $i \in [n]$.
  If $t \in S$, then $label(t, \varepsilon) = t$.

- $subtree : T\langle\Sigma\rangle(S) \times I\!N^* -\!\!\rightarrow T\langle\Sigma\rangle(S)$ delivers the *subtree of a tree* $t \in T\langle\Sigma\rangle(S)$ *reached by a path* $p \in paths(t)$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $subtree(\sigma(t_1, \ldots, t_n), p) = \sigma(t_1, \ldots, t_n)$, if $p = \varepsilon$,
  $subtree(\sigma(t_1, \ldots, t_n), p) = subtree(t_i, p')$, if $p = ip'$ for some $i \in [n]$.
  If $t \in S$, then $subtree(t, \varepsilon) = t$.

- $repl : T\langle\Sigma\rangle(S) \times I\!N^* \times T\langle\Sigma\rangle(S) -\!\!\rightarrow T\langle\Sigma\rangle(S)$ delivers the *tree obtained from a tree* $t \in T\langle\Sigma\rangle(S)$ *by replacing the subtree reached by a path* $p \in paths(t)$*, by another tree* $t' \in T\langle\Sigma\rangle(S)$.
  If $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma^{(n)}$, $n \geq 0$, and $t_1, \ldots, t_n \in T\langle\Sigma\rangle(S)$, then
  $repl(\sigma(t_1, \ldots, t_n), p, t') = t'$, if $p = \varepsilon$,
  $repl(\sigma(t_1, \ldots, t_n), p, t') = \sigma(t_1, \ldots, repl(t_i, p', t'), \ldots, t_n)$, if $p = ip'$ for some $i \in [n]$.
  If $t \in S$, then $repl(t, \varepsilon, t') = t'$.
  In the following we use the more convenient notation $t[p \leftarrow t']$ instead of $repl(t, p, t')$.

For every tree $t \in T\langle\Sigma\rangle(S)$ and for every path $p \in paths(t)$, the path p determines exactly one node of t. This node will be denoted by $node(t, p)$.

For every tree $t \in T\langle\Sigma\rangle(S)$ and for every path $p \in paths(t)$ we define the following subset $P(t, p)$ of $paths(t)$:

$$P(t, p) = \{p\} \cdot paths(subtree(t, p))$$

In Figure 1 (a) we have hatched that part of t which consists of nodes that are determined by the paths in $P(t, p)$.

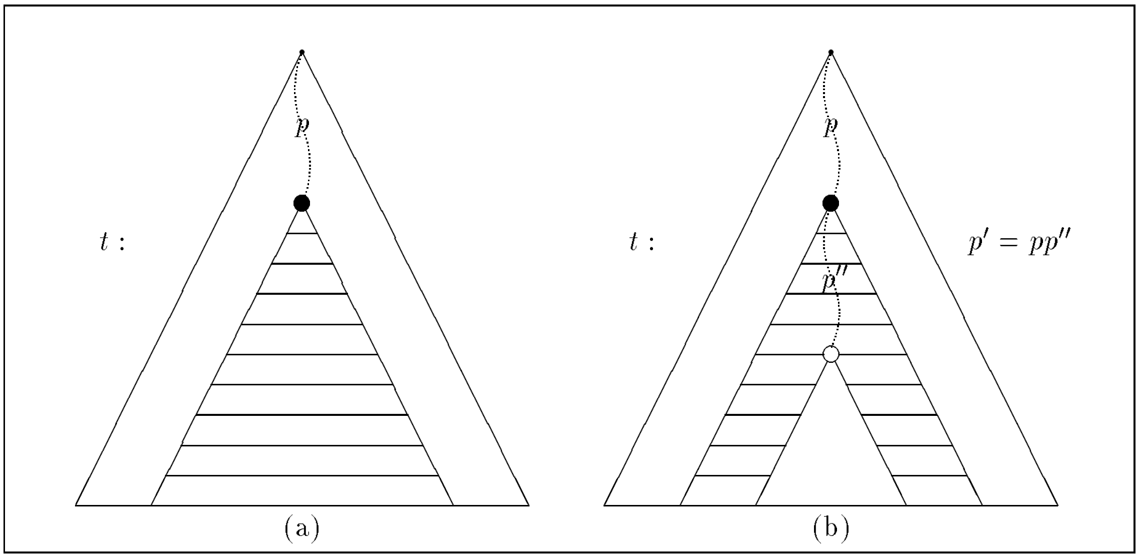

Figure 1: The subsets $P(t, p)$ and $P(t, p - p')$ with $p' = pp''$ of $paths(t)$.

For every tree $t \in T\langle\Sigma\rangle(S)$ and for every paths $p, p' \in paths(t)$, such that there is a string p'' with $p' = pp''$, we use the following abbreviation:

$$P(t, p - p') = P(t, p) - P(t, p')$$

Note that for every $t \in T\langle\Sigma\rangle(S)$ and $p \in paths(t)$ the equation $P(t, p - p) = \emptyset$ holds.

Figure 1 (b) shows that part of t which consists of nodes that are determined by the paths in $P(t, p - p')$. Note here that the path p leading to the root of $subtree(t, p)$ is included into $P(t, p - p')$ and that the path p' leading to the root of $subtree(t, p')$ is excluded from $P(t, p - p')$. In the figure this fact is expressed by a filled and a non–filled cycle, respectively.

In this paper we use a more general notion of pattern than usual and than it was defined in [KV94a]. This generalization is necessary to handle the special properties of macro tree transducers in the pumping lemma. The definition of patterns is followed by an example which is illustrated by a figure.

Let Σ be a ranked alphabet, $t \in T\langle\Sigma\rangle$, let $n \in I\!N$, and let U be another ranked alphabet with $U \cap \Sigma = \emptyset$. A tree $t' \in T\langle\Sigma \cup U\rangle(Y_n)$ is called a *pattern in t*, if there is a ranked alphabet $V = \{v\}$ with $v \notin \Sigma$ and $rank(v) = n$, there is a tree $t'' \in T\langle\Sigma \cup \{v\}\rangle$, and for every $u \in U$ there is a tree $t_u \in T\langle\Sigma\rangle(Y_{rank(u)})$, such that

$t = pat_2(pat_1(t''))$, where

$$\begin{array}{l}
pat_1 : T\langle\Sigma \cup \{v\}\rangle \longrightarrow T\langle\Sigma \cup U\rangle \text{ is defined by:}\\
\text{For every } k \geq 0, \sigma \in \Sigma^{(k)}, \text{ and } t_1, \ldots, t_k \in T\langle\Sigma \cup \{v\}\rangle,\\
\quad pat_1(\sigma(t_1, \ldots, t_k)) \;=\; \sigma(pat_1(t_1), \ldots, pat_1(t_k)).\\
\text{For every } t_1, \ldots, t_n \in T\langle\Sigma \cup \{v\}\rangle,\\
\quad pat_1(v(t_1, \ldots, t_n)) \;=\; t'[y_j/pat_1(t_j)\ ;\ j \in [n]].
\end{array}$$

and

$$\begin{array}{l}
pat_2 : T\langle\Sigma \cup U\rangle \longrightarrow T\langle\Sigma\rangle \text{ is defined by:}\\
\text{For every } k \geq 0, \sigma \in \Sigma^{(k)}, \text{ and } t_1, \ldots, t_k \in T\langle\Sigma \cup U\rangle,\\
\quad pat_2(\sigma(t_1, \ldots, t_k)) \;=\; \sigma(pat_2(t_1), \ldots, pat_2(t_k)).\\
\text{For every } u \in U \text{ and } t_1, \ldots, t_{rank(u)} \in T\langle\Sigma \cup U\rangle,\\
\quad pat_2(u(t_1, \ldots, t_{rank(u)})) \;=\; t_u[y_j/pat_2(t_j)\ ;\ j \in [rank(u)]].
\end{array}$$

The simpler definition of patterns in [KV94a] fits into this definition with $n = 0$ (and thus $t' \in T\langle\Sigma \cup U\rangle$) and $rank(u) = 0$ for every $u \in U$ (and thus $t_u \in T\langle\Sigma\rangle$ for every $u \in U$). Then we simply have $t = pat_2(pat_1(t'')) = t''[v/t'[u/t_u\ ;\ u \in U]]$.

In the following example we can see that a pattern t' in a tree t does not need to represent a "connected piece" of t. It can cover nodes of t which are not directly related. Additionally, "parts" of t', which only consist of symbols from Σ, may occur an arbitrary number of times in t. This depends on the possibility that symbols from Y can occur arbitrarily often in a tree t_u with $u \in U$. Thus our notion of pattern is very general.

Example 2.1 Let $\Sigma = \{\alpha^{(0)}, \beta^{(0)}, \gamma^{(2)}, \delta^{(2)}, \sigma^{(2)}\}$, $t = \gamma(\sigma(\alpha, \delta(\delta(\beta, \alpha), \beta)), \alpha) \in T\langle\Sigma\rangle$, $n = 1$, and $U = \{u^{(2)}\}$.

The tree $t' = \sigma(y_1, u(y_1, \beta)) \in T\langle\Sigma \cup U\rangle(\{y_1\})$ is a pattern in t, since there is the ranked alphabet $V = \{v^{(n)}\} = \{v^{(1)}\}$ and there are the trees $t'' = \gamma(v(\alpha), \alpha) \in T\langle\Sigma \cup \{v\}\rangle$ and $t_u = \delta(\delta(y_2, y_1), y_2) \in T\langle\Sigma\rangle(Y_2)$, such that

$$
\begin{aligned}
& pat_2(pat_1(t'')) \\
= \; & pat_2(pat_1(\gamma(v(\alpha), \alpha))) \\
= \; & pat_2(\gamma(pat_1(v(\alpha)), \alpha)) \\
= \; & pat_2(\gamma(\sigma(pat_1(\alpha), u(pat_1(\alpha), \beta)), \alpha)) \\
= \; & pat_2(\gamma(\sigma(\alpha, u(\alpha, \beta)), \alpha)) \\
= \; & \gamma(\sigma(\alpha, pat_2(u(\alpha, \beta))), \alpha) \\
= \; & \gamma(\sigma(\alpha, \delta(\delta(pat_2(\beta), pat_2(\alpha)), pat_2(\beta))), \alpha) \\
= \; & \gamma(\sigma(\alpha, \delta(\delta(\beta, \alpha), \beta)), \alpha) \\
= \; & t.
\end{aligned}
$$

In Figure 2 we show the trees t'', t', t_u, $pat_1(t'')$, and $t = pat_2(pat_1(t''))$. We put the nodes of the pattern t', which are labeled by symbols of Σ, into a cycle. Those nodes of $pat_1(t'')$ and t, which are labeled by symbols of Σ and which originate from nodes of the pattern t' are also put into a cycle. □

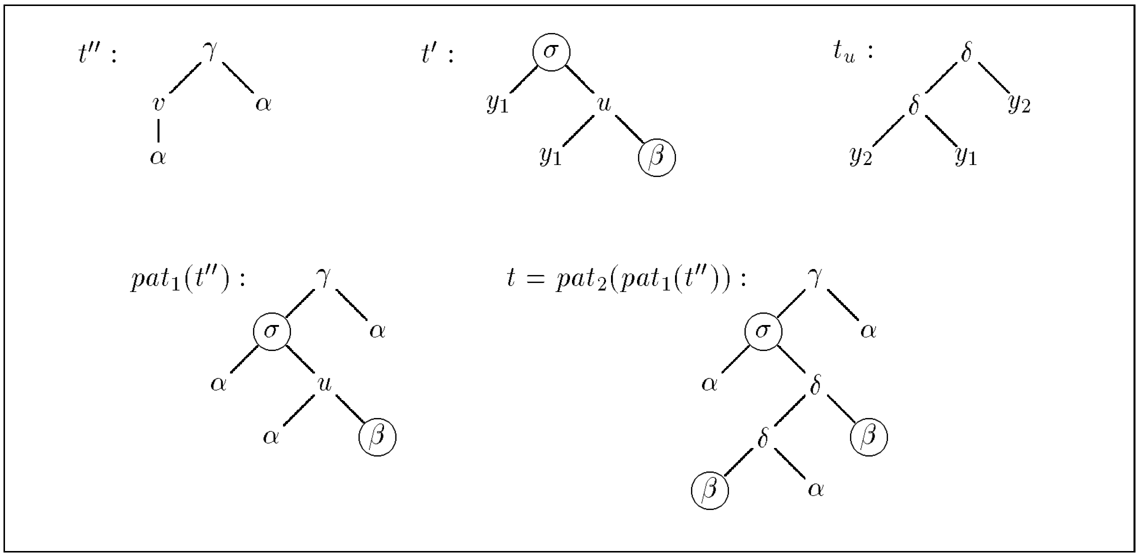

Figure 2: Pattern t' in the tree $t = pat_2(pat_1(t''))$.

3 Macro Tree Transducers

This section defines macro tree transducers, their derivation relations, certain restrictions on them, and their output languages. Additionally, we recall results from the literature and already prove some helping statements.

3.1 Definition and Properties

Macro tree transducers are presented for example in [Eng80, EV85] as concept for studying syntax–directed semantics. A macro tree transducer can be seen as an extended scheme of primitive recursion over trees, allowing simultaneous recursion and nested function calls in parameter positions. For purposes concerning the applications of the pumping lemma, our definition of macro tree transducers differs a little bit from the original one. This difference concerns the initial function which we replace by a so called initial right–hand side.

A ranked alphabet F of functions (or states) is the first component in the definition of a macro tree transducer M. We specify a ranked input alphabet Σ of input symbols. M takes an argument e where e is a tree over Σ, called *input tree* (or *control tree* in analogy to [KV94a]) and produces an *output tree* which is built up over a ranked alphabet Δ of output symbols. Roughly speaking, the derivations of M will start with an initial right–hand side $\tilde{\rho}$. The kernel of the definition of a macro tree transducer is the finite set R of rewrite rules. For every function and for every input symbol there is exactly one rule. Thus we only consider total deterministic macro tree transducers. The possible right–hand sides of rules are fixed at the end of the definition.

Definition 3.1 A *macro tree transducer* is a tuple $(F, \Delta, \Sigma, \tilde{\rho}, R)$ where

- F is the ranked alphabet of *functions*; for every $f \in F$, $rank_F(f) \geq 1$.
- Δ is the nonempty ranked alphabet of *output symbols* with $F \cap \Delta = \emptyset$.
- Σ is the nonempty ranked alphabet of *input symbols* with $F \cap \Sigma = \emptyset$.
- $\tilde{\rho} \in RHS(F, \Delta, \{x_1\}, \emptyset)$ is the *initial right–hand side*.
- R is the finite set of *rules*:

  For every $n \geq 0$, $f \in F^{(n+1)}$, $k \geq 0$, and $\sigma \in \Sigma^{(k)}$, the set R contains exactly one rule of the form
  $$f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n) \to \rho$$
  with $\rho \in RHS(F, \Delta, X_k, Y_n)$.

For every $k \geq 0$ and $n \geq 0$, the set of *right–hand sides over* F, Δ, X_k, *and* Y_n, denoted by $RHS(F, \Delta, X_k, Y_n)$, is the smallest subset RHS of $T\langle F \cup \Delta\rangle(X_k \cup Y_n)$ such that:

(i) For every $r \geq 0$, $\delta \in \Delta^{(r)}$, and $\rho_1, \ldots, \rho_r \in RHS$, the tree $\delta(\rho_1, \ldots, \rho_r) \in RHS$.

(ii) For every $m \geq 0$, $f \in F^{(m+1)}$, $x_i \in X_k$, and $\rho_1, \ldots, \rho_m \in RHS$, the tree $f(x_i, \rho_1, \ldots, \rho_m) \in RHS$.

(iii) For every $i \in [n]$, the tree $y_i \in RHS$. $\Box$

Our choice, to change the definition of a macro tree transducer replacing the initial function (of rank 1) by an initial right–hand side is founded by the applications of the pumping lemma presented in Section 6. On the one hand, often the initial function f_{in} is only used to call directly other functions which perform the "real work" on the input trees. Thus, the initial function f_{in} should not be counted, when we shall talk about classes of tree transformations induced by macro tree transducers with bounded number of functions. Instead of f_{in} the initial right–hand side $\tilde{\rho}$ has been introduced and the "real work" can start directly. On the other hand, if the initial function f_{in} also performs "real work" on the input trees, then the definition $\tilde{\rho} = f_{in}(x_1)$ is appropriate and f_{in} would be counted.

For a macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ and for every $n \geq 0$, $f \in F^{(n+1)}$, $k \geq 0$, and $\sigma \in \Sigma^{(k)}$, we call the rule of R with left–hand side $f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n)$ the (f, σ)–rule. The right–hand side of this rule is denoted by $rhs(f, \sigma)$. If $m \geq 0$, $f' \in F^{(m+1)}$, $i \in [k]$, $\rho_1, \ldots, \rho_m \in RHS(F, \Delta, X_k, Y_n)$, and $f'(x_i, \rho_1, \ldots, \rho_m)$ occurs in $rhs(f, \sigma)$, then (f', x_i) is called a *function occurrence* of $rhs(f, \sigma)$. The first argument of every function is called the *recursion argument*. The other arguments are called *context parameters*. Additionally we use the following abbreviations:

$$\begin{array}{lcl} maxrank_{M,\Delta} & = & max(\{rank_\Delta(\delta) \mid \delta \in \Delta\}) \\ & & \text{(maximum rank of output symbols)}, \\ maxheight_{M,R} & = & max(\{height(\rho) \mid (\lambda \to \rho) \in R\} \cup \{height(\tilde{\rho})\}) \\ & & \text{(maximum height of right–hand sides)}. \end{array}$$

Example 3.2 We define the macro tree transducer $M_1 = (F_1, \Delta_1, \Sigma_1, \tilde{\rho}_1, R_1)$ with:

$$\begin{array}{lcl} F_1 & = & \{f^{(2)}, g^{(2)}\}, \\ \Delta_1 & = & \{T^{(3)}, A^{(1)}, B^{(1)}, L^{(1)}, R^{(1)}, E^{(0)}\}, \\ \Sigma_1 & = & \{\sigma^{(2)}, \alpha^{(0)}\}, \\ \tilde{\rho}_1 & = & f(x_1, E), \text{ and} \\ R_1 & = & \{ \begin{array}[t]{lcll} f(\sigma(x_1, x_2), y_1) & \to & T(f(x_1, L(y_1)), f(x_2, R(y_1)), g(x_1, g(x_2, E))), & (1) \\ f(\alpha, y_1) & \to & B(y_1), & (2) \\ g(\sigma(x_1, x_2), y_1) & \to & g(x_1, A(g(x_2, y_1))), & (3) \\ g(\alpha, y_1) & \to & A(y_1) \qquad \} & (4) \end{array} \end{array}$$

The macro tree transducer M_1 takes a binary tree e over the ranked alphabet $\Sigma_1 = \{\sigma^{(2)}, \alpha^{(0)}\}$ as argument and it delivers a tree t which has almost the same structure as e. Every node n, which is labeled by a binary input symbol σ, is substituted by a node labeled by a ternary output symbol T, where the additional third subtree is a monadic tree over the ranked alphabet $\{A^{(1)}, E^{(0)}\}$ which is as high as the size of the input subtree rooting at n. Every node n, which is labeled by an input symbol α, is substituted by an encoding of the reverse path from the root of e to n. The encoding of a reverse path is a monadic tree over the ranked alphabet $\{B^{(1)}, L^{(1)}, R^{(1)}, E^{(0)}\}$, where the symbol L (and R) represent the left child (and the right child, respectively) of a node and the symbol B (and E) is the first symbol (and the last symbol, respectively) of each path encoding. In Figure 3 we show an example input tree e and the calculated output tree t. $\square$

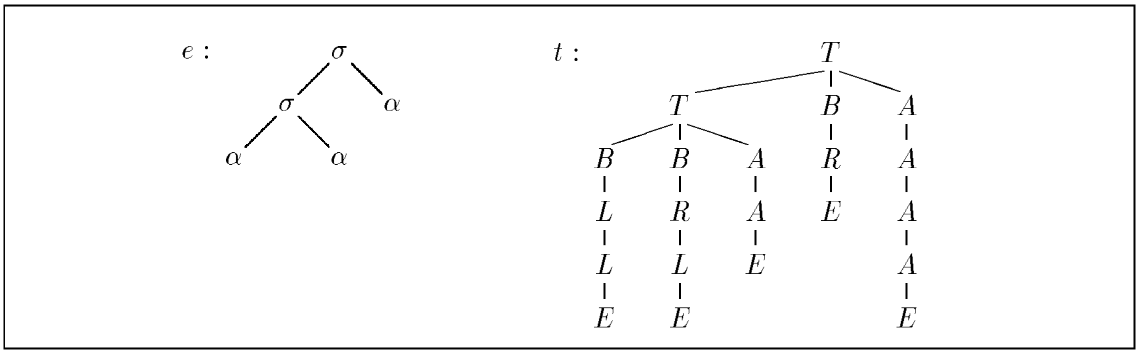

Figure 3: The input tree e and the calculated output tree t.

Now we define a classification scheme for subclasses of macro tree transducers which are induced by their number of functions and by their number of functions with certain ranks, respectively.

Definition 3.3 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer.

- Let $c \in I\!N$. M is a (c)*–tree transducer*, if $card(F) \leq c$.
- Let $k \in I\!N$, for every $i \in [k]$ let $r_i \in I\!N_+$ with $r_j > r_i$, if $i, j \in [k]$ with $j > i$, and for every $i \in [k]$ let $c_i \in I\!N$. M is a $(c_1 * r_1, \ldots, c_k * r_k)$*–tree transducer*, if, for every $i \in [k]$, $card(F^{(r_i)}) \leq c_i$, and for every $r \in I\!N_+ - \{r_i \mid i \in [k]\}$, $card(F^{(r)}) = 0$. $\square$

For every $i \in [k]$ the part $c_i * r_i$ in the notation of a $(c_1 * r_1, \ldots, c_k * r_k)$–tree transducer M means that M has at most c_i functions of rank r_i. Thus $c_i * r_i$ should be seen as an element of the set $I\!N \cdot \{*\} \cdot I\!N_+$ of strings and the symbol $*$ must not be interpreted for example by the multiplication operation.

Note that in the definition of a $(c_1 * r_1, \ldots, c_k * r_k)$–tree transducer M we allow that there are some $i \in [k]$ with $c_i = 0$. The explicit emphasis that M does not have functions with rank r_i is not necessary, because this fact can also be implicitely expressed by omitting the part $0 * r_i$ from the above denotation. But for later purposes it makes sense to allow the explicit emphasis, for example if we want to make a statement for a $(\ldots, c_i * r_i, \ldots)$–tree transducer, where c_i may range over the whole set of natural numbers. However, we try to avoid these superfluous denotation parts. Also note that every (c)–tree transducer is also a $(c + k)$–tree transducer for every $k \geq 0$. Moreover, every $(c_1 * r_1, \ldots, c_k * r_k)$–tree transducer is also a $(w_0, c_1 * r_1, w_1, \ldots, c_k * r_k, w_k)$–tree transducer, where for every $j \in [k] \cup \{0\}$ the part w_j is a sequence of the form $c_{j,1} * r_{j,1}, \ldots, c_{j,n_j} * r_{j,n_j}$ such that (i) $r_{j,1} < r_{j,2} < \ldots < r_{j,n_j}$, (ii) if $j \geq 1$, then $r_j < r_{j,1}$, and (iii) if $j < k$, then $r_{j,n_j} < r_{j+1}$.

Example 3.4 The macro tree transducer M_1 from Example 3.2 can be denoted, e.g., as (2)–tree transducer, as (4)–tree transducer, as $(2*2)$–tree transducer, and as $(1*1, 3*2, 0*4)$–tree transducer. M_1 is, e.g., no (1)–tree transducer and no $(1*2, 2*3)$–tree transducer. □

In the next definition we inductively describe the set of sentential forms of macro tree transducers. Rather than the usual definition, where subtrees of an input tree e occur in sentential forms as recursion argument, we use paths of e instead. This choice makes it possible to determine the exact occurrence of a subtree in an input tree. Thus, for a given input tree e, a sentential form is a tree over functions, output symbols, and paths of e. Moreover, the recursion argument of a function is always a path of e and vice versa a path may only occur in the recursion argument of a function. The input tree e itself will parameterize the derivation relation (cf. Definition 3.6). For later purposes we define the set of sentential forms for an arbitrary subset $P \subseteq paths(e)$ of paths and for a superset $\Delta' \supseteq \Delta$ of output symbols.

Definition 3.5 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer, $e \in T\langle\Sigma\rangle$, $P \subseteq paths(e)$ and Δ' be a ranked alphabet with $\Delta \subseteq \Delta'$. The set of (F, P, Δ')–*sentential forms*, denoted by $SF(F, P, \Delta')$, is the smallest subset SF of $T\langle F \cup \Delta'\rangle(I\!N^*)$ such that:

(i) For every $\delta \in \Delta'^{(r)}$ with $r \geq 0$ and $t_1, \ldots, t_r \in SF$, the tree $\delta(t_1, \ldots, t_r) \in SF$.

(ii) For every $f \in F^{(n+1)}$ with $n \geq 0$, $p \in P$, and $t_1, \ldots, t_n \in SF$, the tree $f(p, t_1, \ldots, t_n) \in SF$. □

If the input tree e is known, a (usual) sentential form sf' with subtrees of e as recursion arguments can be obtained from a sentential form $sf \in SF(F, P, \Delta')$ by $sf' = sf[p/subtree(e, p)\ ;\ p \in P]$. Note that the reverse transformation is impossible, because in general a subtree of e does not determine a unique path of e.

For a macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ and for an input tree $e \in T\langle\Sigma\rangle$, the set of *function occurrences of* e, denoted by $fun(e)$, is the set $\{(f, p) \mid f \in F, p \in paths(e)\}$. If $f(p, t_1, \ldots, t_n) \in SF(F, P, \Delta')$ is a subtree of another sentential form $sf \in SF(F, P, \Delta')$, then (f, p) is called a *function occurrence of* sf.

Now we describe the derivation relation of a macro tree transducer M with respect to an input tree e. For later purposes, we restrict the derivation relation to work only on particular parts of e. That means, we parameterize the derivation relation with a subset $P \subseteq paths(e)$ and only if $p \in P$, then a function occurrence (f, p) of a sentential form can be derived further.

