PHD THESIS

Pebble Macro Tree Transducers with Strong Pebble Handling

Loránd Muzamel

Department of Foundations of Computer Science
University of Szeged
Árpád tér 2., H-6720 Szeged, Hungary
`muzamel@inf.u-szeged.hu`

Supervisor: **Professor Zoltán Fülöp**

Ph.D. School in Computer Science

May 29, 2010

Contents

1 Introduction

Tree translations play an important role, among others, in the specification of the syntax-directed semantics of a programming language [Iro61, Knu68, Knu71, WM95], in functional programs working on tree structured data [Vog91], and in the specification and implementation of XML transformations [MN01, BMN02, MBPS05a], and XML query languages [Via01].

Tree transducers are computation models for studying the abstract properties of the different tree translations which exist in practice. For instance, the macro tree transducer [Eng80, Eng81, CF82, EV85] is a model for syntax-directed translations, the attributed tree transducer [Fül81, FV98] is a model for the translations realized by attribute grammars [Knu68, Knu71], and the n-pebble tree transducer [MSV03] is a model for XML query languages and transformations. The pebble macro tree transducer of [EM03] is a combination of the pebble tree transducer and the macro tree transducer, hence it is a model which is suitable for studying the relationship between pebble tree transducers and macro tree transducers. Each tree transducer computes a tree transformation, which is a binary relation over abstract trees, i.e., trees over ranked alphabets.

In this thesis we consider pebble macro tree transducers of [EM03] and equip them with "strong pebbles" (see [EH05, MSS06]). Strong pebble handling means that the last dropped pebble can be lifted regardless of the position of the reading head. Hence we generalize the original model because it allows to lift the last pebble only if the reading head is on its position. In spite of the generalization, we leave the name pebble macro tree transducer unchanged in our thesis and refer to the original definition of [EM03] as a pebble macro tree transducer with "weak pebbles".

An n-pebble macro tree transducer M is a finite-state machine that translates input trees to output trees. Let us take an input tree s to M. M has a reading head, which is a pointer u to a node of s, and can move this pointer up and down along the edges of s. The n pebbles, denoted by $1, \ldots, n$, can be dropped at and lifted from the nodes of s in a stack-like fashion: if $0 \leq l \leq n$ pebbles are on s and the symbol of node u is scanned, then the following can be made.

- Pebble $(l + 1)$ can be dropped at u (provided $l < n$).
- Pebble l can be lifted (provided $l \geq 1$) regardless of its position (strong pebble handling).

In contrast, weak pebble handling of the original definition allows to lift pebble l only if it is on the current node u, cf. [EM03].

M makes a computation over sentential forms which are trees over the output symbols and so called configurations. The computation starts with the initial configuration, i.e., a tree consisting of a single node, which contains the information that M is in the initial state with the reading head at the root of s and no pebbles on s. A step of the computation is made in the way that an outside-active configuration node $\langle q, h\rangle$ of the current sentential form ξ is extended. (The concept outside-active means that no ancestor of $\langle q, h\rangle$ in ξ is a configuration, hence we define an OI-semantics for M.) Here q is the current state and $h = (u, \pi)$, where u is the current node and $\pi = [u_1; \ldots ; u_l]$, $l \leq n$ is a vector containing the nodes of s which hold pebbles $1, \ldots , l$. M can test the label of the current node u, its "child position", (i.e., whether u is the root of s or the jth child of its parent), and the presence of pebbles at node u. Depending on the test, M generates a tree which may contain further configurations and is replaced for $\langle q, h\rangle$ in a second-order fashion (meaning that a node, anywhere inside the sentential form, is replaced). Pebble tree transducers, in contrast, have their configurations only at leaf nodes of the sentential form. In this way M computes the next sentential form ξ'. If a sentential form t is computed which contains no configurations, then it is called an output tree for s and it is said that M translates s into t. There may be computations which never terminate because they get into an infinite cycle. In this case no output tree is computed. We call this phenomenon circularity.

The problem whether n-pebble macro tree transducers are more powerful than n-pebble macro tree transducers with weak pebble handling (i.e., than n-pebble macro tree transducers of [EM03]) is still open. We are only able to demonstrate the advantage of the use of strong pebbles: in Example 3.6 we give a 1-pebble tree transducer such that the equivalent pebble tree transducers with weak pebble, at least those which we know, have more states and rules.

In the following we give a short summary of the topics which we consider in the thesis.

We define three concepts of circularity: weak circularity, circularity and strong circularity. The hierarchy of the three concepts, not surprisingly, is that strong circularity implies circularity, which implies weak circularity (Lemma 4.3). Hence, the most natural concept is the strong circularity because the "smallest" condition for a pebble macro tree transducer M to guarantee that no computation of M gets into a cycle is that M is not strongly circular. Yet, we use also the other two circularity concepts because we could only prove some of our results by assuming that a pebble tree transducer is noncircular (e.g., in Lemma 8.4) or a pebble macro tree transducer is not weakly circular (e.g., the main results in Section 8.3).

We also consider the composition and the decomposition of tree transformations computed by n-pebble macro tree transducers. In general, in the composition theory of tree transformations we consider if the composition of two (or more) tree transformations computed by some tree transducers can be computed by a single tree transducer. The composition appears in applications in a natural way: in multi-pass compilers, as a model for deforestation in functional languages [Küh98, Voi02], and as implementations of queries to a (possibly iterated) view of an XML database. The decomposition of a tree transformation computed by a tree transducer means to consider if the tree transformation appears as the composition of (mainly two) tree transformations computed by "simpler" machines. The decomposition may help to understand the work of the original machine.