Definition 3.6 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer. Let $e \in T\langle\Sigma\rangle$ and $P \subseteq paths(e)$. Let Δ' be a ranked alphabet with $\Delta \subseteq \Delta'$. The *derivation relation of* M *with respect to* e *and* P, denoted by $\Rightarrow_{M,e,P}$, is a binary relation on $SF(F, paths(e), \Delta')$ defined as follows:

For every $t_1, t_2 \in SF(F, paths(e), \Delta')$, $\quad t_1 \Rightarrow_{M,e,P} t_2$, $\quad$ iff

- there is a $t' \in SF(F, paths(e), \Delta' \cup \{u\})$ in which the 0–ary symbol $u \notin F \cup \Delta'$ occurs exactly once,
- there is an $n \in I\!N$ and a function $f \in F^{(n+1)}$,
- there is a path $p \in P$ with $label(e,p) = \sigma$ for some $\sigma \in \Sigma^{(k)}$ with $k \in I\!N$,
- there is a rule $f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n) \to \rho$ in R, and
- there are $t'_1, \ldots, t'_n \in SF(F, paths(e), \Delta')$,

such that

- $t_1 = t'[u/f(p, t'_1, \ldots, t'_n)]$ and
- $t_2 = t'[u/\rho[x_i/pi\ ;\ i \in [k]][y_i/t'_i\ ;\ i \in [n]]]$. $\square$

If M or e are known from the context, then we drop the corresponding indices from $\Rightarrow$. If $P = paths(e)$, then we drop P.

Example 3.7 Let M_1 be the macro tree transducer defined in Example 3.2 and let $e = \sigma(\sigma(\alpha, \alpha), \alpha)$ be the input tree. We show a derivation of M_1, which starts with the sentential form $\tilde{\rho}_1[x_1/\varepsilon] = f(\varepsilon, E)$, abbreviating $\Rightarrow_{M_1,e,paths(e)}$ by $\Rightarrow$. The number of the applied rule is indicated as a subscript. The input tree and the calculated output tree are also shown in Figure 3.

$$
\begin{array}{ll}
 & f(\varepsilon, E) \\
\Rightarrow_{(1)} & T(f(1, L(E)), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(1)} & T(T(f(11, L(L(E))), f(12, R(L(E))), g(11, g(12, E))), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(2)} & T(T(B(L(L(E))), f(12, R(L(E))), g(11, g(12, E))), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(2)} & T(T(B(L(L(E))), B(R(L(E))), g(11, g(12, E))), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(4)} & T(T(B(L(L(E))), B(R(L(E))), A(g(12, E))), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(4)} & T(T(B(L(L(E))), B(R(L(E))), A(A(E))), f(2, R(E)), g(1, g(2, E))) \\
\Rightarrow_{(2)} & T(T(B(L(L(E))), B(R(L(E))), A(A(E))), B(R(E)), g(1, g(2, E))) \\
\Rightarrow_{(3)} & T(T(B(L(L(E))), B(R(L(E))), A(A(E))), B(R(E)), g(11, A(g(12, g(2, E))))) \\
\Rightarrow^{+} & T(T(B(L(L(E))), B(R(L(E))), A(A(E))), B(R(E)), A(A(A(A(E)))))
\end{array}
$$
$\square$

The (usual) derivation relation with subtrees of e as recursion arguments in sentential forms can be obtained in the same manner as it was explained after Definition 3.5. Thus every derivation with paths of e as recursion arguments in sentential forms corresponds to exactly one derivation with subtrees of e as recursion arguments in sentential forms (, but not vice versa). Thus we can use results from the literature about derivations of macro tree transducers.

According to [EV85], total deterministic macro tree transducers compute total functions from input trees to output trees. This is due to the fact that their derivation relations are confluent and noetherian and thus every sentential form has a unique normal form. This result carries over to the macro tree transducers presented in this paper, because here we only have the difference that the derivations start with more general sentential forms. These are obtained from the initial right–hand side $\tilde{\rho}$ by the substitution of the path ε leading to the root of a concrete input tree for the variable x_1 (cf. Definition 3.8). This is the basis for the definition of the tree transformation which is computed by a macro tree transducer.

Definition 3.8 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro 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 $e \in T\langle\Sigma\rangle$,

$$\tau(M)(e) = nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon]). \qquad \square$$

In the following, the tree $\tilde{\rho}[x_1/\varepsilon]$ is called the *initial sentential form of* M.

The following lemma relates the heights of input trees and corresponding output trees of macro tree transducers. It is an adaptation of Theorem 3.24 in [EV85] to our definition of macro tree transducers. For later purposes we state in the first part of the lemma a general result for derivations on subtrees of input trees and specialize this result in the second part to derivations which start with the initial sentential form of a macro tree transducer.

Lemma 3.9 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer and let $e \in T\langle\Sigma\rangle$.

(a) If $p \in paths(e)$, $\Delta' \supseteq \Delta$, $t_1 \in SF(F, \{p\}, \Delta')$, and $t_2 = nf(\Rightarrow_{M,e,P(e,p)}, t_1)$, then

$$height(t_2) \leq (max(\{maxheight_{M,R}, height(t_1)\}))^{height(subtree(e,p))+1}.$$

(b) If $t = nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon])$, then

$$height(t) \leq (maxheight_{M,R})^{height(e)+1}.$$

Proof. (a) Assume that first an imaginary rule application delivers the sentential form $t_1[p/\varepsilon]$ and then the computation is simulated on the input tree $subtree(e, p)$. This can be realized by transforming M into a (conventional) macro tree transducer $M' = (F \cup \{f_{in}\}, \Delta, \Sigma \cup \{root\}, f_{in}(x_1), R')$ where $f_{in} \notin F$, $root \notin \Sigma$ and R' contains the rules of R and the rule $f_{in}(root(x_1)) \rightarrow t_1[p/x_1]$. Then $nf(\Rightarrow_{M',root(subtree(e,p))}, f_{in}(\varepsilon))$ must be calculated instead of $nf(\Rightarrow_{M,e,P(e,p)}, t_1)$, yielding the same result t_2, because both normal forms only consist of symbols from Δ'. Since $max(\{maxheight_{M,R}, height(t_1)\})$ is the maximum height of right–hand sides of rules in R' and thus is the constant c of Theorem 3.24 in [EV85], this theorem delivers

$$\begin{aligned} & height(t_2) \\ \leq\ & (max(\{maxheight_{M,R}, height(t_1)\}))^{height(root(subtree(e,p)))} \\ =\ & (max(\{maxheight_{M,R}, height(t_1)\}))^{height(subtree(e,p))+1}. \end{aligned}$$

(b) This statement follows from Statement (a) with $p = \varepsilon$, $t_1 = \tilde{\rho}[x_1/\varepsilon]$, and $t_2 = t$, because $height(t_1) = height(\tilde{\rho}[x_1/\varepsilon]) = height(\tilde{\rho}) \leq maxheight_{M,R}$. $\Box$

For a given input tree e and for a given path p of e we calculate the normal form nf of $\tilde{\rho}[x_1/\varepsilon]$ with respect to the set $P(e, \varepsilon - p)$ of paths, i.e. the derivation is blocked at every function occurrence $(f, p) \in fun(e)$. Then we can define the following objects inductively on nf:

- The sequence $funseq(e, p)$ is the sequence of those functions f, for which a function occurrence $(f, p) \in fun(e)$ is encountered on a depth–first left–to–right tree traversal of nf.
- The set $funset(e, p)$ is the set of those functions f, which occur in $funseq(e, p)$. Thus it is the set of functions for which there is a function occurrence $(f, p) \in fun(e)$ in a sentential form during an arbitrary derivation starting from the initial sentential form. This notion is similar to the concepts of *state–set* in [ERS80] and of *attset* in [KV94a]. In contrast to the definitions presented there, we directly define $funset(e, p)$ for an input tree e and not via a concrete derivation.
- The natural number $funnum(e, p)$ is the length of $funseq(e, p)$.
- The multiset $funmulti(e, p)$ is similar to $funset(e, p)$ taking multiple function occurrences into account.

Definition 3.10 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer. Let $e \in T\langle\Sigma\rangle$ and $p \in paths(e)$. Then we define

- the *function–sequence* $funseq(e, p) \in F^*$ *of* e *and* p by

  $funseq(e, p) = seq_p(nf(\Rightarrow_{e,P(e,\varepsilon-p)}, \tilde{\rho}[x_1/\varepsilon]))$, where

  $seq_p : SF(F, \{p\}, \Delta) \longrightarrow F^*$ is defined as follows:
  For every $\delta \in \Delta^{(r)}$, $r \geq 0$, and $t_1, \ldots, t_r \in SF(F, \{p\}, \Delta)$,
  $$seq_p(\delta(t_1, \ldots, t_r)) = seq_p(t_1) \circ \ldots \circ seq_p(t_r).$$
  For every $f \in F^{(n+1)}$, $n \geq 0$, and $t_1, \ldots, t_n \in SF(F, \{p\}, \Delta)$,
  $$seq_p(f(p, t_1, \ldots, t_n)) = f \circ seq_p(t_1) \circ \ldots \circ seq_p(t_n).$$

- the *function–set* $funset(e, p) \in \mathcal{P}(F)$ *of* e *and* p by

  $funset(e, p) = set(funseq(e, p))$, where

  $set : F^* \longrightarrow \mathcal{P}(F)$ is defined as follows:
  $$set(\varepsilon) = \emptyset.$$
  For every $f \in F$ and $s \in F^*$,
  $$set(fs) = \{f\} \cup set(s).$$

- the *function-number* $funnum(e,p) \in I\!N$ *of* e *and* p by

  $funnum(e,p) = |funseq(e,p)|$

- the *function-multiset* $funmulti(e,p) : F \longrightarrow I\!N$ *of* e *and* p by

  $funmulti(e,p) = multi(funseq(e,p))$, where

  $multi : F^* \longrightarrow (F \longrightarrow I\!N)$ is defined as follows:
  For every $f \in F$, $\quad multi(\varepsilon)(f) = 0.$
  For every $f \in F$ and $s \in F^*$, $\quad multi(fs)(f) = multi(s)(f) + 1.$
  For every $f, f' \in F$ with $f \neq f'$ and $s \in F^*$, $\quad multi(f's)(f) = multi(s)(f).$ □

This kind of definition, restricting the derivation relation to the set $P(e, \varepsilon - p)$ of paths, would not have been applicable to attributed tree transducers in [KV94a], because a path p may exist, such that an attribute occurrence $a'(p)$ depends on another attribute occurrence $a(p)$ with $a \neq a'$. The way of definition there is not suitable here, because it would make it difficult to take the number of equal function occurrences into account.

Note further that in Definition 3.10 it would have been sufficient to block the derivation at the path p in e by excluding only the path p from the set $paths(e)$ instead of excluding all paths with prefix p.

Example 3.11 Let M_1 be the macro tree transducer defined in Example 3.2 and let $e = \sigma(\sigma(\alpha, \alpha), \alpha)$ be the input tree. Then we have for example:

$$
\begin{array}{lrcl}
(1) & funseq(e,\varepsilon) & = & seq_\varepsilon(nf(\Rightarrow_{e,P(e,\varepsilon-\varepsilon)}, f(\varepsilon, E))) \\
 & & = & seq_\varepsilon(f(\varepsilon, E)) \\
 & & = & f, \\
 & funset(e,\varepsilon) & = & \{f\}, \\
 & funnum(e,\varepsilon) & = & 1,
\end{array}
$$

$$
\begin{array}{lrcl}
(2) & funseq(e,11) & = & seq_{11}(nf(\Rightarrow_{e,P(e,\varepsilon-11)}, f(\varepsilon, E))) \\
 & & = & seq_{11}(T(T(f(11, L(L(E))), B(R(L(E))), g(11, A(E))), \\
 & & & \qquad B(R(E)), g(11, A(A(A(E)))))) \\
 & & = & f\,g\,g, \\
 & funset(e,11) & = & \{f, g\}, \\
 & funnum(e,11) & = & 3, \\
 & funmulti(e,11)(g) & = & multi(f\,g\,g)(g) \\
 & & = & multi(g\,g)(g) \\
 & & = & multi(g)(g) + 1 \\
 & & = & multi(\varepsilon)(g) + 2 \\
 & & = & 2.
\end{array}
$$

□

3.2 Producing, Nondeleting, and Noncopying Macro Tree Transducers

The pumping lemmata in the next sections are only valid for special kinds of macro tree transducers. In the following definition we introduce the concepts of producing (the right–hand side of every rule contains at least one output symbol), nondeleting (every variable which occurs in the left–hand side of a rule occurs at least once in the corresponding right–hand side), and noncopying (every variable which occurs in the left–hand side of a rule and which represents a context parameter, occurs at most once in the corresponding right–hand side) macro tree transducers.

Definition 3.12 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer. M is

- *producing*, if, for every rule $\lambda \to \rho$ in R, the size of ρ with respect to Δ is at least 1, i.e., $size_\Delta(\rho) \geq 1$,
- *nondeleting*, if,
  - for every rule $f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n) \to \rho$ in R, the sizes $size_{\{x_i\}}(\rho) \geq 1$ for every $i \in [k]$ and $size_{\{y_i\}}(\rho) \geq 1$ for every $i \in [n]$, and
  - $size_{\{x_1\}}(\tilde{\rho}) \geq 1$,
- *noncopying* (*in parameter positions*), if, for every rule $f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n) \to \rho$ in R, the sizes $size_{\{y_i\}}(\rho) \leq 1$ for every $i \in [n]$. □

The additional condition $size_{\{x_1\}}(\tilde{\rho}) \geq 1$ for a nondeleting macro tree transducer implies $\tilde{\rho} \notin T\langle\Delta\rangle$ and thus there is a function occurrence (f, ε) in $\tilde{\rho}[x_1/\varepsilon]$ and every input tree really is inspected.

The nondeleting–conditions $size_{\{x_i\}}(\rho) \geq 1$ and $size_{\{y_i\}}(\rho) \geq 1$ for every rule $\lambda \to \rho$ and for every x_i and y_i, respectively, in λ imply that every subtree of an input tree is inspected. This corresponds to the visiting–condition in the sense of [KV94a], which requires in the notation of this paper $funset(e, p) \neq \emptyset$ for every input tree e and for every path $p \in paths(e)$ (cf. also Lemma 3.15).

In the rest of this paper we always mean producing and nondeleting macro tree transducers, when we talk about macro tree transducers. The noncoying–condition will be used in a later phase of this paper to prove a certain detail of the pumping lemma. We explain later that the noncopying–condition is no restriction in the context of our applications presented in Section 6.

We denote the *classes of tree transformations computed* by (producing and nondeleting) macro tree transducers, (c)–tree transducers, and $(c_1 * r_1, \ldots, c_k * r_k)$–tree transducers by MT, $MT_{(c)}$, and $MT_{(c_1*r_1,\ldots,c_k*r_k)}$, respectively. Note that there are different possibilities to denote the same class of tree transformations, depending on the possibility to emphasize explicitely that there are no functions with a certain rank (cf. also the explanation following Definition 3.3).

Observation 3.13 In Definition 3.1 we also allow $F = \emptyset$ for a (deleting) macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$. The further consequences of $F = \emptyset$ are $R = \emptyset$ and $\tilde{\rho} \in T\langle\Delta\rangle$. Thus a (deleting) macro tree transducer without functions computes a constant tree transformation. Additionally, $MT_{(0)}$ and $MT_{()}$ denote the class of tree transformations of producing and nondeleting macro tree transducers without functions. But this class is empty, because the nondeleting–condition implies $size_{\{x_1\}}(\tilde{\rho}) \geq 1$ and thus $\tilde{\rho} \notin T\langle\Delta\rangle$ in contradiction to the consequence $\tilde{\rho} \in T\langle\Delta\rangle$ of $F = \emptyset$. Since there exist producing and nondeleting macro tree transducers with exactly one function, the class $MT_{(1)} \neq \emptyset$ and for every $r \in I\!N_+$ the class $MT_{(1*r)} \neq \emptyset$, and we obtain the following two easy observations

- $MT_{(0)} \subset MT_{(1)}$ and
- $MT_{()} \subset MT_{(1*r)}$ for every $r \in I\!N_+$,

which are the first steps of the hierarchies that are presented in Section 6. □

The nondeleting–condition $size_{\{x_1\}}(\tilde{\rho}) \geq 1$ implies $height(\tilde{\rho}) \geq 2$, because an occurrence of x_1 in $\tilde{\rho}$ can only be the first argument of an occurrence of a function. Thus, for every nondeleting macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ we have $maxheight_{M,R} \geq 2$.

In the next sections we often start derivations with a sentential form t_1 with $height(t_1) = 2$. Then $height(t_1) \leq maxheight_{M,R}$ holds and we can specialize Lemma 3.9 to the following observation:

Observation 3.14 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a nondeleting macro tree transducer, let $e \in T\langle\Sigma\rangle$, $p \in paths(e)$, $\Delta' \supseteq \Delta$, $t_1 \in SF(F, \{p\}, \Delta')$ with $height(t_1) = 2$, and $t_2 = nf(\Rightarrow_{M,e,P(e,p)}, t_1)$, then

$$height(t_2) \leq (maxheight_{M,R})^{height(subtree(e,p))+1}.$$

□

The following lemma presents a result for nondeleting macro tree transducers which we shall need later. It states the fact that for every input tree e and for every path p of e, the function–number of e and p is at least 1.

Lemma 3.15 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a nondeleting macro tree transducer. Let $e \in T\langle\Sigma\rangle$ and $p \in paths(e)$. Then,

$$funnum(e, p) \geq 1.$$

Proof. We prove $funnum(e, pj) \geq funnum(e, p)$ for every $p, pj \in paths(e)$ with $j \in I\!N_+$. Then the statement of the lemma can be deduced with the transitivity of $\geq$ from $funnum(e, \varepsilon) = |seq_\varepsilon(nf(\Rightarrow_{e,P(e,\varepsilon-\varepsilon)}, \tilde{\rho}[x_1/\varepsilon]))| = |seq_\varepsilon(\tilde{\rho}[x_1/\varepsilon])| \geq 1$. (In fact, this proof even induces the result that the function–number cannot decrease on an arbitrary path from the root of e to a leaf node.)

First we have to prove the following Statement (1) by structural induction on the set $SF(F, \{p\}, \Delta)$. Recall the definition of seq_p from Definition 3.10.

(1) For every $t \in SF(F, \{p\}, \Delta)$, $\quad |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t))| \geq |seq_p(t)|$.

There are two cases:

Case I: For every $n \geq 0$, $\delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in SF(F, \{p\}, \Delta)$, such that $t = \delta(t_1, \ldots, t_n)$,

$$\begin{array}{ll}
 & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, \delta(t_1, \ldots, t_n)))| \\
= & |seq_{pj}(\delta(nf(\Rightarrow_{e,P(e,p-pj)}, t_1), \ldots, nf(\Rightarrow_{e,P(e,p-pj)}, t_n)))| \\
= & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_1)) \circ \ldots \circ seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_n))| \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Definition of } seq_{pj}) \\
= & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_1))| + \ldots + |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_n))| \\
\geq & |seq_p(t_1)| + \ldots + |seq_p(t_n)| \qquad\qquad\qquad\qquad \text{(Induction Hypothesis)} \\
= & |seq_p(t_1) \circ \ldots \circ seq_p(t_n)| \\
= & |seq_p(\delta(t_1, \ldots, t_n))| \qquad\qquad\qquad\qquad\qquad \text{(Definition of } seq_p)
\end{array}$$

Case II: Let $label(e, p) = \sigma \in \Sigma$. For every $n \geq 0$, $f \in F^{(n+1)}$, and $t_1, \ldots, t_n \in SF(F, \{p\}, \Delta)$, such that $t = f(p, t_1, \ldots, t_n)$,

$$\begin{array}{ll}
 & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, f(p, t_1, \ldots, t_n)))| \\
= & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, rhs(f, \sigma)[x_i/pi \;;\; i \in [rank(\sigma)]][y_i/t_i \;;\; i \in [n]]))| \\
 & \quad \text{(Application of the } (f, \sigma)\text{–rule)} \\
\geq & 1 + |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_1))| + \ldots + |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, t_n))| \\
 & \quad (M \text{ nondeleting } (*)) \\
\geq & 1 + |seq_p(t_1)| + \ldots + |seq_p(t_n)| \\
 & \quad \text{(Induction Hypothesis)} \\
= & |f \circ seq_p(t_1) \circ \ldots \circ seq_p(t_n)| \\
= & |seq_p(f(p, t_1, \ldots, t_n))| \\
 & \quad \text{(Definition of } seq_p)
\end{array}$$

$(*)$: Since M is nondeleting, $rhs(f, \sigma)$ contains $y_1, \ldots, y_n$ and there is an $f' \in F$, such that $rhs(f, \sigma)$ contains at least one function occurrence (f', x_j). Thus, $rhs(f, \sigma)[x_i/pi \;;\; i \in [rank(\sigma)]][y_i/t_i \;;\; i \in [n]]$ contains $t_1, \ldots, t_n$ and at least one function occurrence (f', pj).

With the help of Statement (1) we can prove as follows:

$$\begin{array}{lll}
 & funnum(e, pj) & \\
= & |seq_{pj}(nf(\Rightarrow_{e,P(e,\varepsilon-pj)}, \tilde{\rho}[x_1/\varepsilon]))| & \text{(Definition of } funnum(e, pj)) \\
= & |seq_{pj}(nf(\Rightarrow_{e,P(e,p-pj)}, nf(\Rightarrow_{e,P(e,\varepsilon-p)}, \tilde{\rho}[x_1/\varepsilon])))| & \\
\geq & |seq_p(nf(\Rightarrow_{e,P(e,\varepsilon-p)}, \tilde{\rho}[x_1/\varepsilon]))| & \text{(Statement (1))} \\
= & funnum(e, p) & \text{(Definition of } funnum(e, p)) \quad \square
\end{array}$$

3.3 Output Languages of Macro Tree Transducers

The pumping lemmata which we introduce in the next sections, deal with output languages of tree transformations of macro tree transducers. The output language of a tree transformation τ is defined as the range of τ.

Definition 3.16 Let Σ and Δ be ranked alphabets and $\tau : T\langle\Sigma\rangle \longrightarrow T\langle\Delta\rangle$ be a tree transformation. The *output language of* τ, denoted by $L_{out}(\tau)$ is defined as follows:

$$L_{out}(\tau) = \{t \in T\langle\Delta\rangle \mid \text{there is an } e \in T\langle\Sigma\rangle \text{ such that } \tau(e) = t\}. \qquad \square$$

If $\tau(M)$ is a tree transformation computed by a macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$, we simply write $L_{out}(M)$ instead of $L_{out}(\tau(M))$ and we simply call $L_{out}(M)$ the *output language of* M instead of the output language of the tree transformation computed by M. If Δ is monadic, then the output language of M is also called *monadic*. Clearly, if $L_{out}(M)$ only contains trees which are built up from symbols of the subset $\Delta^{(0)} \cup \Delta^{(1)} \subseteq \Delta$, then there is a macro tree transducer M' with monadic ranked alphabet $\Delta^{(0)} \cup \Delta^{(1)}$ of output symbols and with $L_{out}(M') = L_{out}(M)$. Thus, if a monadic language L is an output language of a macro tree transducer M, we always assume that M has a monadic ranked alphabet Δ of output symbols. Hence, L is a monadic output language.

We denote the *classes of output languages* of (producing and nondeleting) macro tree transducers, (c)–tree transducers, and $(c_1 * r_1, \ldots, c_k * r_k)$–tree transducers by MT_{out}, $MT_{(c),out}$, and $MT_{(c_1*r_1,\ldots,c_k*r_k),out}$, respectively.

If we want to prove that a certain tree transformation τ is not an element of the class MT, then the output language $L_{out}(\tau)$ can be very useful. It would suffice to show for example with the help of a pumping lemma that $L_{out}(\tau) \notin MT_{out}$. Thus, since $L_{out}(\tau)$ is not the range of a macro tree transducer, τ cannot be the tree transformation computed by a macro tree transducer. This principle carries over to all classes $MT_{(c)}$ and $MT_{(c_1*r_1,\ldots,c_k*r_k)}$.

For the sake of convenience, we now omit the parantheses for arguments of monadic output symbols in the rest of the paper; the parantheses for arguments of functions remain.

Example 3.17 Let M_1 be the macro tree transducer defined in Example 3.2 and let d be the derivation of Example 3.7.

In the following we would write rule (1) of M_1 in the form

$$f(\sigma(x_1, x_2), y_1) \rightarrow T(f(x_1, Ly_1), f(x_2, Ry_1), g(x_1, g(x_2, E))).$$

In analogy we would write the last but one sentential form of d that was shown in Example 3.7 as

$$T(T(BLLE, BRLE, AAE), BRE, g(11, Ag(12, g(2, E)))). \qquad \square$$

We want to conclude this section with a result concerning monadic output languages: If a nondeleting macro tree transducer M has a monadic output language $L_{out}(M)$, then there is also a nondeleting macro tree transducer M' with $L_{out}(M') = L_{out}(M)$ which has only functions with rank 1 or rank 2, because the functions of M with higher rank cannot occur in derivations. This is due to the fact that M is nondeleting. Thus in a proof that a monadic language can be no output language of a (producing and nondeleting) macro tree transducer it is sufficient to concern only macro tree transducers with functions of rank 1 or rank 2.

Lemma 3.18 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a nondeleting (and producing) macro tree transducer and $\Delta = \Delta^{(0)} \cup \Delta^{(1)}$. Then,

$$L_{out}(M) \in \bigcup_{i,j \geq 0} MT_{(i*1, j*2), out}.$$

Proof. We prove that every function f with $rank(f) \geq 3$ is irrelevant, showing for every input tree $e \in T\langle\Sigma\rangle$ and for every derivation $d = (\tilde{\rho}[x_1/\varepsilon] \Rightarrow^+_{M,e} nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon]))$ that it cannot contain a sentential form in which a function $f \in \bigcup_{n\geq 3} F^{(n)}$ occurs.

Let $e \in T\langle\Sigma\rangle$ and d be an arbitrary derivation $(\tilde{\rho}[x_1/\varepsilon] \Rightarrow^+_{M,e} nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon]))$. We prove by induction on the structure of e that there are no function occurrences (f, p) with $f \in \bigcup_{n\geq 3} F^{(n)}$ and $p \in paths(e)$ in sentential forms of d.

Actually, the induction is performed on paths through e, where the induction base contains the above statement for every path p which leads to a leaf of e. In the induction step the above statement has to be proved for every path p leading to a subtree e' of e which is no leaf. In this case the induction hypothesis can be used for paths pi with $i \in \mathbb{N}_+$ leading to the direct subtrees of e':

<u>Case 1:</u> $p \in paths(e)$ with $subtree(e, p) = \sigma \in \Sigma^{(0)}$:

Assume that there is $n \geq 2$ and $f \in F^{(n+1)}$, such that there is a function occurrence (f, p) in a sentential form sf of d. Since d computes the normal form $nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon]) \in T\langle\Delta\rangle$ and $label(e, p) = subtree(e, p) = \sigma$, in at least one derivation step the rule $f(\sigma, y_1, \ldots, y_n) \to rhs(f, \sigma)$ has to be applied. Because $rhs(f, \sigma) \in RHS(\emptyset, \Delta, \emptyset, Y_n) = T\langle\Delta\rangle(Y_n)$ and Δ is monadic, there is at most one y_i with $i \in [n]$ in $rhs(f, \sigma)$. Thus M is deleting in contradiction to our assumption and the induction base is proved.

<u>Case 2:</u> $p \in paths(e)$ with $subtree(e, p) = \sigma(e_1, \ldots, e_k)$, $\sigma \in \Sigma^{(k)}$, and $e_1, \ldots, e_k \in T\langle\Sigma\rangle$:

Assume that there is $n \geq 2$ and $f \in F^{(n+1)}$, such that there is a function occurrence (f, p) in a sentential form sf of d. Since d computes the normal form $nf(\Rightarrow_{M,e}, \tilde{\rho}[x_1/\varepsilon]) \in T\langle\Delta\rangle$ and $label(e, p) = \sigma$, in at least one derivation step the rule $f(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_n) \to rhs(f, \sigma)$ has to be applied. There are two cases:

- $rhs(f, \sigma) \in RHS(F^{(1)} \cup F^{(2)}, \Delta, X_k, Y_n)$:

  Since Δ is monadic and the first argument of every function has to be an element of X_k, there is at most one y_i with $i \in [n]$ in $rhs(f, \sigma)$. Thus M is deleting in contradiction to our assumption.

- $rhs(f, \sigma) \in RHS(F, \Delta, X_k, Y_n) - RHS(F^{(1)} \cup F^{(2)}, \Delta, X_k, Y_n)$:

  Thus, there is $m \geq 2$, $f' \in F^{(m+1)}$, and $i \in [k]$, such that there is a function occurrence (f', x_i) in $rhs(f, \sigma)$. Then the rule application delivers a function occurrence (f', pi) in the resulting sentential form. This is a contradiction to the induction hypothesis, because $subtree(e, pi) = e_i$ is a direct subtree of $\sigma(e_1, \ldots, e_k)$.

Thus the induction step is proved. □

From the proof of Lemma 3.18 we further gain the following observation:

Observation 3.19 If $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ is a nondeleting (and producing) macro tree transducer with monadic output language $L_{out}(M)$ which has only functions with rank 1 or rank 2, then M additionally is noncopying. This is due to the fact that (i) for every $f \in F^{(1)}$, $k \geq 0$, and $\sigma \in \Sigma^{(k)}$ the right–hand side $rhs(f, \sigma) \in RHS(F, \Delta, X_k, \emptyset)$, and (ii) for every $f \in F^{(2)}$, $k \geq 0$, and $\sigma \in \Sigma^{(k)}$ the right–hand side $rhs(f, \sigma) \in RHS(F^{(1)} \cup F^{(2)}, \Delta, X_k, \{y_1\})$. Thus, since Δ is monadic and the first argument of every function has to be an element of X_k, there is at most one occurrence of y_1 in $rhs(f, \sigma)$.

Hence, because of Lemma 3.18, the noncopying–condition is no additional restriction, if we consider monadic output languages of nondeleting (and producing) macro tree transducers. □

4 Pumping Lemma for Macro Tree Transducers

This section first presents the central ideas of the pumping lemma for output languages of macro tree transducers. Further we introduce the pumping index of a macro tree transducer. If the size of an output tree is at least the pumping index, then it can be pumped. Finally, the pumping lemma itself together with its proof is presented.

4.1 Motivation

We illustrate the following considerations by Figure 4. Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer. If we choose a sufficiently large output tree $t \in L_{out}(M)$, then every input tree e with $\tau(M)(e) = t$ is high enough, such that it has a path p from the root of e to a leaf node, on which two different nodes x_1 and x_2 can be found such that

- there exist strings p_1, p_2, and p_3 such that $|p_2| > 0$ and $p = p_1p_2p_3$,
- $x_1 = node(e, p_1)$ and $x_2 = node(e, p_1p_2)$, and
- the function–sets $funset(e, p_1)$ and $funset(e, p_1p_2)$ are equal.

In Figure 4 an input tree e, the path $p = p_1p_2p_3$, and the nodes x_1 and x_2 are indicated. Note that we deviate a little bit from the previous representation of trees by drawing horizontal lines through the root and the nodes x_1 and x_2 of e to add further information: We assume here that $funset(e, p_1) = funset(e, p_1p_2) = \{f_1, f_2\} \subseteq F$. The corresponding function occurrences (f_1, p_1) and (f_2, p_1) (and (f_1, p_1p_2) and (f_2, p_1p_2)) are represented by nodes on the line through x_1 (and x_2, respectively), which are simply labeled by the function symbols f_1 and f_2. The initial sentential form $\tilde{\rho}[x_1/\varepsilon]$ is represented by a node on the line through the root of e, which is simply labeled by $\tilde{\rho}$.

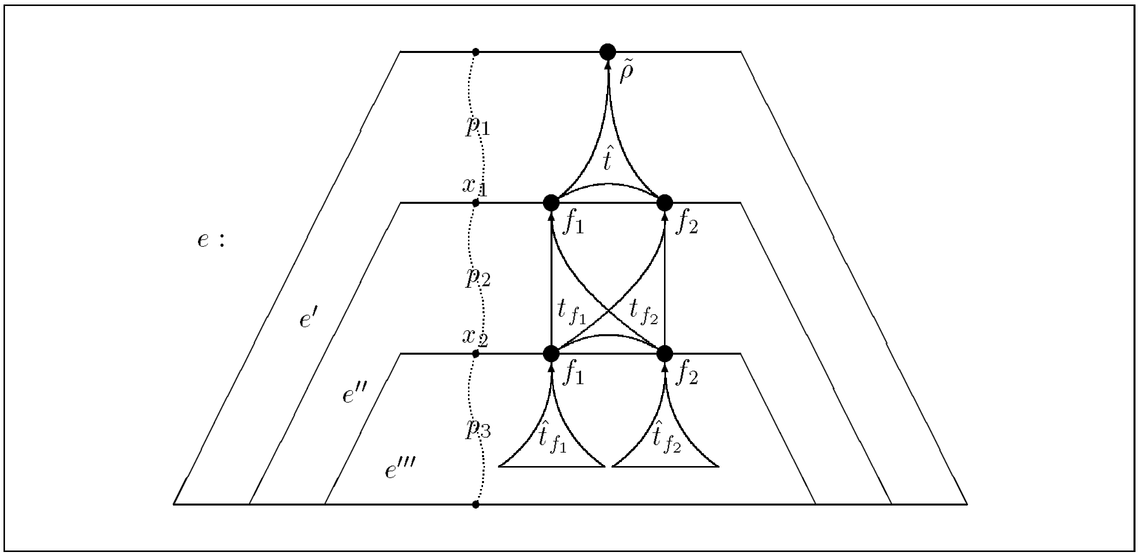

Figure 4: Input tree e with input patterns and induced output patterns.

The nodes x_1 and x_2 define a decomposition of e into three *input patterns* e', e'', and e'''. Intuitively,

- e' is the tree e without the subtree which has x_1 as root.
- e'' is the tree which has x_1 as root without the subtree which has x_2 as root.
- e''' is the tree which has x_2 as root.

This decomposition of e induces a decomposition of the output tree t into a certain *output pattern* $\hat{t}$ and certain *output patterns* t_f and $\hat{t}_f$ for every function $f \in funset(e, p_1)$ ($= funset(e, p_1p_2)$):

- The tree $\hat{t}$ corresponds to the normal form of $\tilde{\rho}[x_1/\varepsilon]$ that is calculated only on the nodes of e'.
- For every function $f \in funset(e, p_1)$ with $rank(f) = n + 1 \geq 1$, the tree t_f (and $\hat{t}_f$) corresponds to the normal form of $f(p_1, y_1, \ldots, y_n)$ (and $f(p_1p_2, y_1, \ldots, y_n)$, respectively) that is calculated only on the nodes of e'' (and e''', respectively). Here the variables $y_1, \ldots, y_n$ are viewed as additional output symbols with rank 0.

Because of the restrictions of the derivation relation, the pattern $\hat{t}$ (and, for every $f \in funset(e, p_1)$, the pattern t_f) can contain subtrees of the form $f'(p_1, t_1, \ldots, t_m)$ (and $f'(p_1p_2, t_1, \ldots, t_m)$, respectively), where $f' \in funset(e, p_1)$ has the rank $m + 1 \geq 1$. Since later we shall define the pumping process in such a way that it only needs the knowledge about the output patterns, the path–information p_1 (and p_1p_2, respectively) in such subtrees, which refers to the input tree e, is no more suitable. Thus we transform the pattern $\hat{t}$ (and, for every $f \in funset(e, p_1)$, the pattern t_f) by substituting subtrees of the form $f'(p_1, t_1, \ldots, t_m)$ and $f'(p_1p_2, t_1, \ldots, t_m)$ by $f'(t_1, \ldots, t_m)$, where f' is a new symbol with rank m. For the sake of understanding we choose exactly the functions as names for the new symbols. Based on the rank, the reader can retrieve whether symbols or functions are concerned at a time. We only mention already here that the lost path–information is compensated in the definition of the pumping process.

In Figure 4 these output patterns are indicated. The root of every output pattern is represented by an arrow. Besides output symbols, also the new symbols f_1 and f_2 can occur in $\hat{t}$ and in t_f for every $f \in \{f_1, f_2\}$. Figure 4 represents this fact by integrating f_1 and f_2 at x_1 (and x_2) into $\hat{t}$ (and t_{f_1} and t_{f_2}, respectively). The figure suggests that these symbols appear only as leaf nodes of the patterns, but clearly, in general this is not true, because the underlying functions can have context parameters. However, our intention was to simplify the representation. Additionally, context parameters can occur in t_f and $\hat{t}_f$ for every $f \in \{f_1, f_2\}$. Also this fact is not expressed in the figure.

If we construct new input trees by repeating the input pattern e'' arbitrarily often, we can get new output trees by translating the new input trees. All of them are by definition elements of $L_{out}(M)$. Since $funset(e, p_1) = funset(e, p_1p_2)$, also the input patterns of the new input trees are compatible in the sense that there is the same function–set on, roughly speaking, every level of the input tree. Thus, since we are dealing with total deterministic macro tree transducers only, for every $f \in funset(e, p_1)$, the output pattern t_f must be used for every repetition of e'' to obtain the new output tree. Figure 5 shows the situation in which e'' is repeated twice.

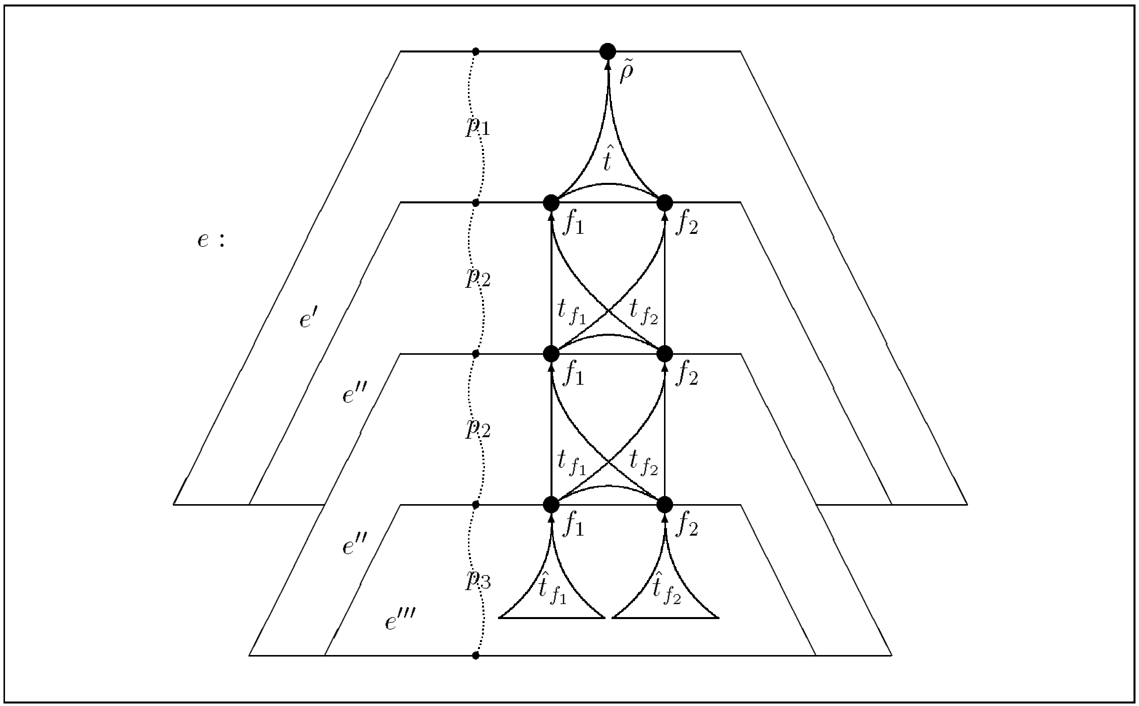

Figure 5: Input tree with two repetitions of e'' and output patterns.

Note that for the pumping process it is not necessary that the nodes x_1 and x_2 are labeled by the same symbol, in contrast to the pumping lemma for context–free languages (cf. for example [BPS61]). This is due to the fact that we only deal with ranked alphabets rather than heterogeneous signatures; thus only the rank of the symbols is important when building up trees.

For every $r \geq 0$, we denote the input tree, which has r repetitions of e'', by e_r and the normal form $nf(\Rightarrow_{e_r}, \tilde{\rho}[x_1/\varepsilon])$ by $\underline{tree}(r)$. For the calculation of an output tree $\underline{tree}(r)$ in the pumping lemma, we shall define a recursive function $\underline{tree}_1$ (and $\underline{tree}_2$), which performs the construction of $\underline{tree}(r)$ by working on a certain repetition of e'' (and working on e''', respectively) and by using the output patterns defined above as follows:

We start with the pattern $\hat{t}$ which represents the calculation on e'. Then we apply r times the function $\underline{tree}_1$ to the current intermediate result. Roughly speaking, every application of $\underline{tree}_1$ replaces every occurrence of a symbol f in a current intermediate result by the output pattern t_f which represents a calculation on a repetition of e'' and substitutes every occurrence of a variable y_i in t_f by the recursive application of $\underline{tree}_1$ on the subtree rooting at the i–th child of the occurrence of f. Every application of $\underline{tree}_1$ yields a new intermediate result. Finally, we apply the function $\underline{tree}_2$ to the last intermediate result. Roughly speaking, the application of $\underline{tree}_2$ replaces every occurrence of a symbol f in this intermediate result by the output pattern $\hat{t}_f$ which represents a calculation on e''' and substitutes every occurrence of a variable y_i in $\hat{t}_f$ by the recursive application of $\underline{tree}_2$ on the subtree rooting at the i–th child of the occurrence of f. This application of $\underline{tree}_2$ yields the output tree $\underline{tree}(r)$, because there is no new symbol f' in a pattern $\hat{t}_f$ which has to be replaced.

We further explain these informal considerations by the following example:

Example 4.1 Let M_1 be the macro tree transducer defined in Example 3.2. For simplicity we repeat the initial right–hand side $\tilde{\rho}$ and the set R of rules of M_1:

$$\begin{array}{rcll}
\tilde{\rho} & = & f(x_1, E), \text{ and} & \\
R & = & \{f(\sigma(x_1,x_2),y_1) & \rightarrow \; T(f(x_1,Ly_1), f(x_2,Ry_1), g(x_1,g(x_2,E))), \\
 & & f(\alpha,y_1) & \rightarrow \; By_1, \\
 & & g(\sigma(x_1,x_2),y_1) & \rightarrow \; g(x_1, Ag(x_2,y_1)), \\
 & & g(\alpha,y_1) & \rightarrow \; Ay_1 \qquad\qquad\qquad\qquad\qquad\qquad \}
\end{array}$$

Although the pumping lemma only guarantees to work for an output tree t with $size(t) \geq pi_{M_1}$ for a certain natural number pi_{M_1} (which is called the pumping index of M_1), it often also works for smaller output trees. Here again we take the output tree $t = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon])$ which was shown in Figure 3, where $e = \sigma(\sigma(\alpha,\alpha),\alpha)$ is the input tree. Indeed, as the reader can check later, the output tree t is smaller than pi_{M_1}.

We choose the path $p = 11$ in e with its subpaths $p_1 = 1$, $p_2 = 1$, and $p_3 = \varepsilon$. Thus we consider the two nodes $node(e,1)$ and $node(e,11)$ which can be reached from the root of e by paths $p_1 = 1$ and $p_1p_2 = 11$, respectively. Note that $\sigma = label(e,1)$ and $\alpha = label(e,11)$, and

$$\begin{array}{cl}
 & funset(e,1) \\
= & set(seq_1(nf(\Rightarrow_{e,P(e,\varepsilon-1)}, f(\varepsilon,E)))) \\
= & set(seq_1(T(f(1,LE), BRE, g(1,AE)))) \\
= & \{f,g\} \\
= & set(seq_{11}(T(T(f(11,LLE), BRLE, g(11,AE)), BRE, g(11,AAAE)))) \\
= & set(seq_{11}(nf(\Rightarrow_{e,P(e,\varepsilon-11)}, f(\varepsilon,E)))) \\
= & funset(e,11).
\end{array}$$

In Figure 6 we show the input tree e and the patterns e', e'', and e''' with the nodes reached by the paths ε, 1, and 11, respectively, as roots. The pruned subtrees are substituted by a new symbol w. For later purposes, we frame those parts of the patterns which only consist of input symbols. In fact, we have $e = e'[w/e''[w/e''']]$.

In analogy to the preceding explanations we obtain the patterns in the output tree t as follows: we calculate

- the tree $\hat{t}$ as normal form of the initial sentential form $\tilde{\rho}[x_1/\varepsilon] = f(\varepsilon, E)$, where the derivation relation is restricted to e' and thus to the paths ε and 2 of e,
- the tree t_f (and t_g) as normal form of $f(1,y_1)$ (and $g(1,y_1)$, respectively), where the derivation relation is restricted to e'' and thus to the paths 1 and 12 of e,

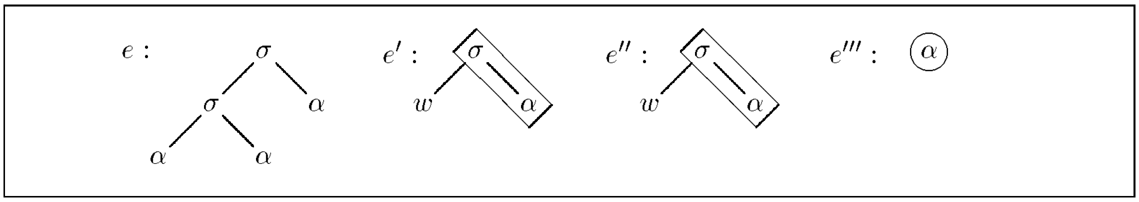

Figure 6: Input tree e and its decomposition.

- the tree $\hat{t}_f$ (and $\hat{t}_g$) as normal form of $f(11, y_1)$ (and $g(11, y_1)$, respectively), where the derivation relation is restricted to e''' and thus to the path 11 of e,

and replace subtrees of the form $f(1, t')$ and $f(11, t')$ (and $g(1, t')$ and $g(11, t')$) by $f(t')$ (and $g(t')$), where f (and g, respectively) is a new symbol with rank 1. Thus, the output patterns are computed as follows:

$$\begin{aligned}
\hat{t} &= nf(\Rightarrow_{e,\{\varepsilon,2\}}, f(\varepsilon, E))[f(1, LE)/fLE,\ g(1, AE)/gAE] \\
&= T(f(1, LE), BRE, g(1, AE))[f(1, LE)/fLE,\ g(1, AE)/gAE] \\
&= T(fLE, BRE, gAE),
\end{aligned}$$

$$\begin{aligned}
t_f &= nf(\Rightarrow_{e,\{1,12\}}, f(1, y_1))[f(11, Ly_1)/fLy_1,\ g(11, AE)/gAE] \\
&= T(f(11, Ly_1), BRy_1, g(11, AE))[f(11, Ly_1)/fLy_1,\ g(11, AE)/gAE] \\
&= T(fLy_1, BRy_1, gAE),
\end{aligned}$$

$$\begin{aligned}
t_g &= nf(\Rightarrow_{e,\{1,12\}}, g(1, y_1))[g(11, AAy_1)/gAAy_1] \\
&= g(11, AAy_1)[g(11, AAy_1)/gAAy_1] \\
&= gAAy_1,
\end{aligned}$$

$$\begin{aligned}
\hat{t}_f &= nf(\Rightarrow_{e,\{11\}}, f(11, y_1)) \\
&= By_1,
\end{aligned}$$

$$\begin{aligned}
\hat{t}_g &= nf(\Rightarrow_{e,\{11\}}, g(11, y_1)) \\
&= Ay_1.
\end{aligned}$$

Since in this example the function symbols do not occur nested in the above normal forms, we are able to present the substitutions in this easy form. In general, the function symbols occur nested and we have to define a recursive transformation function which realizes the substitutions.

In Figure 7 we show the output patterns defined above framing those parts which only consist of output symbols.

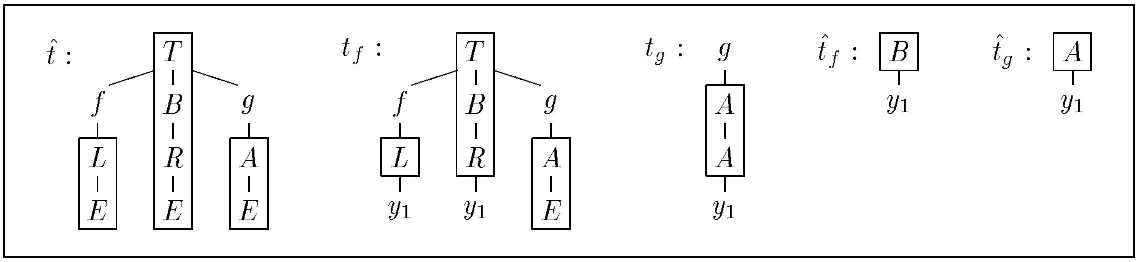

Figure 7: Output patterns of t.

Now we show the pumping process in the cases

- $r = 0$, thus the pattern e'' is dropped, yielding the input tree $e_0 = e'[w/e''']$, and
- $r = 1$, thus the pattern e'' occurs once, yielding the input tree $e_1 = e'[w/e''[w/e''']]$ which is e again.

We do not show here the pumping process for a case $r \geq 2$, because the calculations become to big and because the whole mechanism can already be studied with the two chosen cases.

As it was outlined informally before presenting this example, we can calculate the corresponding output trees $\underline{tree}(0)$ and $t = \underline{tree}(1)$ with the help of the functions $\underline{tree}_1$ and $\underline{tree}_2$ as follows. Recall that ${\underline{tree}_1}^i$ denotes the i–fold application of $\underline{tree}_1$.

$$
\begin{aligned}
& \underline{tree}(0) \\
= \; & \underline{tree}_2({\underline{tree}_1}^0(T(fLE, BRE, gAE))) \\
= \; & \underline{tree}_2(T(fLE, BRE, gAE)) \\
= \; & T(\underline{tree}_2(fLE), BRE, \underline{tree}_2(gAE)) \\
= \; & T(By_1[y_1/\underline{tree}_2(LE)], BRE, Ay_1[y_1/\underline{tree}_2(AE)]) \\
= \; & T(BLE, BRE, AAE)
\end{aligned}
$$

$$
\begin{aligned}
& \underline{tree}(1) \\
= \; & \underline{tree}_2({\underline{tree}_1}^1(T(fLE, BRE, gAE))) \\
= \; & \underline{tree}_2(T(\underline{tree}_1(fLE), BRE, \underline{tree}_1(gAE))) \\
= \; & \underline{tree}_2(T(T(fLy_1, BRy_1, gAE)[y_1/\underline{tree}_1(LE)], BRE, gAAy_1[y_1/\underline{tree}_1(AE)])) \\
= \; & \underline{tree}_2(T(T(fLLE, BRLE, gAE), BRE, gAAAE)) \\
= \; & T(T(\underline{tree}_2(fLLE), BRLE, \underline{tree}_2(gAE)), BRE, \underline{tree}_2(gAAAE)) \\
= \; & T(T(By_1[y_1/\underline{tree}_2(LLE)], BRLE, Ay_1[y_1/\underline{tree}_2(AE)]), BRE, Ay_1[y_1/\underline{tree}_2(AAAE)]) \\
= \; & T(T(BLLE, BRLE, AAE), BRE, AAAAE)
\end{aligned}
$$

In Figure 8 we show the output trees $\underline{tree}(0)$, $\underline{tree}(1)$, and their decompositions into (the different parts of) the output patterns. Every part of an output pattern is labeled with the input pattern which causes it. We also show the input trees, corresponding to the output trees, and their decompositions into input patterns. □

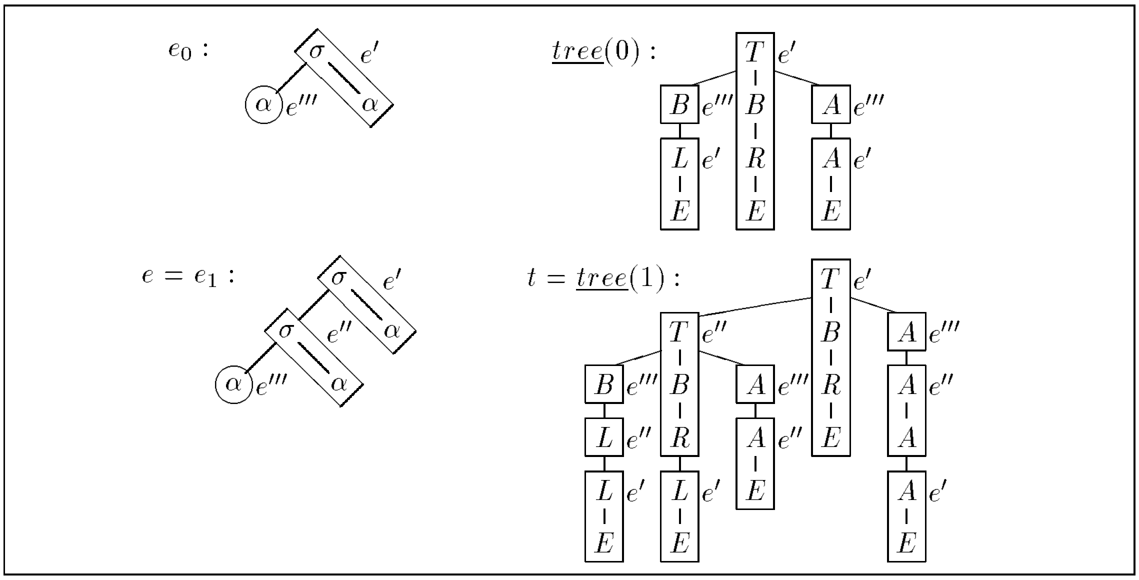

Figure 8: Decompositions of e_0, e_1, $\underline{tree}(0)$, and $\underline{tree}(1)$.

4.2 Pumping Index

In Subsection 4.1 we have seen that for a macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ and an input tree $e \in T\langle\Sigma\rangle$ we need a certain height of e, such that we have a repetition of a function–set on a path. Since there are $2^{card(F)} - 1$ possibilities for nonempty subsets of F, we need at least the height $2^{card(F)}$ of e. This height can be guaranteed, if the size of the output tree $t = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon])$ is large enough. In the following we define for M the pumping index $pi_M \in I\!N$ and prove that if the size of t is at least pi_M, then the height condition of e is satisfied. For later purposes we define pi_M in such a way that we can prove $height(e) \geq 2^{card(F)} + 1$.

Definition 4.2 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer.

The *pumping index* pi_M *of* M is defined by

$$pi_M = 1 + \sum_{i=0}^{pi'_M - 2} (maxrank_{M,\Delta})^i, \qquad \text{where}$$

$$pi'_M = 1 + (maxheight_{M,R})^{2^{card(F)}+1}.$$

□

Lemma 4.3 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer. Let $t \in L_{out}(M)$.

If $size(t) \geq pi_M$, then for every $e \in T\langle\Sigma\rangle$ such that $t = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon])$, the height $height(e) \geq 2^{card(F)} + 1$.

Proof. Consider $t \in L_{out}(M)$ with $size(t) \geq pi_M$. We examine an arbitrary input tree $e \in T\langle\Sigma\rangle$ with $\tilde{\rho}[x_1/\varepsilon] \Rightarrow_e^+ t$. The proof consists of two implications:

1. If $size(t) \geq pi_M = 1 + \sum_{i=0}^{pi'_M - 2}(maxrank_{M,\Delta})^i$, then $height(t) \geq pi'_M$.

   Explanation: Since $maxrank_{M,\Delta}$ is the maximum rank of output symbols of M, there are at most $(maxrank_{M,\Delta})^i$ output symbols in an output tree t which can be reached by a path $p \in paths(t)$ with $|p| = i$. Thus, for an output tree t with $height(t) \leq pi'_M - 1$, the size $size(t) \leq \sum_{i=0}^{pi'_M - 2}(maxrank_{M,\Delta})^i = pi_M - 1$.

2. If $height(t) \geq pi'_M = 1 + (maxheight_{M,R})^{2^{card(F)}+1}$, then $height(e) \geq 2^{card(F)} + 1$.

   Explanation: Since $maxheight_{M,R}$ is the maximum height of right–hand sides of rules of M, by Lemma 3.9 for an input tree e with $height(e) \leq 2^{card(F)}$ the corresponding output tree t has the height $height(t) \leq (maxheight_{M,R})^{2^{card(F)}+1} = pi'_M - 1$. □

We only mention here that for a macro tree transducer M the constant pi'_M can also serve as a pumping index in another pumping lemma which works with heights of output trees instead of sizes. Though at first sight such a pumping lemma seems to be a little bit easier, it turned out that the exact calculation of heights in applications is more difficult. However, in our applications in Section 6 there is no difference between sizes and heights, because there we restrict ourselves to monadic output languages.

4.3 Pumping Lemma and its Proof

Theorem 4.4 (Pumping Lemma)
Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a (nondeleting and producing) macro tree transducer with pumping index pi_M.

For every $t \in L_{out}(M)$ with $size(t) \geq pi_M$

- there exists a ranked alphabet $(U, rank_U)$ with $U \subseteq F$, $card(U) \geq 1$, and $rank_U(f) = rank_F(f) - 1$ for every $f \in U$,
- there exists a tree $\hat{t} \in T\langle \Delta \cup U \rangle$ with $size_\Delta(\hat{t}) \geq 1$,
- for every $f \in U$, there exists a tree $t_f \in T\langle \Delta \cup U \rangle(Y_{rank_U(f)})$ with $1 \leq size_\Delta(t_f) < pi_M$ and $size_U(t_f) < pi_M$,
- for every $f \in U$, there exists a tree $\hat{t}_f \in T\langle \Delta \rangle(Y_{rank_U(f)})$ with $1 \leq size_\Delta(\hat{t}_f) < pi_M$,
- for every $f \in U$, the symbol f occurs in $\hat{t}$,
- for every $f \in U$, there is an $f' \in U$ such that f occurs in $t_{f'}$,
- for every $f \in U$ and for every $y \in Y_{rank_U(f)}$, the symbol y occurs in t_f and $\hat{t}_f$,

such that

1. $t = \underline{tree}(1)$ and

2. for every $r \geq 0$, the tree $\underline{tree}(r) \in L_{out}(M)$,

where the function

$\underline{tree} : I\!N \longrightarrow T\langle\Delta\rangle$ is defined by:
For every $r \in I\!N$, $\underline{tree}(r) \;=\; \underline{tree}_2(\underline{tree}_1{}^r(\hat{t}))$.

with

$\underline{tree}_1 : T\langle\Delta \cup U\rangle \longrightarrow T\langle\Delta \cup U\rangle$ is defined by:
For every $n \in I\!N, \delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in T\langle\Delta \cup U\rangle$,
$\underline{tree}_1(\delta(t_1, \ldots, t_n)) \;=\; \delta(\underline{tree}_1(t_1), \ldots, \underline{tree}_1(t_n))$.
For every $n \in I\!N, f \in U^{(n)}$, and $t_1, \ldots, t_n \in T\langle\Delta \cup U\rangle$,
$\underline{tree}_1(f(t_1, \ldots, t_n)) \;=\; t_f[y_j/\underline{tree}_1(t_j)\ ;\ j \in [n]]$.

and

$\underline{tree}_2 : T\langle\Delta \cup U\rangle \longrightarrow T\langle\Delta\rangle$ is defined by:
For every $n \in I\!N, \delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in T\langle\Delta \cup U\rangle$,
$\underline{tree}_2(\delta(t_1, \ldots, t_n)) \;=\; \delta(\underline{tree}_2(t_1), \ldots, \underline{tree}_2(t_n))$.
For every $n \in I\!N, f \in U^{(n)}$, and $t_1, \ldots, t_n \in T\langle\Delta \cup U\rangle$,
$\underline{tree}_2(f(t_1, \ldots, t_n)) \;=\; \hat{t}_f[y_j/\underline{tree}_2(t_j)\ ;\ j \in [n]]$.

Proof. Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer with pumping index pi_M.

Consider $t \in L_{out}(M)$ with $size(t) \geq pi_M$. By Lemma 4.3 we know that, for every input tree e with $\tilde{\rho}[x_1/\varepsilon] \Rightarrow_e^+ t$, the condition $height(e) \geq 2^{card(F)} + 1$ holds.

We choose an arbitrary such input tree e and a path $p \in paths(e)$ from the root of e to a leaf of e with maximum length; then $|p| \geq 2^{card(F)}$.

Since there are exactly $2^{card(F)} - 1$ possibilities to choose an arbitrary, nonempty subset of F and since $|p| \geq 2^{card(F)}$, we can choose strings p_1, p_2, and p_3, with

- $p = p_1 p_2 p_3$,
- $|p_2| \geq 1$,
- $|p_2 p_3| \leq 2^{card(F)} - 1$, and
- $funset(e, p_1) = funset(e, p_1 p_2)$.

The third condition means that we consider a repetition in the function–sets of only those paths which lead to the first $2^{card(F)}$ nodes on p, starting from the leaf at p.

Note that it would have been sufficient to have $height(e) \geq 2^{card(F)}$ and thus $|p| \geq 2^{card(F)} - 1$, but later in the proof of the size conditions for the output patterns we again use the pumping index pi_M to avoid the definition of a new constant. Otherwise we would have had another formulation of the pumping lemma with two constants (like in [BPS61], there the constants are called p and q).

We define the ranked alphabet $(U, rank_U)$ as follows: $U = funset(e, p_1) \subseteq F$ and $rank_U(f) = rank_F(f) - 1$ for every $f \in U$. In fact, $card(U) \geq 1$, because M is non-deleting and thus every symbol of the input tree must be visited by a function (cf. also Lemma 3.15).

Let $w \notin \Sigma$ with $rank(w) = 0$. We define trees $e' \in T\langle\Sigma \cup \{w\}\rangle$ and $e'' \in T\langle\Sigma \cup \{w\}\rangle$, where both, e' and e'', have exactly one occurrence of w, and $e''' \in T\langle\Sigma\rangle$ with the help of p_1, p_2, and p_3 as follows:

$$\begin{array}{lcl} e' & = & e[p_1 \leftarrow w], \\ e'' & = & subtree(e[p_1p_2 \leftarrow w], p_1), \text{ and} \\ e''' & = & subtree(e, p_1p_2). \end{array}$$

Then the representation $e = e'[w/e''[w/e''']]$ holds. The reader can find these patterns of e and the paths leading to them in Figure 4.

In the sequel we need the subsets $P(e, \varepsilon - p_1)$, $P(e, p_1 - p_1p_2)$, and $P(e, p_1p_2)$ of $paths(e)$, which, roughly speaking, lead from the root of e to the nodes in the three parts e', e'', and e''', respectively. Recall that the path p_1 leading to the root of e'' is excluded from $P(e, \varepsilon - p_1)$ and that the path p_1p_2 leading to the root of e''' is excluded from $P(e, p_1 - p_1p_2)$. To calculate the desired output patterns that are needed for the pumping process, we restrict the derivation relation of M to the sets $P(e, \varepsilon - p_1)$, $P(e, p_1 - p_1p_2)$, and $P(e, p_1p_2)$, respectively (cf. Definition 3.6).

For the definition of the output patterns we define a function $\underline{trans}$ that transforms sentential forms containing variables of Y_m with $m = max(\{rank_F(f) - 1 \mid f \in F\})$, which are viewed as additional output symbols with rank 0, into trees which are built up from output symbols, symbols of the set U, and variables of Y_m. For later purposes we define $\underline{trans}$ on the bigger domain $SF(F, \{p_1{p_2}^l \mid l \geq 0\}, \Delta \cup Y_m)$ than the domain $SF(F, \{p_1, p_1p_2\}, \Delta \cup Y_m)$, which would be needed in this moment:

$\underline{trans} : SF(F, \{p_1{p_2}^l \mid l \geq 0\}, \Delta \cup Y_m) \longrightarrow T\langle\Delta \cup U\rangle(Y_m)$ is defined by:
For every $n \in I\!N, \delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in SF(F, \{p_1{p_2}^l \mid l \geq 0\}, \Delta \cup Y_m)$,
$$\underline{trans}(\delta(t_1, \ldots, t_n)) = \delta(\underline{trans}(t_1), \ldots, \underline{trans}(t_n)).$$
For every $y \in Y_m$,
$$\underline{trans}(y) = y.$$
For every $n \in I\!N, f \in F^{(n+1)}, l \in I\!N$, and $t_1, \ldots, t_n \in SF(F, \{p_1{p_2}^l \mid l \geq 0\}, \Delta \cup Y_m)$,
$$\underline{trans}(f(p_1{p_2}^l, t_1, \ldots, t_n)) = f(\underline{trans}(t_1), \ldots, \underline{trans}(t_n)).$$

With the help of the function $\underline{trans}$ we can define the desired output patterns (cf. Figure 4) as follows:

$$\begin{array}{lcl} \hat{t} & = & \underline{trans}(nf(\Rightarrow_{e,P(e,\varepsilon - p_1)}, \tilde{\rho}[x_1/\varepsilon])). \\ \multicolumn{3}{l}{\text{For every } n \in I\!N \text{ and } f \in U^{(n)},} \\ t_f & = & \underline{trans}(nf(\Rightarrow_{e,P(e,p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n))) \quad \text{and} \\ \hat{t}_f & = & nf(\Rightarrow_{e,P(e,p_1 p_2)}, f(p_1 p_2, y_1, \ldots, y_n)). \end{array}$$

By this definition every pattern has the type, which is required by the pumping lemma.

Now we prove the size conditions for the patterns:

1. $size_\Delta(\hat{t}) \geq 1$:

   We have $size_\Delta(\hat{t}) \geq 1$, because $|p_2 p_3| \leq 2^{card(F)} - 1$ and $|p| \geq 2^{card(F)}$, and thus $|p_2 p_3| < |p|$ and $p_1 \neq \varepsilon$. Therefore there must be at least one rule application to calculate $nf(\Rightarrow_{e,P(e,\varepsilon - p_1)}, \tilde{\rho}[x_1/\varepsilon]) \neq \tilde{\rho}[x_1/\varepsilon]$. Note that M is producing and that the function $\underline{trans}$ does not change the produced output symbols.

2. For every $n \in I\!N$, $f \in U^{(n)}$, $1 \leq size_\Delta(t_f) < pi_M$:

   We have $size_\Delta(t_f) \geq 1$, because $p_2 \neq \varepsilon$ and thus there must be at least one rule application to calculate $nf(\Rightarrow_{e,P(e,p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n)) \neq f(p_1, y_1, \ldots, y_n)$. Note again that M is producing and that the function $\underline{trans}$ does not change the produced output symbols. We have $size_\Delta(t_f) < pi_M$, because the calculation of $nf(\Rightarrow_{e,P(e,p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n))$ only takes place on the part e'' of the input tree:

   Since p is a longest path in e, its subpath $p_2 p_3$ with $|p_2 p_3| \leq 2^{card(F)} - 1$ is a longest path of $e''[w/e''']$. Then $height(e''[w/e''']) = height(subtree(e, p_1)) \leq 2^{card(F)}$ and because of Observation 3.14 we have $height(nf(\Rightarrow_{e,P(e,p_1)},\ f(p_1, y_1, \ldots, y_n))) \leq (maxheight_{M,R})^{2^{card(F)}+1} < pi'_M$ and thus the intermediate result

   $$size(nf(\Rightarrow_{e,P(e,p_1)}, f(p_1, y_1, \ldots, y_n))) \leq \sum_{i=0}^{pi'_M - 2} (maxrank_{M,\Delta})^i < pi_M$$

   Since the function $\underline{trans}$ leaves output symbols unchanged and since already generated output symbols cannot be deleted during the further computation on the tree e''', we can deduce

   $$\begin{array}{cl} & size_\Delta(t_f) \\ = & size_\Delta(\underline{trans}(nf(\Rightarrow_{e,P(e,p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n)))) \\ = & size_\Delta(nf(\Rightarrow_{e,P(e,p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n))) \\ \leq & size_\Delta(nf(\Rightarrow_{e,P(e,p_1)}, f(p_1, y_1, \ldots, y_n))) \\ \leq & size(nf(\Rightarrow_{e,P(e,p_1)}, f(p_1, y_1, \ldots, y_n))) \\ < & pi_M. \end{array}$$

3. For every $n \in I\!N$, $f \in U^{(n)}$, $size_U(t_f) < pi_M$:

   Assume that $size_U(t_f) \geq pi_M$. With the definitions of t_f and <u>*trans*</u> we can deduce $size_F(nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, f(p_1, y_1, \ldots, y_n))) \geq pi_M$. Thus, since M is nondeleting, at least pi_M applications of $(f, label(e, p_1p_2))$–rules with $f \in F$ are needed to calculate $nf(\Rightarrow_{e,P(e,p_1)}, f(p_1, y_1, \ldots, y_n))$. Because of the producing–condition, every rule application causes at least one output symbol, and because of the nondeleting condition, none of these symbols can be deleted. Then we have

$$size(nf(\Rightarrow_{e,P(e,p_1)}, f(p_1, y_1, \ldots, y_n))) \geq pi_M$$

   in contradiction to the intermediate result in 2.

4. For every $n \in I\!N$, $f \in U^{(n)}$, $1 \leq size_\Delta(\hat{t}_f) < pi_M$:

   We have $size_\Delta(\hat{t}_f) \geq 1$, because $nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))$ can only consist of output symbols and of $y_1, \ldots, y_n$ and thus cannot be equal to $f(p_1p_2, y_1, \ldots, y_n)$. Thus, there is at least one rule application to calculate the normal form. We have $size_\Delta(\hat{t}_f) < pi_M$, because the calculation of $nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))$ only takes place on the part e''' of the control tree. Since p is a longest path in e, its subpath p_3 with $|p_3| < |p_2p_3| \leq 2^{card(F)} - 1$ is a longest path of e'''. Now we can apply a similar argumentation as in 2. to deduce

$$\begin{aligned}
& size_\Delta(\hat{t}_f) \\
= \; & size_\Delta(nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))) \\
\leq \; & size(nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))) \\
< \; & pi_M.
\end{aligned}$$

In the next step we have to check, whether every symbol $f \in U$ occurs in $\hat{t}$ and in at least one $t_{f'}$, and whether, for every $f \in U$, every symbol $y \in Y_{rank_U(f)}$ occurs in t_f and $\hat{t}_f$:

Because of Definition 3.10 we have

$$\begin{aligned}
U &= funset(e, p_1) \\
&= set(seq_{p_1}(nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))
\end{aligned}$$

Thus $f \in U$ implies that there is a function occurrence (f, p_1) in $nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])$. Then by the definition of $\hat{t}$ and <u>*trans*</u>, the symbol f occurs in $\hat{t}$.

Because of Definition 3.10 we also have

$$\begin{aligned}
U &= funset(e, p_1p_2) \\
&= set(seq_{p_1p_2}(nf(\Rightarrow_{e,P(e,\varepsilon-p_1p_2)}, \tilde{\rho}[x_1/\varepsilon]))) \\
&= set(seq_{p_1p_2}(nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) \\
&= \bigcup_{f' \in U} set(seq_{p_1p_2}(nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, f'(p_1, y_1, \ldots, y_n))))
\end{aligned}$$

The last equation holds, because $U = funset(e, p_1)$ and because, roughly speaking, a function occurrence (f, p_1p_2) can only be introduced starting a derivation with a function occurrence (f', p_1) with $f' \in funset(e, p_1)$. Thus $f \in U$ implies that there is at least one $f' \in U$

such that there is a function occurrence (f, p_1p_2) in $nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, f'(p_1, y_1, \ldots, y_n))$. Then by the definition of $t_{f'}$ and $\underline{trans}$, the symbol f occurs in $t_{f'}$.

For every $f \in U$ and for every $y \in Y_{rank_U(f)}$, the symbol y occurs at least once in t_f and $\hat{t}_f$, because M is nondeleting. Thus in the definition of t_f and $\hat{t}_f$ by normal forms a symbol y cannot be deleted and the function $\underline{trans}$ leaves it unchanged.

The next item of this proof is to show that $t = \underline{tree}(1)$ and that for every $r \geq 0$, the property $\underline{tree}(r) \in L_{out}(M)$ holds.

We abbreviate the input tree, which is built up by repeating $r \geq 0$ times the input pattern e'', by e_r:

$$e_r = e'[w/\underbrace{e''[w/\ldots e''[w/}_{r\text{ times}}e'''\underbrace{]\ldots]}_{r\text{ times}}]$$

Thus, in particular, $e = e_1$.

First we have to verify the following Statements (1a) and (1b) concerning the functions $\underline{tree}_1$ and $\underline{tree}_2$, respectively:

(1a) For every $r \geq 0$, $1 \leq l \leq r$, and $t \in SF(F, \{p_1{p_2}^{l-1}\}, \Delta)$,
$\underline{tree}_1(\underline{trans}(t)) = \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, t))$.

(1b) For every $r \geq 0$ and $t \in SF(F, \{p_1{p_2}^{r}\}, \Delta)$,
$\underline{tree}_2(\underline{trans}(t)) = nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{r})}, t)$.

Roughly speaking, Statement (1a) expresses the following fact: The result of first calculating the normal form of a sentential form $t \in SF(F, \{p_1{p_2}^{l-1}\}, \Delta)$ on the l–th repetition of e'' in e_r and then transforming it with $\underline{trans}$ is equal to the result of first transforming t and then applying $\underline{tree}_1$, which uses for every $f \in U$ the output pattern t_f to simulate the calculation of the normal form. Statement (1b) is an analogous result relating the calculation on the occurrence of e''' in e_r with the function $\underline{tree}_2$. Here the transformation of the normal form can be omitted, because it only consists of output symbols.

Let $r \geq 0$ and $1 \leq l \leq r$. We prove Statement (1a) by structural induction on the set $SF(F, \{p_1{p_2}^{l-1}\}, \Delta)$. There are two cases:

<u>Case I:</u> For every $n \geq 0$, $\delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in SF(F, \{p_1{p_2}^{l-1}\}, \Delta)$, such that $t = \delta(t_1, \ldots, t_n)$,

$$\begin{aligned}
& \underline{tree}_1(\underline{trans}(\delta(t_1, \ldots, t_n))) \\
= \; & \underline{tree}_1(\delta(\underline{trans}(t_1), \ldots, \underline{trans}(t_n))) && \text{(Definition of } \underline{trans}\text{)} \\
= \; & \delta(\underline{tree}_1(\underline{trans}(t_1)), \ldots, \underline{tree}_1(\underline{trans}(t_n))) && \text{(Definition of } \underline{tree}_1\text{)} \\
= \; & \delta(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, t_1)), \ldots, \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, t_n))) \\
& && \text{(Induction Hypothesis)} \\
= \; & \underline{trans}(\delta(nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, t_1), \ldots, nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, t_n))) \\
& && \text{(Definition of } \underline{trans}\text{)} \\
= \; & \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1{p_2}^{l-1}-p_1{p_2}^{l})}, \delta(t_1, \ldots, t_n))).
\end{aligned}$$

Case II: For every $n \geq 0$, $f \in F^{(n+1)}$, and $t_1, \ldots, t_n \in SF(F, \{p_1 {p_2}^{l-1}\}, \Delta)$, such that $t = f(p_1 {p_2}^{l-1}, t_1, \ldots, t_n)$,

$$\begin{aligned}
& \underline{tree}_1(\underline{trans}(f(p_1 {p_2}^{l-1}, t_1, \ldots, t_n))) \\
= \; & \underline{tree}_1(f(\underline{trans}(t_1), \ldots, \underline{trans}(t_n))) \\
& \quad \text{(Definition of } \underline{trans}\text{)} \\
= \; & t_f[y_j / \underline{tree}_1(\underline{trans}(t_j)) \; ; \; j \in [n]] \\
& \quad \text{(Definition of } \underline{tree}_1\text{)} \\
= \; & \underline{trans}(nf(\Rightarrow_{e, P(e, p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n)))[y_j / \underline{tree}_1(\underline{trans}(t_j)) \; ; \; j \in [n]] \\
& \quad \text{(Definition of } t_f\text{)} \\
= \; & \underline{trans}(nf(\Rightarrow_{e, P(e, p_1 - p_1 p_2)}, f(p_1, y_1, \ldots, y_n))) \\
& \qquad [y_j / \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, t_j)) \; ; \; j \in [n]] \\
& \quad \text{(Induction Hypothesis)} \\
= \; & \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, f(p_1 {p_2}^{l-1}, y_1, \ldots, y_n))) \\
& \qquad [y_j / \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, t_j)) \; ; \; j \in [n]] \\
& \quad \text{(Same result on the occurrence of } e'' \text{ in } e \\
& \quad \;\text{and on the } l\text{–th repetition of } e'' \text{ in } e_r \text{ under } \underline{trans}\text{)} \\
= \; & \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, f(p_1 {p_2}^{l-1}, y_1, \ldots, y_n)) \\
& \qquad [y_j / nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, t_j) \; ; j \in [n]]) \\
& \quad (\underline{trans}(y_j) = y_j \quad (*)) \\
= \; & \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, f(p_1 {p_2}^{l-1}, t_1, \ldots, t_n))).
\end{aligned}$$

$(*)$: The correctness of this step can be proved by showing the statement

$$\begin{aligned}
& \underline{trans}(t)[y_j / \underline{trans}(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, t_j)) \; ; \; j \in [n]] \\
= \; & \underline{trans}(t[y_j / nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{l-1} - p_1 {p_2}^{l})}, t_j) \; ; j \in [n]])
\end{aligned}$$

for every $t \in SF(F, \{p_1 {p_2}^{l}\}, \Delta \cup Y_n)$ by structural induction. We omit this proof here and state only the reason $\underline{trans}(y_j) = y_j$ for every $j \in [n]$.

Let $r \geq 0$. We prove Statement (1b) by structural induction on the set $SF(F, \{p_1 {p_2}^{r}\}, \Delta)$. There are two cases:

Case I: For every $n \geq 0$, $\delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in SF(F, \{p_1 {p_2}^{r}\}, \Delta)$, such that $t = \delta(t_1, \ldots, t_n)$,

$$\begin{aligned}
& \underline{tree}_2(\underline{trans}(\delta(t_1, \ldots, t_n))) \\
= \; & \underline{tree}_2(\delta(\underline{trans}(t_1), \ldots, \underline{trans}(t_n))) && \text{(Definition of } \underline{trans}\text{)} \\
= \; & \delta(\underline{tree}_2(\underline{trans}(t_1)), \ldots, \underline{tree}_2(\underline{trans}(t_n))) && \text{(Definition of } \underline{tree}_2\text{)} \\
= \; & \delta(nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{r})}, t_1), \ldots, nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{r})}, t_n)) && \text{(Induction Hypothesis)} \\
= \; & nf(\Rightarrow_{e_r, P(e_r, p_1 {p_2}^{r})}, \delta(t_1, \ldots, t_n)).
\end{aligned}$$

Case II: For every $n \geq 0$, $f \in F^{(n+1)}$, and $t_1, \ldots, t_n \in SF(F, \{p_1 p_2{}^r\}, \Delta)$, such that $t = f(p_1 p_2{}^r, t_1, \ldots, t_n)$,

$$
\begin{array}{rl}
& \underline{tree}_2(\underline{trans}(f(p_1p_2{}^r, t_1, \ldots, t_n))) \\
= & \underline{tree}_2(f(\underline{trans}(t_1), \ldots, \underline{trans}(t_n))) \\
& \quad \text{(Definition of } \underline{trans}) \\
= & \hat{t}_f[y_j/\underline{tree}_2(\underline{trans}(t_j))\ ;\ j \in [n]] \\
& \quad \text{(Definition of } \underline{tree}_2) \\
= & nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))[y_j/\underline{tree}_2(\underline{trans}(t_j))\ ;\ j \in [n]] \\
& \quad \text{(Definition of } \hat{t}_f) \\
= & nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))[y_j/nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^r)}, t_j)\ ;\ j \in [n]] \\
& \quad \text{(Induction Hypothesis)} \\
= & nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^r)}, f(p_1p_2{}^r, y_1, \ldots, y_n))[y_j/nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^r)}, t_j)\ ;\ j \in [n]] \\
& \quad \text{(Same result on the occurrence of } e''' \text{ in } e \text{ and in } e_r) \\
= & nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^r)}, f(p_1p_2{}^r, t_1, \ldots, t_n)).
\end{array}
$$

Now we show that the following Statement (2) concerning the multiple application of the function $\underline{tree}_1$ holds:

(2) For every $r \geq 0$, $0 \leq l \leq r$, and $t \in SF(F, \{p_1\}, \Delta)$,

$$\underline{tree}_1{}^l(\underline{trans}(t)) \quad = \quad \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1 - p_1p_2{}^l)}, t))$$

Statement (2) relates the calculation on the topmost l occurrences of e'' in e_r with the l–fold application of the function $\underline{tree}_1$ as follows: The result of first calculating the normal form of a sentential form $t \in SF(F, \{p_1\}, \Delta)$ on the topmost l occurrences of e'' in e_r and then transforming it with $\underline{trans}$ is equal to first transforming t and then applying l times $\underline{tree}_1$.

Let $r \geq 0$ and $t \in SF(F, \{p_1\}, \Delta)$. We prove Statement (2) by finite (mathematical) induction on $0 \leq l \leq r$ with induction base $l = 0$:

$$
\begin{array}{rll}
& \underline{tree}_1{}^0(\underline{trans}(t)) & \\
= & \underline{trans}(t) & \\
= & \underline{trans}(nf(\Rightarrow_{e_r,\emptyset}, t)) & (t \text{ already normal form}) \\
= & \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1 - p_1p_2{}^0)}, t)) &
\end{array}
$$

In the case $1 \leq l \leq r$ we can prove as follows:

$$
\begin{array}{rll}
& \underline{tree}_1{}^l(\underline{trans}(t)) & \\
= & \underline{tree}_1(\underline{tree}_1{}^{l-1}(\underline{trans}(t))) & \text{(Definition of } \underline{tree}_1{}^l) \\
= & \underline{tree}_1(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1 - p_1p_2{}^{l-1})}, t))) & \text{(Induction Hypothesis)} \\
= & \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^{l-1} - p_1p_2{}^l)}, nf(\Rightarrow_{e_r,P(e_r,p_1 - p_1p_2{}^{l-1})}, t))) & \\
& & \text{(Statement (1a))} \\
= & \underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1 - p_1p_2{}^l)}, t)) &
\end{array}
$$

Then we can prove for every $r \geq 0$ the equation $\underline{tree}(r) = nf(\Rightarrow_{e_r}, \tilde{\rho}[x_1/\varepsilon])$:

$$
\begin{array}{lll}
& \underline{tree}(r) & \\
= & \underline{tree}_2(\underline{tree}_1{}^r(\hat{t})) & \text{(Definition of } \underline{tree}(r)) \\
= & \underline{tree}_2(\underline{tree}_1{}^r(\underline{trans}(nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) & \text{(Definition of } \hat{t}) \\
= & \underline{tree}_2(\underline{tree}_1{}^r(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) & (e' \text{ in } e_r \text{ unchanged}) \\
= & \underline{tree}_2(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1-p_1p_2{}^r)}, nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) & \\
& & \text{(Statement (2))} \\
= & \underline{tree}_2(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1p_2{}^r)}, \tilde{\rho}[x_1/\varepsilon]))) & \\
= & nf(\Rightarrow_{e_r,P(e_r,p_1p_2{}^r)}, nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1p_2{}^r)}, \tilde{\rho}[x_1/\varepsilon])) & \text{(Statement (1b))} \\
= & nf(\Rightarrow_{e_r}, \tilde{\rho}[x_1/\varepsilon]) &
\end{array}
$$

This equation has the following two desired consequences:

1. $\underline{tree}(1) = nf(\Rightarrow_{e_1}, \tilde{\rho}[x_1/\varepsilon]) = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon]) = t$.
2. For every $r \geq 0$, $\quad \underline{tree}(r) = nf(\Rightarrow_{e_r}, \tilde{\rho}[x_1/\varepsilon]) = \tau(M)(e_r) \in L_{out}(M)$. $\square$

We conclude this section with an observation concerning the requirements of the macro tree transducers to be producing and nondeleting.

The producing–condition does not affect the pumping process itself. But it is necessary for the proof that the output patterns consist of at least one output symbol. This result, however, is an existential fact for the proof of the growth–hierarchy of pumped output trees in the next section and thus for the applications in Section 6.

If we had dropped the nondeleting–condition, then the proof of the pumping lemma itself would have been impossible. Since for the input tree e and for every subpath p' of the chosen path p, $funset(e, p') \neq \emptyset$ cannot be guaranteed, the construction in the proof is no more feasible.

5 Growth of Output Trees

In this section we discuss the conditions under which we can guarantee a growth of output trees during the pumping process. This growth will be important for the applications which are presented in the next section. We shall define the extended pumping index of a macro tree transducer. If the size of an output tree is at least the extended pumping index, then pumping yields output trees forming a strict hierarchy concerning their sizes. This is the main additional statement of the extended pumping lemma which is proved in this section and which is only valid for noncopying macro tree transducers.

The first reason for separating this section from the previous section is to simplify the comprehension of this paper. An important aim of Section 4 was to explain the central ideas of the pumping lemma. An integration of the growth–property into that section would have disturbed the reader to much. The second reason for the separation is that here we have the additional restriction for macro tree transducers to be noncopying, whereas the pumping lemma of the previous section does not need this restriction.

5.1 Motivation

The two applications of the pumping lemma in this paper work according to the following principle: In order to show that a tree language L is no output language of a macro tree transducer we assume the contrary; i.e. there is a macro tree transducer M such that $L = L_{out}(M)$. Then we choose a tree $t \in L_{out}(M)$ with $size(t) \geq pi_M$. By the pumping lemma, the tree t can be built up from certain output patterns such that $t = \underline{tree}(1)$. Now we consider the output tree $\underline{tree}(0)$ and obtain the following contradiction: On the one hand the pumping lemma tells us that $\underline{tree}(0)$ also is an element of $L_{out}(M)$. On the other hand we can prove that this is not true. This proof uses among other things the *growth–property*

$$size(\underline{tree}(1)) > size(\underline{tree}(0)).$$

In this section we discuss sufficient conditions which imply $size(\underline{tree}(1)) > size(\underline{tree}(0))$. We shall see that the ultimately chosen conditions even imply the *general growth–property*

$$size(\underline{tree}(r+1)) > size(\underline{tree}(r)) \text{ for every } r \geq 0.$$

At first sight the growth–property seems to be a trivial consequence of the pumping process, because one could imagine that a growth of input trees automatically involves a growth of the corresponding output trees. But it turned out that our requirements of the previous section in general do not imply this growth–property and that the search for a sufficient condition has become one of the hardest items in the preparation of this paper. We demonstrate the difficulties with the following example:

Example 5.1 We define the macro tree transducer $M_2 = (F_2, \Delta_2, \Sigma_2, \tilde{\rho}_2, R_2)$ with:

$$\begin{aligned} F_2 &= \{f^{(2)}, g^{(2)}\}, \\ \Delta_2 &= \{A^{(1)}, E^{(0)}\}, \\ \Sigma_2 &= \{\gamma^{(1)}, \alpha^{(0)}\}, \\ \tilde{\rho}_2 &= f(x_1, f(x_1, g(x_1, E))), \text{ and} \end{aligned}$$

$$
\begin{array}{rllll}
R_2 & = & \{f(\gamma(x_1), y_1) & \to & Ag(x_1, y_1), & & (1)\\
& & f(\alpha, y_1) & \to & A^5 y_1, & & (2)\\
& & g(\gamma(x_1), y_1) & \to & Af(x_1, y_1), & & (3)\\
& & g(\alpha, y_1) & \to & Ay_1 & \}. & (4)
\end{array}
$$

Let $e = \gamma^n(\alpha)$ be an input tree with $n \geq 0$. The macro tree transducer M_2 uses simultaneous recursion, i.e. the function f (and the function g) calls on every node x of e, which is labeled by γ the function g (and the function f, respectively) on the child of x (cf. the rules (1) and (3), respectively). Since the initial right–hand side $\tilde{\rho}_2$ contains two function occurrences (f, x_1) and only one function occurrence (g, x_1), for every i with $0 \leq i \leq n$ we have $funseq(e, 1^i) = ffg$, if i is even, and $funseq(e, 1^i) = ggf$, if i is odd. Thus, if n is even, then the (f, α)–rule (and the (g, α)–rule) is applied twice (and once, respectively). If n is odd, then the (f, α)–rule (and the (g, α)–rule) is applied once (and twice, respectively). Because of the different numbers of output symbols in $rhs(f, \alpha)$ and $rhs(g, \alpha)$ (cf. the rules (2) and (4)), roughly speaking, the size of output trees pulsates in dependence of the size of input trees:

The tree transformation $\tau(M_2)$ can be described for every $n \geq 0$ as follows:

- If n is even, then $\tau(M_2)(\gamma^n(\alpha)) = A^{3n+11}E$.
- If n is odd, then $\tau(M_2)(\gamma^n(\alpha)) = A^{3n+7}E$.

Thus, for every $n \geq 0$ the following two conditions hold:

- If n is even, then
  $size(\tau(M_2)(\gamma^{n+1}(\alpha))) = 3n + 11 < 3n + 12 = size(\tau(M_2)(\gamma^n(\alpha)))$.
- If n is odd, then
  $size(\tau(M_2)(\gamma^{n+1}(\alpha))) = 3n + 15 > 3n + 8 = size(\tau(M_2)(\gamma^n(\alpha)))$.

Though already the first of the above conditions demonstrates that a growth of input trees does not imply a growth of output trees, we want to show the problems also in the context of the pumping lemma:

Since $maxheight_{M_2,R_2} = 6$, $card(F_2) = 2$, and $maxrank_{M_2,\Delta_2} = 1$, we have the pumping index $pi_{M_2} = pi'_{M_2} = 1 + 6^5 = 7777$ (cf. Definition 4.2).

Thus, if we choose for example the output tree $t = A^{7780}E \in L_{out}(M_2)$ with $size(t) = 7781 \geq pi_{M_2}$, then we can follow up the main steps of the proof of Theorem 4.4. We inspect the input tree $e = \gamma^{2591}(\alpha)$ with $\tilde{\rho}_2[x_1/\varepsilon] \Rightarrow_e^+ t$ and the (unique) path $p = 1^{2591} \in paths(e)$ with maximum length. Note that 2591 is odd. Since for every path $p' \in paths(e)$ such that there is a string p'' with $p = p'p''$ the condition $funset(e, p') = \{f, g\}$ holds, we can choose $p_1 = 1^{2590}$, $p_2 = 1$, and $p_3 = \varepsilon$. Thus $e' = \gamma^{2590}(w)$, $e'' = \gamma(w)$ and $e''' = \alpha$ and we obtain $e_0 = e'[w/e'''] = \gamma^{2590}(\alpha)$ and $\underline{tree}(0) = nf(\Rightarrow_{e_0}, \tilde{\rho}_2[x_1/\varepsilon]) = A^{7781}E$, because 2590 is even. Now we have found the result $size(\underline{tree}(1)) = size(t) = 7781 < 7782 = size(\underline{tree}(0))$ which demonstrates our difficulties. □

Note that the problems do not depend on our way of defining macro tree transducers, because the initial sentential form $\tilde{\rho}_2[x_1/\varepsilon]$ can also be generated by usual macro tree transducers (cf. also the discussion after Definition 3.1).

We think that the same problems also arise in the proof of the pumping lemma for top–down tree–to–string transducers in [ERS80]. It can be demonstrated by slightly extending and modifying Example 5.1 that the proof detail which shows the growth of strings in general is not true: It is not clear that every repetition of the pumped input pattern (denoted by t_1 in [ERS80]) produces at least one more output symbol, if at least one pumped output string (denoted by w_i in [ERS80]) contains an output symbol. Nevertheless we think that the pumping lemma itself is correct and that the proof can be extended may be with the same technique as in this paper. At this point we have to repeat explicitely that the pumping lemma in [ERS80] had a big influence on our pumping lemmata.

From Example 5.1 we have learned that our fundamental condition

$$funset(e, p_1) = funset(e, p_1p_2)$$

is not sufficient to prove the growth–property. In this example the function–sequences ffg and ggf alternate on the path from the root of an input tree to the leaf node. If we choose p_1 and p_2, such that $funseq(e, p_1) \neq funseq(e, p_1p_2)$, as it was done in the example, then the above problem can arise from the different number of occurrences of the function f (and g, respectively) in the function–sequences. A solution for this example would be to choose p_2 with an even length. Hence, since the function–sequences alternate, we have $funseq(e, p_1) = funseq(e, p_1p_2)$ and the problem disappears.

But in general, a substitution of the condition $funset(e, p_1) = funset(e, p_1p_2)$ by the stronger condition

$$funseq(e, p_1) = funseq(e, p_1p_2)$$

is not appropriate, because then it would be impossible to handle macro tree transducers which have no repetitions of function–sequences on paths of input trees. In contrast to the function–sets, there is no estimation for the number of possible function–sequences, because their lengths are not limited.

The following condition weakens the above condition: For every function $f \in F$, the number of occurrences of f in $funseq(e, p_1p_2)$ has to be at least the number of occurrences of f in $funseq(e, p_1)$. This fact implies the growth–property $size(\underline{tree}(1)) > size(\underline{tree}(0))$, because, roughly speaking, $\underline{tree}(1)$ contains at least as many of the different output patterns as $\underline{tree}(0)$. This condition can be better expressed with the notion function–multiset (cf. Definition 3.10): If we choose strings p_1 and p_2, such that for every function $f \in F$ the condition

$$funmulti(e, p_1p_2)(f) \geq funmulti(e, p_1)(f)$$

holds, then we have the result $size(\underline{tree}(1)) > size(\underline{tree}(0))$. For example, the condition $funmulti(e, p_1p_2)(f) \geq funmulti(e, p_1)(f)$ would be satisfied, if we could prove that every function occurrence (f, p_1) in a derivation causes at least one function occurrence (f, p_1p_2).

We do not follow here this idea, because we failed till now in proving the existence of such strings for sufficiently large output trees and in defining a pumping index which specifies the term "sufficiently large". Nevertheless, we conjecture that this proof is possible and that therefore the above condition for function–multisets would be another possible starting point.

In order to prove the growth–property, we follow now another course. First, we informally develop a sufficient condition for the growth–property $size(\underline{tree}(1)) > size(\underline{tree}(0))$. In the proof of the extended pumping lemma we shall see that this condition also implies the general growth–property $size(\underline{tree}(r+1)) > size(\underline{tree}(r))$ for every $r \geq 0$.

Recalling the motivation of the pumping lemma from Section 4, we can give a rough comparison between the sizes of $\underline{tree}(1)$ and $\underline{tree}(0)$. This comparison leads to a relation between the lengths of the subpaths p_2 and p_3. We mention already here that the comparison depends heavily on the noncopying–condition for macro tree transducers and explain the reason after the last of the following items.

- The trees $\underline{tree}(1)$ and $\underline{tree}(0)$ both contain the output pattern $\hat{t}$. Thus, they do not differ in their parts which are induced by the input pattern e'. Hence, the size comparison does not have to take the size of $\hat{t}$ into account.

- The tree $\underline{tree}(1)$ contains for every $f \in funset(e, p_1)$ occurrences of the output pattern t_f which are induced by the input pattern e''. Each of these output patterns has at least the size $|p_2|$, because the nondeleting–condition implies that at least one rule has to be applied at every node of the subpath p_2 to calculate an output pattern t_f. Because of the producing–condition every rule application delivers at least one output symbol.

  Since $funnum(e, p_1)$ is the total number of occurrences of such output patterns in $\underline{tree}(1)$, the input pattern e'' causes at least $funnum(e, p_1) \cdot |p_2|$ output symbols in $\underline{tree}(1)$.

- The tree $\underline{tree}(1)$ also contains for every $f \in funset(e, p_1 p_2)$ occurrences of the output pattern $\hat{t}_f$ which are induced by the input pattern e'''. According to the pumping lemma of the previous section, each of these output patterns has at least the size 1. This estimation suffices here.

  Since $funnum(e, p_1 p_2)$ is the total number of occurrences of such output patterns in $\underline{tree}(1)$, the input pattern e''' causes at least $funnum(e, p_1 p_2) \cdot 1$ output symbols in $\underline{tree}(1)$.

- The tree $\underline{tree}(0)$ contains for every $f \in funset(e, p_1)$ occurrences of the output pattern $\hat{t}_f$ which are induced by the input pattern e'''. Since p_3 is a longest path of e''', we have $height(e''') = |p_3|+1$. Then Observation 3.14 which relates the heights of input and output trees delivers that every such output pattern has at most the height $(maxheight_{M,R})^{|p_3|+2}$. The relation between heights and sizes of trees induces that the output patterns have at most the size $\sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$.

  Since $funnum(e, p_1)$ is the total number of occurrences of such output patterns in $\underline{tree}(0)$, the number of output symbols in $\underline{tree}(0)$ which are caused by the input pattern e''', is at most $funnum(e, p_1) \cdot \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$.

The estimation in the last item depends on the noncopying–condition, because for example, multiple occurrences of one parameter in a tree $\hat{t}_f$ and nested occurrences of f in $\hat{t}$ can cause more than $funnum(e, p_1)$ occurrences of (parts of) output patterns $\hat{t}_f$ in $\underline{tree}(0)$.

The statement of the first item also would be wrong without the noncopying–condition, because parts of $\hat{t}$ can occur more than once in $\underline{tree}(0)$ and $\underline{tree}(1)$. We only mention that of course there are also similar estimations for copying macro tree transducers, but till now we do not have found estimations such that we can deduce the growth–property.

Summarizing the above four results, we obtain the following estimation for the difference between the sizes of $\underline{tree}(1)$ and $\underline{tree}(0)$:

$$\begin{array}{ll} & size(\underline{tree}(1)) - size(\underline{tree}(0)) \\ \geq & funnum(e,p_1) \cdot |p_2| + funnum(e,p_1p_2) \\ & -funnum(e,p_1) \cdot \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k \\ = & funnum(e,p_1) \cdot (|p_2| - \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k) + funnum(e,p_1p_2) \end{array}$$

If the last expression in the above estimation is greater than 0, then we can deduce $size(\underline{tree}(1)) - size(\underline{tree}(0)) > 0$ which represents our desired growth–property. Because of Lemma 3.15, we have $funnum(e,p_1) > 0$ and $funnum(e,p_1p_2) > 0$. Thus,

$$|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$$

is a sufficient condition for the growth–property. Since $maxheight_{M,R}$ and $maxrank_{M,\Delta}$ are constants for a certain macro tree transducer M, this condition tells us the minimum length of p_2 in dependence of the length of p_3. Thus we call this condition *subpath relation.*

In the next subsection we shall use the function $dis_M : I\!N \longrightarrow I\!N$ which calculates the minimum length of p_2 for the length of p_3, such that the subpath relation holds. Because of the subpath relation, for every $n \in I\!N$ this function is defined by $dis_M(n) = \sum_{k=0}^{(maxheight_{M,R})^{n+2}-1} (maxrank_{M,\Delta})^k$. It is called the distance function of a macro tree transducer M, because it delivers the minimum distance $|p_2|$ between the important nodes $node(e,p_1)$ and $node(e,p_1p_2)$ in an input tree e.

Clearly, the subpath relation is only one condition that has to be fulfilled by the paths p_2 and p_3. We still have to obey also our condition $funset(e,p_1) = funset(e,p_1p_2)$, such that we can adopt the main ideas from Section 4. Now the questions arise, how these two conditions can be combined and how a new pumping index can be constructed from this combination. Since the pumping index depends on the minimum height of input trees and thus on the minimum length of a longest path in input trees, we can ask more precisely:

How long must a path p in an input tree e be, such that there are strings p_1, p_2, and p_3 with

- $p = p_1p_2p_3$,
- $funset(e,p_1) = funset(e,p_1p_2)$, and
- $|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$?

To answer this question, we use the following principle: Starting from the leaf $node(e,p)$ of e we move step by step back to the root of e visiting only those nodes on the path p with a certain distance to the previously visited node. The distance between one visited node x and the next visited node x' is given by an application of the distance function dis_M to the distance between x and the leaf $node(e,p)$. Since the distance function represents the subpath relation between the path from x' to x and the path from x to the leaf, we can guarantee the following fact: Even for two arbitrary, visited, and distinct nodes $x_1 = node(e,p_1)$ and $x_2 = node(e,p_1p_2)$, where p_1, p_2, and p_3 are strings with $p = p_1p_2p_3$, the subpath relation $|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$ holds.

On the left side of Figure 9 we show a path p of an input tree e leading to a leaf node x. Three steps (marked with (1), (2), and (3)) of our visiting principle yield in turn the three nodes x', x'', and x''' on the path p which induce the indicated decomposition $p = \tilde{p}p'''p''p'$. To explain the subpath relations in the three steps, we represent in the center of the figure the lengths of the relevant paths by dashed, vertical lines with borders. Now we choose for example $x_1 = x'''$, $x_2 = x'$ and thus $p_1 = \tilde{p}$, $p_2 = p'''p''$, and $p_3 = p'$. This choice is indicated in the repetition of the the input tree e on the right side of the figure. The paths p_2 and p_3 also fulfill the subpath relation, because $|p_2| = |p'''p''| \geq |p''| = dis_M(|p'|) = dis_M(|p_3|)$.

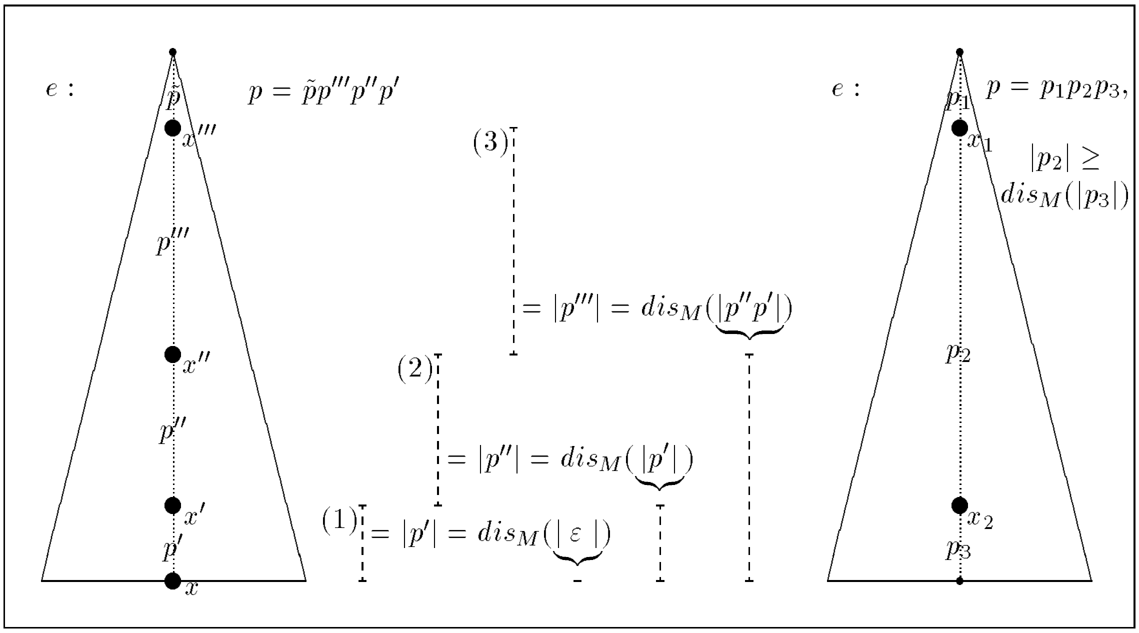

Figure 9: Nodes on a path p determined by the subpath relation.

Now we recall that there are $2^{card(F)} - 1$ possibilities to choose a nonempty subset of the set F of functions. If the path p is long enough, such that we can visit $2^{card(F)}$ nodes on it and these nodes have the distance property according to the above described principle, then there must exist two different nodes $x_1 = node(e,p_1)$ and $x_2 = node(e,p_1p_2)$ of them with $funset(e,p_1) = funset(e,p_1p_2)$ in addition to the subpath relation between p_2 and p_3.

We also can calculate now the required length of the path p: Since we start with the leaf node and have to visit $2^{card(F)} - 1$ further nodes, we have to apply $2^{card(F)} - 1$ times the distance function. We call the distance between the leaf $node(e,p)$ and an arbitrary node

x on the path p the position of x. Thus, the position of a visited node can be calculated recursively as follows: If the node is the leaf, then its position is 0. If the node is not the leaf, then we add the position of the previously visited node and the result of the application of the distance function to the position of the previously visited node. This will be described later on by the position function $pos_M : I\!N \longrightarrow I\!N$ of M which calculates for every $n \geq 0$ the position of the $(n+1)$th visited node. It is recursively defined by $pos_M(0) = 0$ and for every $n \in I\!N$, $pos_M(n+1) = dis_M(pos_M(n)) + pos_M(n)$. Thus, $pos_M(2^{card(F)} - 1)$ is the required length of p. Now a new pumping index can be defined according to this length of p in analogy to Section 4.

We want to finish this motivation with an already mentioned observation. The subpath relation technique, as it was presented above, has not yet proved to be sufficient to show the growth–property for every producing and nondeleting macro tree transducer, but it is sufficient for producing, nondeleting, and noncopying macro tree transducers. Since we only apply our pumping lemmata on monadic languages in the next section, the noncopying–condition is no further restriction, as it was outlined in Observation 3.19. Note that since in this case there is at most one occurrence of a parameter in every right–hand side of a rule, there is at most one occurrence of a parameter in every output pattern.

5.2 Subpath Relation of Input Trees

We first define the distance function and the position function, which have been explained in the previous subsection.

Definition 5.2 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer.

The *distance function* $dis_M : I\!N \longrightarrow I\!N$ *of* M is defined as follows:
For every $n \in I\!N$,

$$dis_M(n) = \sum_{k=0}^{(maxheight_{M,R})^{n+2}-1} (maxrank_{M,\Delta})^k.$$

The *position function* $pos_M : I\!N \longrightarrow I\!N$ *of* M is defined as follows:

$$pos_M(0) = 0 \quad \text{and} \quad \text{for every } n \in I\!N, \quad pos_M(n+1) = dis_M(pos_M(n)) + pos_M(n). \qquad \square$$

In the following lemma we formalize the principle of the previous subsection to visit nodes on a path according to the distance function. The lemma is a generalization of the well known pigeon hole principle: If S is a nonempty set and $seq = s_0 s_1 \ldots s_l$ is a string with $l \geq card(S)$ and $s_i \in S$ for every $i \in [l] \cup \{0\}$, then there are i, j with $0 \leq i < j \leq card(S)$, such that $s_i = s_j$. This principle has already been used in the proof of Theorem 4.4 to show the repetition of a function–set. Besides this repetition of an element of S, the lemma additionally guarantees a certain distance $j - i$ between s_i and s_j.

Lemma 5.3 Let S be a nonempty set, $l \in I\!N$, and $seq = s_0 s_1 \dots s_l$ be a string with $s_i \in S$ for every $i \in [l] \cup \{0\}$. Let $dis : I\!N \longrightarrow I\!N$ be a function with $dis(n) \geq 1$ for every $n \in I\!N$.

If $l \geq pos(card(S))$, where the function $pos : I\!N \longrightarrow I\!N$ is defined by $pos(0) = 0$ and for every $n \in I\!N$, $pos(n+1) = dis(pos(n)) + pos(n)$,

then there are $i, j \in I\!N$ with $0 \leq i < j \leq pos(card(S))$, such that $s_i = s_j$ and $j - i \geq dis(i)$.

Proof. Let $seq = s_0 s_1 \dots s_l$ with $l \geq pos(card(S))$.

Clearly, for every $n \in I\!N$, $pos(n) \geq 0$ holds. Additionally, the function pos is a strong monotonous function, i.e. for every $n \in I\!N$ the condition $pos(n+1) > pos(n)$ holds, because $dis(n) \geq 1$ for every $n \in I\!N$ and thus, since $pos(n) \geq 0$ holds, $pos(n+1) = dis(pos(n)) + pos(n) \geq pos(n) + 1 > pos(n)$.

Since $l \geq pos(card(S))$ and pos is strong monotonous, we can consider the sequence $seq' = s_{pos(0)} s_{pos(1)} \dots s_{pos(card(S))}$ such that for every i with $0 \leq i \leq card(S)$, there is a j with $0 \leq j \leq l$ and $s_{pos(i)} = s_j$, i.e. seq' contains elements of seq. Since $|seq'| = card(S)+1$, there must exist i' and j' with $0 \leq i' < j' \leq card(S)$, such that $s_{pos(i')} = s_{pos(j')}$. We choose $i = pos(i')$ and $j = pos(j')$. Then we have

- $0 \leq i < j \leq pos(card(S))$, because $0 \leq i' < j' \leq card(S)$ and thus, since pos is strong monotonous, $0 = pos(0) \leq pos(i') < pos(j') \leq pos(card(S))$,
- $s_i = s_j$, because $s_{pos(i')} = s_{pos(j')}$, and
- $$\begin{array}{lll} & j - i & \\ = & pos(j') - pos(i') & \text{(Definition of } i, j) \\ \geq & pos(i'+1) - pos(i') & (j' \geq i'+1 \text{ and } pos \text{ strong monotonous}) \\ = & dis(pos(i')) + pos(i') - pos(i') & \text{(Definition of } pos(i'+1)) \\ = & dis(i) & \text{(Definition of } i). \end{array}$$

□

Example 5.4 Let $seq = s_0 s_1 \dots s_l$ with $s_i \in S$ for every $i \in [l] \cup \{0\}$.

- We choose $dis(n) = 1$ for every $n \in I\!N$ and thus $pos(0) = 0$ and for every $n \in I\!N$, $pos(n+1) = dis(pos(n)) + pos(n) = pos(n) + 1$. Then we simply have $pos(n) = n$ for every $n \in I\!N$.

  If $l \geq pos(card(S)) = card(S)$, then there are $i, j \in I\!N$ with $0 \leq i < j \leq pos(card(S)) = card(S)$, such that $s_i = s_j$ and $j - i \geq dis(i) = 1$. This corresponds to the pigeon hole principle.

- We choose $dis(n) = n+1$ for every $n \in I\!N$ and thus $pos(0) = 0$ and for every $n \in I\!N$, $pos(n+1) = dis(pos(n)) + pos(n) = 2 \cdot pos(n) + 1$. It can be shown by induction that $pos(n) = 2^n - 1$ holds for every $n \in I\!N$.

  If $l \geq pos(card(S)) = 2^{card(S)} - 1$, then there are $i, j \in I\!N$ with $0 \leq i < j \leq pos(card(S)) = 2^{card(S)} - 1$, such that $s_i = s_j$ and $j - i \geq dis(i) = i + 1$. This corresponds to the situation, in which the distance $j - i$ between s_j and s_i has to be greater than the distance i between s_i and s_0. □

Now we are able to proof Lemma 5.5 which guarantees for a sufficiently high input tree not only a repetition of a function–set on a path, but also the subpath relation, i.e. the relation between the lengths of the subpaths which is needed to prove the growth–property in the pumping lemma.

The reader should compare Lemma 5.5 with the properties of the strings p_1, p_2, and p_3 at the beginning of the proof of Theorem 4.4. The subpath relation between $|p_2|$ and $|p_3|$ is given by the second item of the conclusion of Lemma 5.5. Additionally, the third item has changed: the upper bound for the length of p_2p_3 has increased by an application of the position function pos_M. Nevertheless we are able to prove later the size conditions for the output patterns, because the additional application of pos_M will also be coded into the extended pumping index.

Lemma 5.5 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer, $e \in T\langle\Sigma\rangle$ and $p \in paths(e)$ a longest path in e from the root to a leaf. Let pos_M be the position function of M.

If $height(e) \geq pos_M(2^{card(F)} - 1) + 1$ then there exist strings p_1, p_2, and p_3, with

- $p = p_1p_2p_3$,
- $|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1}(maxrank_{M,\Delta})^k \geq 1$,
- $|p_2p_3| \leq pos_M(2^{card(F)} - 1)$, and
- $funset(e, p_1) = funset(e, p_1p_2)$.

Proof. Let $p = k_1k_2 \ldots k_{|p|}$ with $k_i \in I\!N_+$ for every $i \in [|p|]$. Since $height(e) \geq pos_M(2^{card(F)} - 1) + 1$, we have $|p| \geq pos_M(2^{card(F)} - 1)$.

To apply Lemma 5.3, we define $S = \mathcal{P}(F) - \{\emptyset\}$, $l = |p|$, and $seq = funset_0 \ldots funset_l$, where $funset_i = funset(e, k_1 \ldots k_{|p|-i})$ for every $i \in [l] \cup \{0\}$. Then $funset_i \in S$ for every $i \in [l] \cup \{0\}$. Let dis_M be the distance function of M. Since M contains at least one output symbol δ with $rank(\delta) > 0$ (otherwise the height of every output tree would be 0) $maxrank_{M,\Delta} \geq 1$ holds. Since the height of every right–hand side of a rule is at least 1, we have $maxheight_{M,R} \geq 1$. These facts deliver $dis_M(n) \geq 1$ for every $n \in I\!N$.

The requirements of Lemma 5.3 are fulfilled, because $l = |p| \geq pos_M(2^{card(F)} - 1) = pos_M(card(S))$ and pos_M is defined with the help of dis_M according to Lemma 5.3.

Thus there are $i, j \in I\!N$ with $0 \leq i < j \leq pos_M(card(S)) = pos_M(2^{card(F)} - 1)$, such that $funset_i = funset_j$ and $j - i \geq dis_M(i)$. We choose

- $p_1 = k_1 \ldots k_{|p|-j}$,
- $p_2 = k_{|p|-j+1} \ldots k_{|p|-i}$, and
- $p_3 = k_{|p|-i+1} \ldots k_{|p|}$.

Then we have

- $p = k_1 \dots k_{|p|} = p_1 p_2 p_3$,
- $|p_2| = j - i \geq dis_M(i) = \sum_{k=0}^{(maxheight_{M,R})^{i+2}-1} (maxrank_{M,\Delta})^k$
$= \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k \geq 1$,
- $|p_2 p_3| = |k_{|p|-j+1} \dots k_{|p|}| = j \leq pos_M(card(S)) = pos_M(2^{card(F)} - 1)$, and
- $funset(e, p_1) = funset(e, k_1 \dots k_{|p|-j}) = funset_j = funset_i$
$= funset(e, k_1 \dots k_{|p|-i}) = funset(e, p_1 p_2)$. □

5.3 Extended Pumping Index

In analogy to the pumping index pi_M of a macro tree transducer M in Definition 4.2 we define the extended pumping index $epi_M \in I\!N$, which additionally guarantees the desired growth–property. In Subsection 5.2 we have seen that for a macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ and an input tree $e \in T\langle\Sigma\rangle$ we need at least the height $pos_M(2^{card(F)} - 1) + 1$ of e. This height can be guaranteed, if the size of the output tree $t = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon])$ is at least epi_M. For later purposes, similar to Section 4, we define epi_M, such that we can prove $height(e) \geq pos_M(2^{card(F)} - 1) + 2$.

Definition 5.6 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer and let pos_M be the position function of M.

The *extended pumping index* epi_M *of* M is defined by

$$epi_M = 1 + \sum_{i=0}^{epi'_M - 2} (maxrank_{M,\Delta})^i, \quad \text{where}$$

$$epi'_M = 1 + (maxheight_{M,R})^{pos_M(2^{card(F)}-1)+2}.$$ □

In analogy to Lemma 4.3 we state the following lemma. The proof of this lemma can be obtained by the proof of Lemma 4.3, substituting $2^{card(F)} + 1$ by $pos_M(2^{card(F)} - 1) + 2$. Thus the proof is left out here.

Lemma 5.7 Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a macro tree transducer with position function pos_M and extended pumping index epi_M. Let $t \in L_{out}(M)$.

If $size(t) \geq epi_M$, then for every $e \in T\langle\Sigma\rangle$ such that $t = nf(\Rightarrow_e, \tilde{\rho}[x_1/\varepsilon])$, the height $height(e) \geq pos_M(2^{card(F)} - 1) + 2$. □

5.4 Extended Pumping Lemma and its Proof

We already have mentioned at the beginning of this section that we can only prove the growth–property with the subpath relation technique, if we restrict ourselves to noncopying macro tree transducers. Thus the notion "extended" refers to the additional ability to prove the growth–property. Clearly, this pumping lemma is also "restricted", if we emphasize the shrinked class of tree transducers, for which the pumping lemma is valid.

Besides the general growth–property, Theorem 5.8 additionally states for every $r \geq 0$ equations for $size(\underline{tree}(r))$ and $size(\underline{tree}(r+1))$. They will also be used in the next section and they also need the noncopying–condition.

Theorem 5.8 (Extended Pumping Lemma)
Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a nondeleting, noncopying, and producing macro tree transducer with extended pumping index epi_M.

For every $t \in L_{out}(M)$ with $size(t) \geq epi_M$

- there exists a ranked alphabet $(U, rank_U)$ with $U \subseteq F$, $card(U) \geq 1$, and $rank_U(f) = rank_F(f) - 1$ for every $f \in U$,
- there exists a tree $\hat{t} \in T\langle \Delta \cup U \rangle$ with $size_\Delta(\hat{t}) \geq 1$,
- for every $f \in U$, there exists a tree $t_f \in T\langle \Delta \cup U \rangle(Y_{rank_U(f)})$ with $1 \leq size_\Delta(t_f) < epi_M$ and $size_U(t_f) < epi_M$,
- for every $f \in U$, there exists a tree $\hat{t}_f \in T\langle \Delta \rangle(Y_{rank_U(f)})$ with $1 \leq size_\Delta(\hat{t}_f) < epi_M$,
- for every $f \in U$, the symbol f occurs in $\hat{t}$,
- for every $f \in U$, there is an $f' \in U$ such that f occurs in $t_{f'}$,
- for every $f \in U$ and for every $y \in Y_{rank_U(f)}$, the symbol y occurs exactly once in t_f and $\hat{t}_f$,

such that

1. $t = \underline{tree}(1)$,
2. for every $r \geq 0$,
   (a) $\underline{tree}(r) \in L_{out}(M)$,
   (b) $size(\underline{tree}(r))$
   $$= size_\Delta(\underline{tree_1}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree_1}^r(\hat{t})) \cdot size_\Delta(\hat{t}_f)),$$
   (c) $size(\underline{tree}(r+1))$
   $$= size_\Delta(\underline{tree_1}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree_1}^r(\hat{t})) \cdot (size_\Delta(t_f) + \sum_{g \in U}(size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g)))),$$ and
   (d) $size(\underline{tree}(r+1)) > size(\underline{tree}(r))$,

where the functions $\underline{tree}$, $\underline{tree_1}$, and $\underline{tree_2}$ are defined as in Theorem 4.4.

Proof. We only show those parts of the proof which are different from the proof of Theorem 4.4 or which do not appear there.

Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a noncopying macro tree transducer with extended pumping index epi_M.

Consider $t \in L_{out}(M)$ with $size(t) \geq epi_M$. By Lemma 5.7 we know that, for every input tree e with $\tilde{\rho}[x_1/\varepsilon] \Rightarrow_e^+ t$, the condition $height(e) \geq pos_M(2^{card(F)} - 1) + 2$ holds.

We choose an arbitrary input tree e and a path $p \in paths(e)$ from the root of e to a leaf of e with maximum length $|p| \geq pos_M(2^{card(F)} - 1) + 1$. By Lemma 5.5 we know that there are strings p_1, p_2, and p_3, with

- $p = p_1 p_2 p_3$,
- $|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k \geq 1$,
- $|p_2 p_3| \leq pos_M(2^{card(F)} - 1)$, and
- $funset(e, p_1) = funset(e, p_1 p_2)$.

We define the ranked alphabet $(U, rank_U)$, the input patterns e', e'', and e''', the function $\underline{trans}$, the output pattern $\hat{t}$ and, for every $f \in U$, the output patterns t_f and $\hat{t}_f$ as in the proof of Theorem 4.4.

The proofs of the size conditions carry over to here with a slight modification, substituting $2^{card(F)}$ by $pos_M(2^{card(F)} - 1) + 1$, pi'_M by epi'_M, and, clearly, pi_M by epi_M.

In the proof of Theorem 4.4 it was shown that, for every $f \in U$ and for every $y \in Y_{rank_U(f)}$, the symbol y occurs at least once in t_f and $\hat{t}_f$. Here the symbol y occurs exactly once in t_f and $\hat{t}_f$, because M is additionally noncopying, and thus in the definition of t_f and $\hat{t}_f$ by normal forms no symbol y can be copied.

To prove the Statements 2.(b), 2.(c), and 2.(d) we first have to verify the following Statements (3a), (3b), and (3c) concerning the sizes of trees which are transformed by the functions $\underline{tree}_2$ and $\underline{tree}_1$:

(3a) For every $t \in T\langle \Delta \cup U \rangle$,
$size_\Delta(\underline{tree}_2(t)) \quad = \quad size_\Delta(t) + \sum_{f \in U}(size_{\{f\}}(t) \cdot size_\Delta(\hat{t}_f))$.

(3b) For every $t \in T\langle \Delta \cup U \rangle$,
$size_\Delta(\underline{tree}_1(t)) \quad = \quad size_\Delta(t) + \sum_{f \in U}(size_{\{f\}}(t) \cdot size_\Delta(t_f))$.

(3c) For every $t \in T\langle \Delta \cup U \rangle$ and $g \in U$,
$size_{\{g\}}(\underline{tree}_1(t)) \quad = \quad \sum_{f \in U}(size_{\{f\}}(t) \cdot size_{\{g\}}(t_f))$.

We prove the Statements (3a) and (3c) by structural induction on the set $T\langle \Delta \cup U \rangle$. The proof of Statement (3b) is completely analogous to that of Statement (3a) and therefore it is left out. For the proof of Statement (3a) we have to check two cases:

Case I: For every $n \geq 0$, $\delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in T\langle \Delta \cup U \rangle$, such that $t = \delta(t_1, \ldots, t_n)$,

$$\begin{array}{lll}
& size_\Delta(\underline{tree}_2(\delta(t_1, \ldots, t_n))) & \\
= & size_\Delta(\delta(\underline{tree}_2(t_1), \ldots, \underline{tree}_2(t_n))) & \text{(Definition of } \underline{tree}_2\text{)} \\
= & 1 + \sum_{i=1}^{n}(size_\Delta(\underline{tree}_2(t_i))) & \text{(Definition of } size_\Delta\text{)} \\
= & 1 + \sum_{i=1}^{n}(size_\Delta(t_i) + \sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_\Delta(\hat{t}_f))) & \text{(Induction Hypothesis)} \\
= & size_\Delta(\delta(t_1, \ldots, t_n)) + \sum_{i=1}^{n}(\sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_\Delta(\hat{t}_f))) & \text{(Definition of } size_\Delta\text{)} \\
= & size_\Delta(\delta(t_1, \ldots, t_n)) + \sum_{f \in U}(\sum_{i=1}^{n}(size_{\{f\}}(t_i)) \cdot size_\Delta(\hat{t}_f)) & \text{(Distributivity)} \\
= & size_\Delta(\delta(t_1, \ldots, t_n)) + \sum_{f \in U}(size_{\{f\}}(\delta(t_1, \ldots, t_n)) \cdot size_\Delta(\hat{t}_f)) & \\
& & \text{(Definition of } size_{\{f\}}\text{).}
\end{array}$$

Case II: For every $n \geq 0$, $f' \in U^{(n)}$, and $t_1, \ldots, t_n \in T\langle \Delta \cup U \rangle$, such that $t = f'(t_1, \ldots, t_n)$,

$$\begin{array}{lll}
& size_\Delta(\underline{tree}_2(f'(t_1, \ldots, t_n))) & \\
= & size_\Delta(\hat{t}_{f'}[y_i/\underline{tree}_2(t_i)\ ;\ i \in [n]]) & \text{(Definition of } \underline{tree}_2\text{)} \\
= & size_\Delta(\hat{t}_{f'}) + \sum_{i=1}^{n}(size_\Delta(\underline{tree}_2(t_i))) & \text{(Def. of } size_\Delta,\ M \text{ nondeleting, noncopying)} \\
= & size_\Delta(\hat{t}_{f'}) + \sum_{i=1}^{n}(size_\Delta(t_i) + \sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_\Delta(\hat{t}_f))) & \\
& & \text{(Induction Hypothesis)} \\
= & size_\Delta(\hat{t}_{f'}) + size_\Delta(f'(t_1, \ldots, t_n)) + \sum_{i=1}^{n}(\sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_\Delta(\hat{t}_f))) & \\
& & \text{(Definition of } size_\Delta\text{)} \\
= & size_\Delta(f'(t_1, \ldots, t_n)) + size_\Delta(\hat{t}_{f'}) + \sum_{f \in U}(\sum_{i=1}^{n}(size_{\{f\}}(t_i)) \cdot size_\Delta(\hat{t}_f)) & \\
& & \text{(Distributivity)} \\
= & size_\Delta(f'(t_1, \ldots, t_n)) + \sum_{f \in U}(size_{\{f\}}(f'(t_1, \ldots, t_n)) \cdot size_\Delta(\hat{t}_f)) & \\
& & \text{(Definition of } size_{\{f\}}\text{).}
\end{array}$$

For the proof of Statement (3c) we again have to check two cases:

Case I: For every $n \geq 0$, $\delta \in \Delta^{(n)}$, and $t_1, \ldots, t_n \in T\langle \Delta \cup U \rangle$, such that $t = \delta(t_1, \ldots, t_n)$,

$$\begin{array}{lll}
& size_{\{g\}}(\underline{tree}_1(\delta(t_1, \ldots, t_n))) & \\
= & size_{\{g\}}(\delta(\underline{tree}_1(t_1), \ldots, \underline{tree}_1(t_n))) & \text{(Definition of } \underline{tree}_1\text{)} \\
= & \sum_{i=1}^{n}(size_{\{g\}}(\underline{tree}_1(t_i))) & \text{(Definition of } size_{\{g\}}\text{)} \\
= & \sum_{i=1}^{n}(\sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_{\{g\}}(t_f))) & \text{(Induction Hypothesis)} \\
= & \sum_{f \in U}(\sum_{i=1}^{n}(size_{\{f\}}(t_i)) \cdot size_{\{g\}}(t_f)) & \text{(Distributivity)} \\
= & \sum_{f \in U}(size_{\{f\}}(\delta(t_1, \ldots, t_n)) \cdot size_{\{g\}}(t_f)) & \text{(Definition of } size_{\{f\}}\text{).}
\end{array}$$

Case II: For every $n \geq 0$, $f' \in U^{(n)}$, and $t_1, \ldots, t_n \in T\langle \Delta \cup U \rangle$, such that $t = f'(t_1, \ldots, t_n)$,

$$\begin{array}{lll}
& size_{\{g\}}(\underline{tree}_1(f'(t_1, \ldots, t_n))) & \\
= & size_{\{g\}}(t_{f'}[y_i/\underline{tree}_1(t_i)\ ;\ i \in [n]]) & \text{(Definition of } \underline{tree}_1\text{)} \\
= & size_{\{g\}}(t_{f'}) + \sum_{i=1}^{n}(size_{\{g\}}(\underline{tree}_1(t_i))) & \text{(Def. of } size_{\{g\}},\ M \text{ nondel., noncop.)} \\
= & size_{\{g\}}(t_{f'}) + \sum_{i=1}^{n}(\sum_{f \in U}(size_{\{f\}}(t_i) \cdot size_{\{g\}}(t_f))) & \\
& & \text{(Induction Hypothesis)} \\
= & size_{\{g\}}(t_{f'}) + \sum_{f \in U}(\sum_{i=1}^{n}(size_{\{f\}}(t_i)) \cdot size_{\{g\}}(t_f)) & \\
& & \text{(Distributivity)} \\
= & \sum_{f \in U}(size_{\{f\}}(f'(t_1, \ldots, t_n)) \cdot size_{\{g\}}(t_f)) & \text{(Definition of } size_{\{f\}}\text{).}
\end{array}$$

We prove the Statements 2.(b) and 2.(c) as follows:

2.(b) For every $r \in I\!N$,

$$
\begin{array}{lll}
& size(\underline{tree}(r)) & \\
= & size_\Delta(\underline{tree}_2(\underline{tree}_1{}^r(\hat{t}))) & \text{(Definition of } \underline{tree}(r)\text{)} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(\hat{t}_f)). & \text{(Statement (3a))}
\end{array}
$$

2.(c) For every $r \in I\!N$,

$$
\begin{array}{lll}
& size(\underline{tree}(r+1)) & \\
= & size_\Delta(\underline{tree}_2(\underline{tree}_1{}^{(r+1)}(\hat{t}))) & \text{(Def. of } \underline{tree}(r+1)\text{)} \\
= & size_\Delta(\underline{tree}_1{}^{(r+1)}(\hat{t})) & \\
& + \sum_{g \in U}(size_{\{g\}}(\underline{tree}_1{}^{(r+1)}(\hat{t})) \cdot size_\Delta(\hat{t}_g)) & \text{(Statement (3a))} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) & \\
& + \sum_{g \in U}(size_{\{g\}}(\underline{tree}_1{}^{(r+1)}(\hat{t})) \cdot size_\Delta(\hat{t}_g)) & \text{(Statement (3b))} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) & \\
& + \sum_{g \in U}(\sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_{\{g\}}(t_f)) \cdot size_\Delta(\hat{t}_g)) & \text{(Statement (3c))} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) & \\
& + \sum_{g \in U}(\sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g))) & \text{(Distributivity)} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) & \\
& + \sum_{f \in U}(\sum_{g \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g))) & \text{(Summation)} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) & \\
& + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot \sum_{g \in U}(size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g))) & \text{(Distributivity)} \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) & \\
& + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot (size_\Delta(t_f) + \sum_{g \in U}(size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g)))). &
\end{array}
$$

Let 2.(e) be the statement

2.(e) For every $r \in I\!N$,

$$
\begin{array}{ll}
& size(\underline{tree}(r+1)) \\
= & size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(t_f)) \\
& + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^{(r+1)}(\hat{t})) \cdot size_\Delta(\hat{t}_f)).
\end{array}
$$

Actually, Statement 2.(e) has already been proved after the third step of the proof of Statement 2.(c). We shall need Statement 2.(e) later.

To show that for every $r \geq 0$ the condition $size(\underline{tree}(r+1)) > size(\underline{tree}(r))$ holds, we first have to prove the following Statements (4a) and (4b) which are estimations for the sizes of the output patterns:

(4a) For every $n \in I\!N$ and $f \in U^{(n)}$, $\quad size_\Delta(t_f) \geq |p_2|$.

(4b) For every $n \in I\!N$ and $f \in U^{(n)}$, $\quad size_\Delta(\hat{t}_f) \leq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k$.

The validity of these two statements can be shown as follows (for the definition of $\underline{trans}$ cf. Theorem 4.4):

(4a) For every $n \in \mathbb{N}$ and $f \in U^{(n)}$,

$$\begin{array}{cll} & size_\Delta(t_f) & \\ = & size_\Delta(\underline{trans}(nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, f(p_1, y_1, \ldots, y_n)))) & \text{(Definition of } t_f\text{)} \\ = & size_\Delta(nf(\Rightarrow_{e,P(e,p_1-p_1p_2)}, f(p_1, y_1, \ldots, y_n))) & (\Delta\text{–symbols unchanged)} \\ \geq & |p_2|, & \end{array}$$

since M is nondeleting and thus for every $p \in P_{p_2}$ with $P_{p_2} = \{p_1p' \mid p' \text{ is a proper prefix of } p_2\}$, there is an $f' \in F$, such that at least one $(f', label(e,p))$–rule has to be applied, each of them producing at least one output symbol. Note that $card(P_{p_2}) = |p_2|$.

(4b) For every $n \in \mathbb{N}$ and $f \in U^{(n)}$,

$$\begin{array}{cll} & size_\Delta(\hat{t}_f) & \\ = & size_\Delta(nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))) & \text{(Definition of } \hat{t}_f\text{)} \\ \leq & size(nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2, y_1, \ldots, y_n))) & \\ \leq & \sum_{k=0}^{height(nf(\Rightarrow_{e,P(e,p_1p_2)}, f(p_1p_2,y_1,\ldots,y_n)))-1} (maxrank_{M,\Delta})^k & \\ \leq & \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1} (maxrank_{M,\Delta})^k, & \end{array}$$

because of Observation 3.14 and since $p = p_1p_2p_3$ is a longest path in e, p_3 is a longest path in $e''' = subtree(e, p_1p_2)$ and thus $height(subtree(e, p_1p_2)) = |p_3| + 1$.

Additionally, we need the following Statement (5), which states the equality of the function–number of an input tree e_r and a path $p_1p_2^l$ with $0 \leq l \leq r$ on the one hand and the number of symbols of U in that tree which results from $\hat{t}$ by applying l times the function $\underline{tree}_1$:

(5) For every $r \geq 0$ and $0 \leq l \leq r$, $\quad size_U(\underline{tree}_1{}^l(\hat{t})) = funnum(e_r, p_1{p_2}^l)$.

$$\begin{array}{cll} & size_U(\underline{tree}_1{}^l(\hat{t})) & \\ = & size_U(\underline{tree}_1{}^l(\underline{trans}(nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) & \text{(Definition of } \hat{t}\text{)} \\ = & size_U(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,p_1-p_1{p_2}^l)}, nf(\Rightarrow_{e,P(e,\varepsilon-p_1)}, \tilde{\rho}[x_1/\varepsilon])))) & \\ & & \text{(Statement (2) from the proof of} \\ & & \text{Theorem 4.4)} \\ = & size_U(\underline{trans}(nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1{p_2}^l)}, \tilde{\rho}[x_1/\varepsilon]))) & (P(e,\varepsilon-p_1) = P(e_r,\varepsilon-p_1)) \\ = & |seq_{p_1p_2^l}(nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1{p_2}^l)}, \tilde{\rho}[x_1/\varepsilon]))| & (*) \\ = & funnum(e_r, p_1{p_2}^l) & \text{(Definition of } funnum\text{)} \end{array}$$

The equality $(*)$ holds, since every function occurrence of the form $(f, p_1{p_2}^l)$ in the normal form $nf(\Rightarrow_{e_r,P(e_r,\varepsilon-p_1{p_2}^l)}, \tilde{\rho}[x_1/\varepsilon])$ is transformed by $\underline{trans}$ into exactly one occurrence of the symbol f.

Now we can summarize the above results for the main Statement 2.(d) of our pumping lemma as follows. For every $r \in I\!N$,

$$\begin{array}{ll}
 & size(\underline{tree}(r+1)) - size(\underline{tree}(r)) \\
= & size_{\Delta}(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_{\Delta}(t_f)) \\
 & + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^{(r+1)}(\hat{t})) \cdot size_{\Delta}(\hat{t}_f)) \\
 & - size_{\Delta}(\underline{tree}_1{}^r(\hat{t})) - \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_{\Delta}(\hat{t}_f)) \quad \text{(Statements 2.(b), 2.(e))} \\
= & \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot (size_{\Delta}(t_f) - size_{\Delta}(\hat{t}_f))) \\
 & + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^{(r+1)}(\hat{t})) \cdot size_{\Delta}(\hat{t}_f)) \\
\geq & \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^r(\hat{t}))) \cdot (|p_2| - \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1}(maxrank_{M,\Delta})^k) \\
 & \qquad \text{(Statements (4a), (4b))} \\
 & + \sum_{f \in U}(size_{\{f\}}(\underline{tree}_1{}^{(r+1)}(\hat{t}))) \qquad (size_{\Delta}(\hat{t}_f) \geq 1) \\
= & size_U(\underline{tree}_1{}^r(\hat{t})) \cdot (|p_2| - \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1}(maxrank_{M,\Delta})^k) \\
 & + size_U(\underline{tree}_1{}^{(r+1)}(\hat{t})) \\
= & funnum(e_r, p_1 p_2{}^r) \cdot (|p_2| - \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1}(maxrank_{M,\Delta})^k) \\
 & + funnum(e_{r+1}, p_1 p_2{}^{r+1}) \qquad \text{(Statement (5))} \\
> & 0,
\end{array}$$

because

1. $funnum(e_r, p_1 p_2{}^r) > 0$ and $funnum(e_{r+1}, p_1 p_2{}^{r+1}) > 0$, because of Lemma 3.15 and
2. $|p_2| \geq \sum_{k=0}^{(maxheight_{M,R})^{|p_3|+2}-1}(maxrank_{M,\Delta})^k$ as consequence of Lemma 5.5. $\square$

6 Applications

In this section we apply our pumping lemma to monadic languages only. Therefore we first present a specialized version of our extended pumping lemma. Then we demonstrate in a first application, how the pumping lemma can be used together with arithmetic arguments. The second application uses structural properties of certain output languages and yields three hierarchies of macro tree transducers.

6.1 Pumping Lemma for Monadic Output Languages

If we restrict the extended pumping lemma in Theorem 5.8 to macro tree transducers with monadic output languages, we obtain the following version of the pumping lemma which is shown in Theorem 6.1. Because of Lemma 3.18 we restrict Theorem 6.1 to macro tree transducers which only have functions of rank 1 or 2. Recall from Observation 3.19 that the noncopying–condition is no further restriction for such macro tree transducers with monadic output languages. Therefore we do not mention this condition explicitely in Theorem 6.1.

Theorem 6.1 (Pumping Lemma for Monadic Output Languages)
Let $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ be a (nondeleting and producing) macro tree transducer with extended pumping index epi_M, $\Delta = \Delta^{(1)} \cup \Delta^{(0)}$, and $F = F^{(1)} \cup F^{(2)}$.

For every $t \in L_{out}(M)$ with $size(t) \geq epi_M$

- there exists a ranked alphabet $(U, rank_U)$ with $U \subseteq F^{(1)} \cup F^{(2)}$, $card(U) \geq 1$, $rank_U(f) = rank_F(f) - 1$ for every $f \in U$, and
  - either $U = U^{(1)}$,
  - or there is exactly one $f \in U^{(0)}$,
- if $U = U^{(1)}$, then
  - there exists an $E \in \Delta^{(0)}$ and a tree $\hat{t} \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{E\}\rangle$ with $size_\Delta(\hat{t}) \geq 1$,
  - for every $f \in U$, there exists a tree $t_f \in T\langle \Delta^{(1)} \cup U^{(1)}\rangle(\{y_1\})$ with $1 \leq size_\Delta(t_f) < epi_M$ and $size_U(t_f) < epi_M$,
  - for every $f \in U$, there exists a tree $\hat{t}_f \in T\langle \Delta^{(1)}\rangle(\{y_1\})$ with $1 \leq size_\Delta(\hat{t}_f) < epi_M$,
- if there is exactly one $f \in U^{(0)}$, then
  - there exists a tree $\hat{t} \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{f\}\rangle$ with $size_\Delta(\hat{t}) \geq 1$,
  - there exists a tree $t_f \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{f\}\rangle$ with $1 \leq size_\Delta(t_f) < epi_M$ and $size_U(t_f) < epi_M$,

- for every $f' \in U - \{f\}$, there exists a tree $t_{f'} \in T\langle \Delta^{(1)} \cup U^{(1)} \rangle(\{y_1\})$ with $1 \leq size_\Delta(t_{f'}) < epi_M$ and $size_U(t_{f'}) < epi_M$,
- there exists an $E \in \Delta^{(0)}$ and a tree $\hat{t}_f \in T\langle \Delta^{(1)} \cup \{E\} \rangle$ with $1 \leq size_\Delta(\hat{t}_f) < epi_M$,
- for every $f' \in U - \{f\}$, there exists a tree $\hat{t}_{f'} \in T\langle \Delta^{(1)} \rangle(\{y_1\})$ with $1 \leq size_\Delta(\hat{t}_{f'}) < epi_M$,

- for every $f \in U$, the symbol f occurs in $\hat{t}$,
- for every $f \in U$, there is an $f' \in U$ such that f occurs in $t_{f'}$,

such that

1. $t = \underline{tree}(1)$,
2. for every $r \geq 0$,

   (a) $\underline{tree}(r) \in L_{out}(M)$,

   (b) $size(\underline{tree}(r))$
   $= size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U} (size_{\{f\}}(\underline{tree}_1{}^r(\hat{t})) \cdot size_\Delta(\hat{t}_f))$,

   (c) $size(\underline{tree}(r+1))$
   $= size_\Delta(\underline{tree}_1{}^r(\hat{t})) + \sum_{f \in U} (size_{\{f\}}(\underline{tree}_1{}^r(\hat{t}))$
   $\cdot(size_\Delta(t_f) + \sum_{g \in U} (size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g))))$, and

   (d) $size(\underline{tree}(r+1)) > size(\underline{tree}(r))$,

where the functions $\underline{tree}$, $\underline{tree}_1$, and $\underline{tree}_2$ are defined as in Theorem 4.4.

Proof. We only have to prove the additional statements of this pumping lemma. The proof is based on the proofs of Theorem 4.4 and Theorem 5.8. Because of Observation 3.19 the macro tree transducer M is additionally noncopying, such that we can adopt the results of Theorem 5.8.

Since $F = F^{(1)} \cup F^{(2)}$ we have $funset(e, p_1) = funset(e, p_1 p_2) \subseteq F^{(1)} \cup F^{(2)}$ and thus $U = U^{(0)} \cup U^{(1)}$.

Because of $\hat{t} \in T\langle \Delta \cup U \rangle = T\langle \Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)} \cup U^{(0)} \rangle$, the tree $\hat{t}$ is monadic. Thus at most one $f \in U^{(0)}$ can occur in $\hat{t}$ and we have $card(U^{(0)}) \leq 1$, because every $f \in U$ has to occur in $\hat{t}$.

Then, either $U = U^{(1)}$, or there is exactly one $f \in U^{(0)}$.

If $U = U^{(1)}$, then:

- There exists an $E \in \Delta^{(0)}$ with $\hat{t} \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{E\} \rangle$, because $\hat{t} \in T\langle \Delta \cup U \rangle = T\langle \Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)} \rangle$.

- For every $f \in U$, the tree $t_f \in T\langle\Delta^{(1)} \cup U^{(1)}\rangle(\{y_1\})$, because $t_f \in T\langle\Delta \cup U\rangle(Y_{rank_U(f)}) = T\langle\Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)}\rangle(\{y_1\})$ and because y_1 has to occur in t_f (and thus no $E \in \Delta^{(0)}$).
- For every $f \in U$, the tree $\hat{t}_f \in T\langle\Delta^{(1)}\rangle(\{y_1\})$, because $\hat{t}_f \in T\langle\Delta\rangle(Y_{rank_U(f)}) = T\langle\Delta^{(1)} \cup \Delta^{(0)}\rangle(\{y_1\})$ and because y_1 has to occur in $\hat{t}_f$ (and thus no $E \in \Delta^{(0)}$).

If there is exactly one $f \in U^{(0)}$, then

- $\hat{t} \in T\langle\Delta^{(1)} \cup U^{(1)} \cup \{f\}\rangle$, because $\hat{t} \in T\langle\Delta \cup U\rangle = T\langle\Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)} \cup U^{(0)}\rangle$ and because f has to occur in $\hat{t}$ (and thus no $E \in \Delta^{(0)}$).
- For every $f' \in U - \{f\}$, the tree $t_{f'} \in T\langle\Delta^{(1)} \cup U^{(1)}\rangle(\{y_1\})$, because $t_{f'} \in T\langle\Delta \cup U\rangle(Y_{rank_U(f')}) = T\langle\Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)} \cup U^{(0)}\rangle(\{y_1\})$ and because y_1 has to occur in $t_{f'}$ (and thus no $E \in \Delta^{(0)}$ and not f).
- $t_f \in T\langle\Delta^{(1)} \cup U^{(1)} \cup \{f\}\rangle$, because $t_f \in T\langle\Delta \cup U\rangle(Y_{rank_U(f)}) = T\langle\Delta^{(1)} \cup \Delta^{(0)} \cup U^{(1)} \cup U^{(0)}\rangle$ and because f does not occur in $t_{f'}$ for every $f' \in U - \{f\}$, it has to occur in t_f (and thus no $E \in \Delta^{(0)}$).
- For every $f' \in U - \{f\}$, the tree $\hat{t}_{f'} \in T\langle\Delta^{(1)}\rangle(\{y_1\})$, because $\hat{t}_{f'} \in T\langle\Delta\rangle(Y_{rank_U(f')}) = T\langle\Delta^{(1)} \cup \Delta^{(0)}\rangle(\{y_1\})$ and because y_1 has to occur in $\hat{t}_{f'}$ (and thus no $E \in \Delta^{(0)}$).
- There exists an $E \in \Delta^{(0)}$ with $\hat{t}_f \in T\langle\Delta^{(1)} \cup \{E\}\rangle$, because $\hat{t}_f \in T\langle\Delta\rangle(Y_{rank_U(f)}) = T\langle\Delta^{(1)} \cup \Delta^{(0)}\rangle$.

Observe that we have left out the condition "for every $f \in U$ and for every $y \in Y_{rank_U(f)}$, the symbol y occurs exactly once in t_f and $\hat{t}_f$" from Theorem 5.8, because this condition is contained in the above requirements for the types of the output patterns. □

Note that not every condition for the output patterns in the pumping lemma is needed in every application. In the following arithmetic proof, for example, we do not use the knowledge about the structure of output patterns as exact as it was stated in Theorem 6.1. In this kind of application the size conditions of the output patterns are very useful. Clearly, the structural proofs in the next but one subsection depend especially on the structure of the output patterns.

6.2 Arithmetic Applications

It is known from Lemma 3.3 of [Eng81] and Lemma 3.24 of [EV85] that, if M is a macro tree transducer, then there is a constant $c > 0$ such that, if $\tau(M)(e) = t$ for an input tree e and an output tree t, then $height(t) \leq c^{height(e)}$ holds. Thus, for example, there does not exist a macro tree transducer, which calculates the tree transformation $\tau_1 : T\langle\{\sigma^{(2)}, \alpha^{(0)}\}\rangle \longrightarrow T\langle\{B^{(1)}, E^{(0)}\}\rangle$ with $\tau_1(e) = B^{2^{2 \cdot size(e)-1}}E$ for every $e \in T\langle\Sigma\rangle$, since $size(e)$ can grow exponentially in $height(e)$.

We also can deduce that there does not exist a macro tree transducer which calculates the tree transformation $\tau_2 : T\langle\{\gamma^{(1)}, \alpha^{(0)}\}\rangle \longrightarrow T\langle\{B^{(1)}, E^{(0)}\}\rangle$ with $\tau_2(\gamma^n(\alpha)) = B^{2^{2^n}} E$ for every $n \geq 0$.

We only mention here that there is a macro attributed tree transducer (cf. Theorem 4.2 of [KV94b]) which calculates τ_1 and that even no macro attributed tree transducer can realize τ_2.

Now the question arises, whether an arbitrary ranked alphabet Σ and a macro tree transducer M, which computes a tree transformation $\tau(M) : T\langle\Sigma\rangle \longrightarrow T\langle\{B^{(1)}, E^{(0)}\}\rangle$ with $L_{out}(M) = \{B^{2^{2^n}} E \mid n \geq 0\}$, exist. We answer this question here for our restricted macro tree transducers: We can use the pumping lemma for monadic output languages to prove that there is no producing and nondeleting macro tree transducer with output language $\{B^{2^{2^n}} E \mid n \geq 0\}$.

Theorem 6.2 $\{B^{2^{2^n}} E \mid n \geq 0\} \notin MT_{out}$

Proof. We assume that there is a producing and nondeleting macro tree transducer N with monadic output language $L_{out}(N) = \{B^{2^{2^n}} E \mid n \geq 0\}$. Then, by Lemma 3.18, there is also a producing and nondeleting macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ with $L_{out}(M) = L_{out}(N)$ and with $F = F^{(1)} \cup F^{(2)}$. By Theorem 6.1, for every $t \in L_{out}(M)$ with $size(t) \geq epi_M$ certain properties hold. We consider the output tree $t = B^{2^{2^{(epi_M{}^2+epi_M+1)}}} E$ with $size(t) = 1 + 2^{2^{(epi_M{}^2+epi_M+1)}} \geq epi_M$.

Now we study an arbitrary choice of a nonempty subset $U \subseteq F^{(1)} \cup F^{(2)}$, of a tree $\hat{t}$, and of trees t_f and $\hat{t}_f$ for every $f \in U$, fulfilling the conditions of Theorem 6.1, such that $t = \underline{tree}(1)$. For every such choice we can deduce the contradiction $\underline{tree}(0) \notin L_{out}(M)$.

With the help of Statement 2.(c) and the size conditions of Theorem 6.1 we can estimate $size(\underline{tree}(1)) = 2^{2^{(epi_M{}^2+epi_M+1)}} + 1$ as follows. Note that in this case $r = 0$ holds.

$$\begin{aligned}
& 1 + 2^{2^{(epi_M{}^2+epi_M+1)}} \\
= \;& size(\underline{tree}(1)) \\
= \;& size_\Delta(\hat{t}) + \textstyle\sum_{f\in U}(size_{\{f\}}(\hat{t}) \cdot (size_\Delta(t_f) + \sum_{g\in U}(size_{\{g\}}(t_f) \cdot size_\Delta(\hat{t}_g)))) \\
< \;& size_\Delta(\hat{t}) + \textstyle\sum_{f\in U}(size_{\{f\}}(\hat{t}) \cdot (epi_M + \sum_{g\in U}(size_{\{g\}}(t_f) \cdot epi_M))) \\
= \;& size_\Delta(\hat{t}) + \textstyle\sum_{f\in U}(size_{\{f\}}(\hat{t}) \cdot (epi_M + (epi_M \cdot size_U(t_f)))) \\
< \;& size_\Delta(\hat{t}) + \textstyle\sum_{f\in U}(size_{\{f\}}(\hat{t}) \cdot (epi_M + epi_M{}^2)) \\
= \;& size_\Delta(\hat{t}) + size_U(\hat{t}) \cdot (epi_M + epi_M{}^2).
\end{aligned}$$

Thus we have the intermediate result

$$size_U(\hat{t}) > \frac{1 + 2^{2^{(epi_M{}^2+epi_M+1)}} - size_\Delta(\hat{t})}{epi_M{}^2 + epi_M}.$$

With the help of Statement 2.(b), the size conditions of Theorem 6.1, and the above intermediate result we can estimate $size(\underline{tree}(0))$ as follows. Note again that $r = 0$ holds.

$$
\begin{array}{ll}
& size(\underline{tree}(0)) \\
= & size_\Delta(\hat{t}) + \sum_{f \in U}(size_{\{f\}}(\hat{t}) \cdot size_\Delta(\hat{t}_f)) \\
\geq & size_\Delta(\hat{t}) + \sum_{f \in U}(size_{\{f\}}(\hat{t}) \cdot 1) \\
= & size_\Delta(\hat{t}) + size_U(\hat{t}) \\
> & size_\Delta(\hat{t}) + \frac{1+2^{2^{(epi_M{}^2+epi_M+1)}} - size_\Delta(\hat{t})}{epi_M{}^2+epi_M} \qquad \text{(Intermediate Result)} \\
= & \frac{1+2^{2^{(epi_M{}^2+epi_M+1)}}}{epi_M{}^2+epi_M} + size_\Delta(\hat{t}) \cdot (1 - \frac{1}{epi_M{}^2+epi_M}) \\
\geq & 1 + \frac{2^{2^{(epi_M{}^2+epi_M+1)}}}{epi_M{}^2+epi_M} + \frac{1-1}{epi_M{}^2+epi_M} \qquad (size_\Delta(\hat{t}) \geq 1 \text{ and } 1 \geq \frac{1}{epi_M{}^2+epi_M}) \\
= & 1 + \frac{2^{(2^{(epi_M{}^2+epi_M)}+2^{(epi_M{}^2+epi_M)})}}{epi_M{}^2+epi_M} \\
= & 1 + 2^{2^{(epi_M{}^2+epi_M)}} \cdot \frac{2^{2^{(epi_M{}^2+epi_M)}}}{epi_M{}^2+epi_M} \\
> & 1 + 2^{2^{(epi_M{}^2+epi_M)}}. \qquad (2^{2^{(epi_M{}^2+epi_M)}} > epi_M{}^2 + epi_M)
\end{array}
$$

Thus $1 + 2^{2^{(epi_M{}^2+epi_M)}} < size(\underline{tree}(0)) < size(\underline{tree}(1)) = 1 + 2^{2^{(epi_M{}^2+epi_M+1)}}$ results from Statement 2.(d) of Theorem 6.1 and therefore $\underline{tree}(0) \notin L_{out}(M) = \{B^{2^{2^n}} E \mid n \geq 0\}$, contradicting Statement 2.(a) of the pumping lemma. $\square$

A similar proof which is left out here shows that also the following result holds:

Theorem 6.3 $\{B^{2^{n^2}} E \mid n \geq 0\} \notin MT_{out}$ $\square$

This theorem states that there are even monadic languages, in which, roughly speaking, the heights of the trees grow slower than double exponentially, and which are not the output language of a producing and nondeleting macro tree transducer. We also mention here that the result $\{B^{2^n} E \mid n \geq 0\} \in MT_{out}$ holds, since a similar macro tree transducer as in Example 4.3 of [EV85] can be constructed which computes the set of all monadic trees with exponential height as output and which additionally is producing.

6.3 Structural Applications and Hierarchies of Macro Tree Transducers

In this subsection we first define an infinite family $(L_{(r)} \mid r \geq 1)$ of monadic languages. For every $r \geq 1$ we show a producing and nondeleting macro tree transducer $M_{(r)}$ with $L_{out}(M_{(r)}) = L_{(r)}$ and with as few as possible functions. Then we apply our pumping lemma to prove for every $r \geq 1$ that $L_{(r)}$ cannot be an output language of a macro tree transducer which has, roughly speaking, fewer functions or functions with smaller rank than $M_{(r)}$. These results are used to present three hierarchies for macro tree transducers with bounds on the number of functions.

Definition 6.4 For every $r \geq 1$ we define the monadic ranked alphabet

$$\Delta_{(r)} = \{A_1^{(1)}, \ldots, A_r^{(1)}, E^{(0)}\}$$

and the monadic language

$$L_{(r)} = \{{A_1}^n \ldots {A_r}^n E \mid n \geq 1\} \subseteq T\langle\Delta_{(r)}\rangle. \qquad \Box$$

In [ERS80] the pumping lemma for top–down tree–to–string transducers is applied to languages which are related to our here defined languages by the correspondence between monadic languages and string languages. Nevertheless, our applications are more difficult to prove, because of the more complicate structure of our pumping lemma.

Lemma 6.5 For every $k \geq 1$,

- $L_{(2k)} \in MT_{(k*2),out}$ and
- $L_{(2k-1)} \in MT_{(1*1,(k-1)*2),out}$.

Proof. Let $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$.

For every $k \geq 1$ we define the macro tree transducers

- $M_{(2k)} = (F_{(2k)}, \Delta_{(2k)}, \Sigma, \tilde{\rho}_{(2k)}, R_{(2k)})$ with:

$$\begin{array}{lcllll} F_{(2k)} & = & \{f_1^{(2)}, \ldots, f_k^{(2)}\}, & & & \\ \tilde{\rho}_{(2k)} & = & f_1(x_1, f_2(x_1, \ldots f_k(x_1, E)\ldots)), \text{ and} & & & \\ R_{(2k)} & = & \{f_i(\gamma(x_1), y_1) & \to & A_{2i-1} f_i(x_1, A_{2i} y_1) & \mid i \in [k]\} \quad \cup \\ & & \{f_i(\alpha, y_1) & \to & A_{2i-1} A_{2i} y_1 & \mid i \in [k]\}, \qquad \text{and} \end{array}$$

- $M_{(2k-1)} = (F_{(2k-1)}, \Delta_{(2k-1)}, \Sigma, \tilde{\rho}_{(2k-1)}, R_{(2k-1)})$ with:

$$\begin{array}{lcllll} F_{(2k-1)} & = & \{f_1^{(2)}, \ldots, f_{k-1}^{(2)}, f_k^{(1)}\}, & & & \\ \tilde{\rho}_{(2k-1)} & = & f_1(x_1, f_2(x_1, \ldots f_{k-1}(x_1, f_k(x_1))\ldots)), \text{ and} & & & \\ R_{(2k-1)} & = & \{f_i(\gamma(x_1), y_1) & \to & A_{2i-1} f_i(x_1, A_{2i} y_1) & \mid i \in [k-1]\} \quad \cup \\ & & \{f_i(\alpha, y_1) & \to & A_{2i-1} A_{2i} y_1 & \mid i \in [k-1]\} \quad \cup \\ & & \{f_k(\gamma(x_1)) & \to & A_{2k-1} f_k(x_1), & \\ & & \;\; f_k(\alpha) & \to & A_{2k-1} E\}. & \end{array}$$

Clearly, for every $r \geq 1$, $L_{out}(M_{(r)}) = L_{(r)}$. Since, for every $k \geq 1$, $card(F_{(2k)}^{(2)}) = k$, $card(F_{(2k-1)}^{(2)}) = k - 1$ and $card(F_{(2k-1)}^{(1)}) = 1$, we can conclude the statement of the lemma. $\Box$

Now we apply the monadic pumping lemma to show that a macro tree transducer which has at most $i \geq 1$ functions of rank 1, at most $k \geq 0$ functions of rank 2, and no other functions cannot produce the output language $L_{(r)}$ for every $r > 2k + 1$. We exclude the case $i = 0$ from Lemma 6.6, because this case is handled by Lemma 6.8 which will deliver a slightly better result for this case.

Lemma 6.6 For every $i \geq 1$, $k \geq 0$, and $r > 2k+1$, $\qquad L_{(r)} \notin MT_{(i*1,k*2),out}$.

Proof. Let $i \geq 1$, $k \geq 0$, and $r > 2k+1$. We assume $L_{(r)} \in MT_{(i*1,k*2),out}$. Thus, there is a producing and nondeleting macro tree transducer $M = (F, \Delta, \Sigma, \tilde{\rho}, R)$ with $\Delta = \Delta_{(r)}$, with $F = F^{(1)} \cup F^{(2)}$, with $card(F^{(1)}) \leq i$, $card(F^{(2)}) \leq k$, and with $L_{out}(M) = L_{(r)}$. Then, by Theorem 6.1, for every $t \in L_{out}(M)$ with $size(t) \geq epi_M$ certain properties hold. We consider the output tree $t = A_1{}^{(epi_M{}^2+epi_M)} \ldots A_r{}^{(epi_M{}^2+epi_M)}E$ with $size(t) = 1 + r \cdot (epi_M{}^2 + epi_M) \geq epi_M$, since $r \geq 1$.

Now we study an arbitrary choice of a nonempty subset $U \subseteq F^{(1)} \cup F^{(2)}$, of a tree $\hat{t}$, and of trees t_f and $\hat{t}_f$ for every $f \in U$, fulfilling the conditions of Theorem 6.1, such that $t = \underline{tree}(1)$. For every such choice we can deduce the contradiction $\underline{tree}(0) \notin L_{out}(M)$.

Because of $card(F^{(2)}) \leq k$ in addition to the conditions of Theorem 6.1, either $card(U^{(0)}) = 0$ and there is a k' with $1 \leq k' \leq k$, such that $card(U^{(1)}) = k'$, or $card(U^{(0)}) = 1$ and there is a k' with $0 \leq k' \leq k$, such that $card(U^{(1)}) = k'$. Thus we have to analyse two cases. Note that if we have $k = 0$, then only the second case can occur.

<u>Case 1:</u> $card(U^{(0)}) = 0$ and there is a k' with $1 \leq k' \leq k$, such that $card(U^{(1)}) = k'$

We know $t = \underline{tree}(1) = \underline{tree}_2(\underline{tree}_1(\hat{t}))$ and $\underline{tree}(0) = \underline{tree}_2(\hat{t})$. Since $\hat{t} \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{E\}\rangle$ and since output symbols are left unchanged by $\underline{tree}_1$ and $\underline{tree}_2$, we can describe the construction of $\underline{tree}(0)$ out of $\underline{tree}(1)$ as follows: we substitute in $\underline{tree}(1)$ for every $f \in U^{(1)}$, roughly speaking, every pattern, which results from f by first applying the function $\underline{tree}_1$ and then $\underline{tree}_2$, by a pattern, which results from f by applying only the function $\underline{tree}_2$. This easy fact must be formalized in the following way, because every f has rank 1:

For every $f \in U^{(1)}$ and $t' \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{E\}\rangle$,

- $\underline{tree}_2(\underline{tree}_1(ft')) = \underline{tree}_2(t_f[y_1/\underline{tree}_1(t')]) = \underline{tree}_2(t_f)[\underline{tree}_2(y_1)/\underline{tree}_2(\underline{tree}_1(t'))]$ (the last equality can be shown by induction on $t_f \in T\langle \Delta^{(1)} \cup U^{(1)}\rangle(\{y_1\})$) and
- $\underline{tree}_2(ft') = \hat{t}_f[y_1/\underline{tree}_2(t')]$.

Thus, for the transformation from $\underline{tree}(1)$ into $\underline{tree}(0)$, for every $f \in U^{(1)}$, root–to–leaf sequences of the output symbols of $\underline{tree}_2(t_f)$ are substituted by root–to–leaf sequences of the output symbols of $\hat{t}_f$.

From the size conditions $size_\Delta(t_f) < epi_M$, $size_U(t_f) < epi_M$, and $size_\Delta(\hat{t}_f) < epi_M$ for every $f \in U^{(1)}$, we deduce for every $f \in U^{(1)}$,

$$\begin{array}{rcl} size_\Delta(\underline{tree}_2(t_f)) & = & size_\Delta(t_f) + \sum_{f' \in U}(size_{\{f'\}}(t_f) \cdot size_\Delta(\hat{t}_{f'})) \\ & < & size_\Delta(t_f) + size_U(t_f) \cdot epi_M \\ & < & epi_M{}^2 + epi_M. \end{array}$$

(Apart from y_1 in $t_f \in T\langle \Delta^{(1)} \cup U^{(1)}\rangle(\{y_1\})$ which is not counted by $size_\Delta$, the first equation is analogous to Statement (3a) in the proof of Theorem 5.8.)

Thus, on the one hand, for every $f \in U^{(1)}$, the pattern $\underline{tree}_2(t_f)$ can overlap at most two successive sequences of an equal output symbol A_j with $1 \leq j \leq r$, respectively, in $\underline{tree}(1)$. Because of $card(U^{(1)}) = k'$, at most $2k' \leq 2k$ sequences of equal output symbols in $\underline{tree}(1)$ can be overlapped by these patterns. Since there are $r > 2k + 1$ different sequences of equal output symbols in $\underline{tree}(1)$, at least two of them are completely parts of $\hat{t}$ and thus appear in $\underline{tree}(0)$ with unchanged length ${epi_M}^2 + epi_M$.

In Figure 10 we present an example situation for Case 1 with $k = k' = 2$, $U^{(1)} = \{f_1, f_2\}$, and $r = 6 > 5 = 2k + 1$. Here the two sequences $A_3 \ldots\ldots A_3$ and $A_6 \ldots\ldots A_6$ are completely parts of $\hat{t}$.

$$\underline{tree}(1) = A_1 \ldots\ldots A_1 A_2 \ldots\ldots A_2 A_3 \ldots\ldots A_3 A_4 \ldots\ldots A_4 A_5 \ldots\ldots A_5 A_6 \ldots\ldots A_6 E$$

Case 1: $\hat{t}$ $\quad \underline{tree}_2(t_{f_1})$ $\quad \hat{t}$ $\quad \underline{tree}_2(t_{f_2})$ $\quad \hat{t}$

Case 2: $\hat{t}$ $\quad \underline{tree}_2(t_{f_1})$ $\quad \hat{t}$ $\quad \underline{tree}_2(t_{f_2})$ $\quad \hat{t}$ $\quad \underline{tree}_2(t_f)$

Figure 10: Example situations for the Cases 1 and 2.

On the other hand we know that $size(\underline{tree}(0)) < size(\underline{tree}(1))$ and that $\underline{tree}(0)$ must contain the unique symbol E with rank 0. Therefore at least one j with $1 \leq j \leq r$ exists, such that $\underline{tree}(0)$ contains less than ${epi_M}^2 + epi_M$ occurrences of A_j.

Since there are j, j' with $1 \leq j, j' \leq r$, such that $\underline{tree}(0)$ contains different numbers of symbols A_j and $A_{j'}$, we have $\underline{tree}(0) \notin L_{out}(M) = L_{(r)}$, contradicting Statement 2.(a) of the pumping lemma.

Case 2: $card(U^{(0)}) = 1$ and there is a k' with $0 \leq k' \leq k$, such that $card(U^{(1)}) = k'$

The argumentation concerning the symbols in $U^{(1)}$ does not differ from Case 1. Here additionally we have to deal with the unique symbol $f \in U^{(0)}$:

Since y_1 does not occur in $t_f \in T\langle \Delta^{(1)} \cup U^{(1)} \cup \{f\}\rangle$, we have

- $\underline{tree}_2(\underline{tree}_1(f)) = \underline{tree}_2(t_f)$ and
- $\underline{tree}_2(f) = \hat{t}_f$.

Thus, for the transformation from $\underline{tree}(1)$ into $\underline{tree}(0)$, also $\underline{tree}_2(t_f)$ is substituted by $\hat{t}_f$. Since the occurrence of f in t_f is translated by $\underline{tree}_2$ into $\hat{t}_f \in T\langle \Delta^{(1)} \cup \{E\}\rangle$, we also know $\underline{tree}_2(t_f) \in T\langle \Delta^{(1)} \cup \{E\}\rangle$.

Thus, $\underline{tree}_2(t_f)$ contains the symbol E, and since $size_\Delta(\underline{tree}_2(t_f)) < {epi_M}^2 + epi_M$ with an analogous argumentation as in Case 1, the pattern $\underline{tree}_2(t_f)$ overlaps only the last sequence of the output symbol A_r in $\underline{tree}(1)$. Like in Case 1, for every $f' \in U^{(1)}$, the pattern $\underline{tree}_2(t_{f'})$ can overlap at most two successive sequences of an equal output symbol A_j with $1 \leq j \leq r$, respectively, in $\underline{tree}(1)$. Because of $card(U^{(0)}) = 1$ and $card(U^{(1)}) = k'$, at most $2k' + 1 \leq 2k + 1$ sequences of equal output symbols in $\underline{tree}(1)$ can be overlapped by these patterns. Since there are $r > 2k + 1$ different sequences of equal output symbols in $\underline{tree}(1)$, at least one of them is completely a part of $\hat{t}$ and thus appears in $\underline{tree}(0)$ with unchanged length ${epi_M}^2 + epi_M$.

Figure 10 also shows an example for Case 2 with $k = k' = 2$, $U^{(1)} = \{f_1, f_2\}$, $U^{(0)} = \{f\}$ and $r = 6 > 5 = 2k + 1$. Here only the sequence $A_3 \ldots\ldots A_3$ is completely a part of $\hat{t}$.

We can deduce $\underline{tree}(0) \notin L_{out}(M) = L_{(r)}$ in analogy to Case 1, contradicting again Statement 2.(a) of the pumping lemma. □

From Lemma 6.5 and Lemma 6.6 we gain the following Theorem 6.7 that presents for every $i \geq 0$ a hierarchy for classes of tree transformations of macro tree transducers with at most i functions of rank 1. In the following this hierarchy is called Hierarchy 1. Figure 11 shows Hierarchy 1 for an arbitrary $i \geq 0$.

Theorem 6.7 For every $i, k \geq 0$, $\quad MT_{(i*1,k*2)} \subset MT_{(i*1,(k+1)*2)}$.

Proof. For every $i \geq 0$ and $k \geq 0$, $MT_{(i*1,k*2)} \subseteq MT_{(i*1,(k+1)*2)}$ holds, because of Definition 3.3 and the notations of tree transformation classes.

For every $i \geq 1$ and $k \geq 0$, by Lemma 6.5, $L_{(2k+2)} \in MT_{((k+1)*2),out}$, and by Lemma 6.6, $L_{(2k+2)} \notin MT_{(i*1,k*2),out}$. Thus, since $MT_{((k+1)*2)} \subseteq MT_{(i*1,(k+1)*2)}$ for every $i \geq 0$, there is a tree transformation τ with ($L_{out}(\tau) = L_{(2k+2)}$ and) $\tau \in MT_{(i*1,(k+1)*2)} - MT_{(i*1,k*2)}$.

For $i = 0$ and for every $k \geq 1$, by Lemma 6.5, $L_{(2k+2)} \in MT_{((k+1)*2),out}$, and by Lemma 6.6, $L_{(2k+2)} \notin MT_{(1*1,k*2),out}$. Thus, since $MT_{(k*2)} \subseteq MT_{(1*1,k*2)}$, there is a tree transformation τ with ($L_{out}(\tau) = L_{(2k+2)}$ and) $\tau \in MT_{((k+1)*2)} - MT_{(k*2)}$.

If $i = k = 0$, then we can argue with Observation 3.13. □

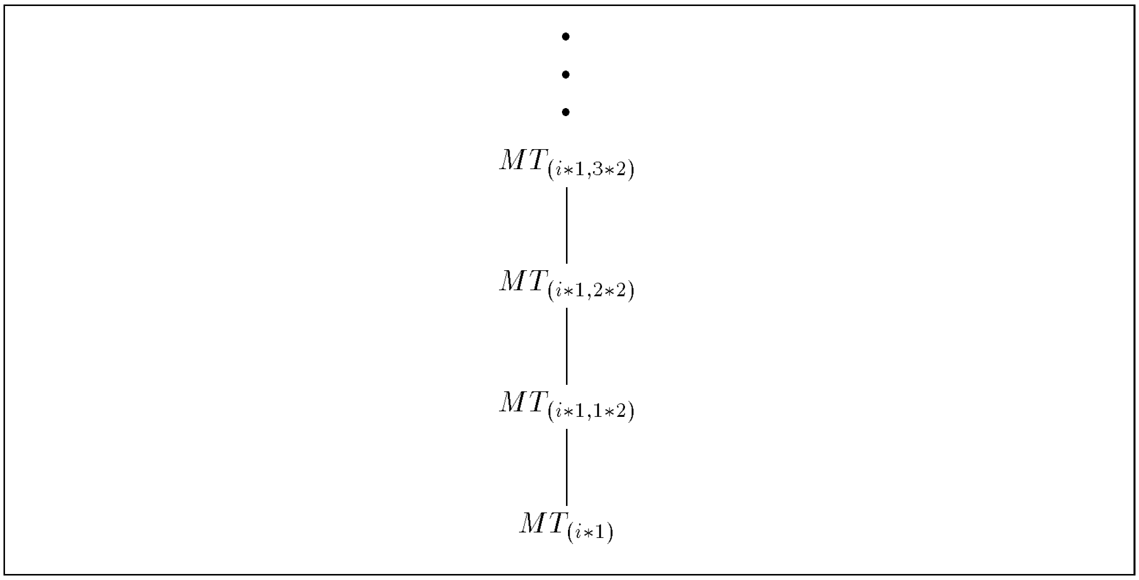

Figure 11: Hierarchy 1 of tree transformation classes for $i \geq 0$.

Hierarchy 1 only relates classes of tree transformations of macro tree transducers with fixed bound on the number of functions with rank 1. Now the question arises, whether there are also inclusions between some classes with different bounds on the number of unary functions or whether they are incomparable. Is it even possible to find a (complete) inclusion diagram for all tree transformation classes $MT_{(i*1,k*2)}$ witk $i, k \geq 0$? The answer here is only a restricted yes, because in the scope of this paper we only can prove an inclusion diagram for all tree transformation classes $MT_{(i*1,k*2)}$ with $0 \leq i \leq 1$ and $k \geq 0$. The reason for this restriction can be found in a certain weakness of our monadic pumping lemma, which, roughly speaking, cannot distinguish macro tree transducers that differ only in their number of unary functions.

In order to prove this inclusion diagram, which we simply call Hierarchy 2 in the following, we prove another strict inclusion and afterwards an incomparability between tree transformation classes. First we have to apply the monadic pumping lemma again to get the following result: a macro tree transducer which has at most $k \geq 1$ functions of rank 2 and no other functions cannot produce the output language $L_{(r)}$ for every $r > 2k$. We exclude the case $k = 0$ from Lemma 6.8, because this case is already contained in Observation 3.13 and cannot be proved by an application of the pumping lemma.

Lemma 6.8 For every $k \geq 1$ and $r > 2k$, $\qquad L_{(r)} \notin MT_{(k*2),out}$.

Proof. Since only functions of rank 2 are allowed here, this proof corresponds to Case 1 of the proof of Lemma 6.6 apart from the fact that here at least one (instead of at least two) of the sequences of equal output symbols is completely a part of $\hat{t}$. □

With the help of Lemma 6.8 we can deduce the strict inclusion which is presented in the following Theorem 6.9:

Theorem 6.9 For every $k \geq 0$, $\qquad MT_{(k*2)} \subset MT_{(1*1,k*2)}$.

Proof. For every $k \geq 0$, $MT_{(k*2)} \subseteq MT_{(1*1,k*2)}$ holds, because of Definition 3.3 and the notations of tree transformation classes.

For every $k \geq 1$, by Lemma 6.5, $L_{(2k+1)} \in MT_{(1*1,k*2),out}$, and by Lemma 6.8, $L_{(2k+1)} \notin MT_{(k*2),out}$. Thus, there is a tree transformation τ with ($L_{out}(\tau) = L_{(2k+1)}$ and) $\tau \in MT_{(1*1,k*2)} - MT_{(k*2)}$.

If $k = 0$, then we can argue with Observation 3.13. □

Now we show, for every $j \geq i \geq 1$, the incomparability between the classes $MT_{(j*2)}$ and $MT_{(1*1,(i-1)*2)}$. Therefore we present Theorem 6.10, which is a consequence of already proved results, and Theorem 6.11, which we shall prove with the help of another example output language $\bar{L}$.

Theorem 6.10 For every $j \geq i \geq 1$, $\qquad MT_{(j*2)} - MT_{(1*1,(i-1)*2)} \neq \emptyset$.

Proof. For every $j \geq i \geq 1$, by Lemma 6.5, $L_{(2i)} \in MT_{(i*2),out}$, and by Lemma 6.6, $L_{(2i)} \notin MT_{(1*1,(i-1)*2),out}$. Thus, since $MT_{(i*2)} \subseteq MT_{(j*2)}$, there is a tree transformation τ with ($L_{out}(\tau) = L_{(2i)}$ and) $\tau \in MT_{(j*2)} - MT_{(1*1,(i-1)*2)}$. □

Theorem 6.11 For every $j, i \geq 1$, $\quad MT_{(1*1,(i-1)*2)} - MT_{(j*2)} \neq \emptyset$.

Proof. Let $\bar{\Delta} = \{E_1{}^{(0)}, E_2{}^{(0)}\}$ be a monadic ranked alphabet and $\bar{L} = \{E_1, E_2\} = T\langle\bar{\Delta}\rangle$ be a monadic language.

We define the macro tree transducer $\bar{M} = (\bar{F}, \bar{\Delta}, \bar{\Sigma}, \bar{\rho}, \bar{R})$ with $\bar{F} = \{f^{(1)}\}$, $\bar{\Sigma} = \{\alpha^{(0)}, \beta^{(0)}\}$, $\bar{\rho} = f(x_1)$, and $\bar{R} = \{f(\alpha) \to E_1, f(\beta) \to E_2\}$. Clearly, $L_{out}(\bar{M}) = \bar{L}$, and thus $\bar{L} \in MT_{(1*1),out}$.

But, for every $j \geq 1$, we have $\bar{L} \notin MT_{(j*2),out}$, because for every macro tree transducer $M = (F, \bar{\Delta}, \Sigma, \tilde{\rho}, R)$ with $F = F^{(2)}$, exactly one output symbol with rank 0 has to occur in $\tilde{\rho} \in RHS(F^{(2)}, \bar{\Delta}, \{x_1\}, \emptyset)$. Since M is nondeleting, this symbol occurs in every output tree of M. Thus, $L_{out}(M) \neq \bar{L}$.

Thus, since $MT_{(1*1)} \subseteq MT_{(1*1,(i-1)*2)}$ for every $i \geq 1$, for every $j, i \geq 1$ there is a tree transformation τ with ($L_{out}(\tau) = \bar{L}$ and) $\tau \in MT_{(1*1,(i-1)*2)} - MT_{(j*2)}$. □

Note that Theorem 6.11 even holds for $j, i \geq 1$ and not only for $j \geq i \geq 1$, which would be required for our incomparability.

From Theorem 6.7, Theorem 6.9, Theorem 6.10, and Theorem 6.11 we can deduce Hierarchy 2, which is shown in Figure 12.

Note that the result of Theorem 6.11 heavily depends on the nondeleting–condition. We conjecture that in the case, where deleting of parameters is allowed, for every $k \geq 0$ a result like "$MT_{(1*1,k*2)} \subseteq MT_{((k+1)*2)}$" holds, because the unary function can be simulated by a binary function deleting its dummy argument at the end. Thus, instead of Hierarchy 2 a "zigzag"–hierarchy could be proved.

As conclusion of this section we establish in the following Theorem 6.12 a third hierarchy of tree transformation classes, in which the total number of functions of macro tree transducers is considered and which can be shown with the help of already proved results. We present this Hierarchy 3 in Figure 13.

Theorem 6.12 For every $k \geq 0$, $\quad MT_{(k)} \subset MT_{(k+1)}$.

Proof. For every $k \geq 0$, $MT_{(k)} \subseteq MT_{(k+1)}$ holds, because of Definition 3.3 and the notations of tree transformation classes.

By Lemma 6.5, for every $k \geq 1$, $L_{(2k+1)} \in MT_{(1*1,k*2),out}$ and thus $L_{(2k+1)} \in MT_{(k+1),out}$.

By Lemma 6.8, for every $k \geq 1$, $L_{(2k+1)} \notin MT_{(k*2),out}$. By Lemma 6.6, for every $k \geq 1$ and $1 \leq i \leq k$, $L_{(2k+1)} \notin MT_{(i*1,(k-i)*2),out}$, since $2k+1 > 2 \cdot (k-i) + 1$. Then we have $L_{(2k+1)} \notin \bigcup_{0 \leq i \leq k} MT_{(i*1,(k-i)*2),out}$. Because of Lemma 3.18 we know that a macro tree transducer may only use functions of rank 1 or 2 to produce $L_{(2k+1)}$. Thus we can conclude $L_{(2k+1)} \notin MT_{(k),out}$.

Thus, there is a tree transformation τ with ($L_{out}(\tau) = L_{(2k+1)}$ and) $\tau \in MT_{(k+1)} - MT_{(k)}$.

If $k = 0$, then we can argue with Observation 3.13. □

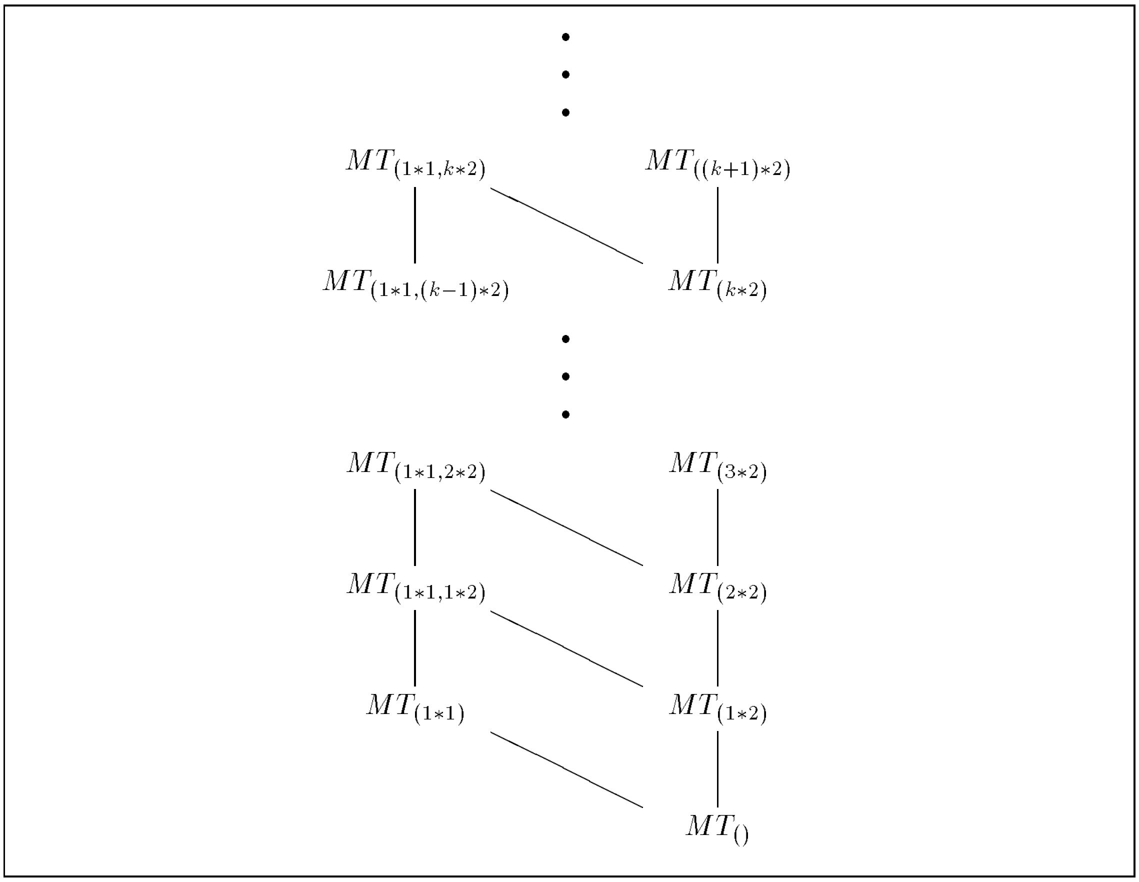

Figure 12: Hierarchy 2 of tree transformation classes.

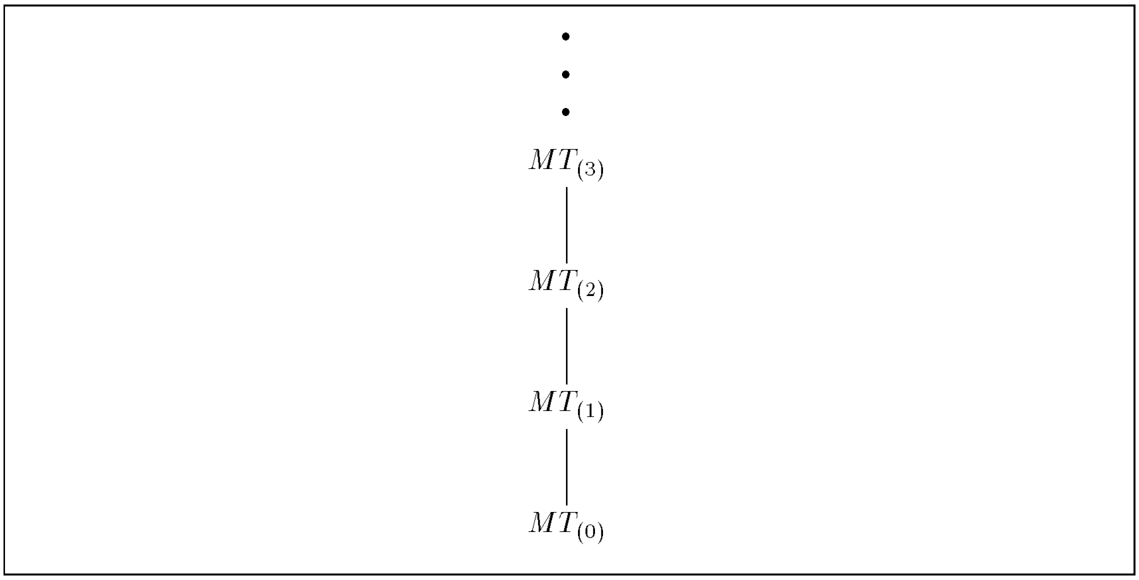

Figure 13: Hierarchy 3 of tree transformation classes.

7 Summary and Further Research Topics

This paper has introduced a pumping lemma for output languages of producing and nondeleting macro tree transducers. We have seen that the pumping process yields output trees which satisfy the growth–property, if we use the additional noncopying–restriction for macro tree transducers. We have applied the pumping lemma to monadic languages and have obtained the following results:

- The language $\{B^{2^{2^n}}E \mid n \geq 0\}$ which contains the trees (over the monadic ranked alphabet $\{B^{(1)}, E^{(0)}\}$) with double exponential height is no output language of a producing and nondeleting macro tree transducer.
- There are the following three hierarchies for tree transformations of producing and nondeleting macro tree transducers:
  - Macro tree transducers form a hierarchy with respect to their total number of functions.
  - They also induce a hierarchy with respect to their number of functions with rank 2, if the maximum number of functions with rank 1 is fixed and if there are no functions with rank 3 or greater.
  - The two hierarchies of the last item for at most one function with rank 1 and for no function with rank 1, respectively, can be combined to one hierarchy. The most important additional foundation for this hierarchy is the fact that, roughly speaking, the addition of one function with rank 1 to a macro tree transducer with only binary functions makes it more powerful.

This paper leaves several further research topics for pumping lemmata for macro tree transducers:

- We repeat a question that has already occured in connection with the pumping lemma for attributed tree transducers in [KV94a]: Are there non–monadic languages which can be proved not to be output languages of macro tree transducers with the help of our pumping lemma in a justifiable expense? In the case of non–monadic languages the proofs become very much harder, because the output patterns can no more be treated like concatenated strings as for example in the proof of Lemma 6.6. The output patterns are non–monadic trees which occur in a non–monadic output tree. The main problem is to find a complete case analysis for all possibilities to construct an output tree with output patterns. Then we have to derive a contradiction for every case.
- We have used the nondeleting–condition for the proof of the pumping lemma itself, the producing–condition for applications, and the noncopying–condition to prove the growth–property. Are there possibilities to relax these restrictions and are the perhaps resulting pumping lemmata are suitable for certain applications?
- Is it possible to use a pumping lemma for (general) macro tree transducers in a proof that the difference set SI_fT–S_fT of subclasses of macro attributed tree transducers

is not empty, as it was conjectured in [KV94b]? Up to now we only have a tree transformation in the class SI_fT from which we can show with the help of our pumping lemma that it cannot be realized with a producing and nondeleting macro tree transducer with only one function. We did not present the proof in this paper, because its complexity is out of all proportion to the result.

- Finally, a pumping lemma for macro attributed tree transducers (cf. [KV94b]) can be combined from the pumping lemmata for attributed tree transducers and macro tree transducers. Then we also would have a pumping lemma for the class SI_fT and perhaps it is possible to prove that the difference set S_fT–SI_fT is not empty, as it also was conjectured in [KV94b].

Acknowledgement

I would like to thank Heiko Vogler for inspiring me to investigate pumping lemmata for macro tree transducers, for carefully reading the preceding version of this paper, and for making useful suggestions to improve the understandability of the text.

References

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

[BPS61] Y. Bar–Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Z. Phonetik. Sprach. Komm.*, 14:143–172, 1961.

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

[Eng75] J. Engelfriet. Bottom–up and top–down tree transformations — a comparison. *Math. Syst. Theory*, 9:198–231, 1975.

[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.

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

[Ési80] Z. Ésik. Decidability results concerning tree transducers. *Acta Cybernetica*, 5:1–20, 1980.

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

[EV91] J. Engelfriet and H. Vogler. Modular tree transducers. *Theoret. Comp. Sci.*, 78:267–304, 1991.

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

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

[KV94a] A. Kühnemann and H. Vogler. A pumping lemma for output languages of attributed tree transducers. *Acta Cybernetica*, 11:261–305, 1994.

[KV94b] A. Kühnemann and H. Vogler. Synthesized and inherited functions — a new computational model for syntax–directed semantics. *Acta Informatica*, 31:431–477, 1994.

[Per76] C.R. Perrault. Intercalation lemmas for tree transducer languages. *J. Comput. Syst. Sci.*, 13:246–277, 1976.

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

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