We give an example of both a composition and a decomposition. It was shown in [EV85], see also [FV98], that the composition of a total and deterministic top-down tree transformation [Eng75, Rou70] and a yield transformation can be computed by a total and deterministic macro tree transducer (a composition result) and vice versa, that each tree transformation computed by a deterministic and total macro tree transducer is equal to the composition of a deterministic and total top-down tree transformation and a yield transformation (a decomposition result). By putting the above composition and decomposition result together, we obtain the characterization $dtMAC = dtTOP \circ dtYIELD$ of the class of tree transformations computed by deterministic and total macro tree transformations, where the notations should be clear from the context. Let us mention that this result leads to show that the composition closure of deterministic and total macro tree transformations and of attributed tree transformations [Fül81, FV98] coincide, see Chapter 6 of [FV98].

Since pebble macro tree transducers are somewhat similar to macro tree transducers, we consider if "yield-like" composition and decomposition results can be obtained for them like the above one. (The fact that for macro attributed tree transformations such a composition and a decomposition result exists [KV94], see also Theorem 7.29 of [FV98], just confirms us to believe that we can find some for pebble macro tree transducers as well.)

First, we prove a yield-like composition result for pebble macro tree transducers. Namely, for every n-pebble tree transducer M and yield tree transformation $yield_g$ (were g is a mapping from leaves to sets of trees), we construct an n-pebble macro tree transducer M' such that $\tau_{M'} = \tau_M \circ yield_g$ holds. If M and $yield_g$ are deterministic (total), then also M' is deterministic (total). Hereby, we obtain the composition result $n\text{-}PTT \circ YIELD \subseteq n\text{-}PMTT$, where $n\text{-}PMTT$ and $n\text{-}PTT$ are the classes of tree transformations computed by n-pebble macro and n-pebble tree transducers, respectively, and $YIELD$ is the class of (nondeterministic) yield tree transformations (Lemma 6.1). This construction is a generalization of the one appearing in the recalled composition of deterministic and total top-down tree transformations and deterministic and total yield tree transformations.

Then we decompose pebble macro tree transformations. Namely, for every pebble macro tree transducer M, we effectively give a pebble tree transducer M' and a yield tree transformation $yield_g$ such that $\tau_M = \tau_{M'} \circ yield_g$ (Lemma 7.7). Hence we get the decomposition result $n\text{-}PMTT \subseteq n\text{-}PTT \circ YIELD$ (Corollary 7.9). Combining this decomposition with the composition result $n\text{-}PTT \circ YIELD \subseteq n\text{-}PMTT$ of Lemma 6.1, we obtain our first main result, i.e., the characterization $n\text{-}PMTT = n\text{-}PTT \circ YIELD$ (Theorem 7.10).

We note that in the proof of Lemma 7.7 we more or less follow the classical technique applied for the decomposition of total and deterministic macro tree transformations in [EV85, FV98]. However, our construction also works for nondeterministic and for circular (i.e, not terminating) pebble macro tree transducers. Unfortunately, it is a weakness of our construction that the pebble tree transducer M' is strongly circular and not deterministic (even if M is not circular and deterministic). We discuss this problem in Section 7.3. Therefore, we also consider if there is another decomposition technique, which preserves determinism and noncircularity (at least for a restricted class of pebble macro tree transducers). It turns out that the answer is positive. In Definition 8.1 we provide an alternative construction method for M' and $yield_g$ which works for some special pebble macro tree transducers. We show in Lemmas 8.4 and

8.6 that, if M is deterministic (or context-linear) and M' is noncircular, then indeed $\tau_M = \tau_{M'} \circ yield_g$ holds. Next we examine whether a reasonable syntactic restriction on M can be made to guarantee that M' is noncircular. A trivial restriction is that M is a macro tree transducer: in this case M' will be a top-down tree transducer, see [EV85], which cannot be circular, hence Lemmas 8.4 and 8.6 work and the decomposition results stated in Corollaries 8.7 and 8.8 hold. Another natural restriction would be that M is noncircular, however it is not sufficient because there is a noncircular pebble macro tree transducer M such that the pebble tree transducer M' is circular, see Example 8.2. An appropriate restriction is that M is not weakly circular. We show that, if M is not weakly circular in the construction mentioned above, then M' is noncircular. Hence, for every not weakly circular and deterministic (or context-linear) M, the decomposition equation $\tau_M = \tau_{M'} \circ yield_g$ holds (Corollary 8.10). Since every partial yield tree transformation can be computed by a noncircular and deterministic 0-pebble tree transducer (Theorem 5.2), we obtain another main result of the thesis: each not weakly circular and deterministic (resp. context-linear) n-pebble macro tree transformation is the composition of a noncircular and total and deterministic (resp. nondeterministic) n-pebble tree transformation and a noncircular and deterministic 0-pebble tree transformation (Theorem 8.11).

The next topic we consider in the thesis is the solution of an open problem raised in Section 8 of [EM03]. Namely, we prove in Theorem 9.6 that each n-pebble tree transducer M (provided $n \geq 1$) can be simulated by an $(n-1)$-pebble macro tree transducer M', i.e., that $n\text{-}PTT \subseteq (n-1)\text{-}PMTT$. The idea behind the construction is that we can replace the power of pebble n (the last pebble) of the pebble tree transducer by macro calls.

There are some important consequences of Theorems 7.10 and 9.6 (Theorem 10.3). The most interesting ones are the decomposition results $n\text{-}PTT \subseteq 0\text{-}PTT^{n+1}$ and $n\text{-}PTT \subseteq sMTT^{n+1}$ for every $n \geq 1$ and their macro versions $n\text{-}PMTT \subseteq 0\text{-}PTT^{n+2}$ and $n\text{-}PMTT \subseteq sMTT^{n+2}$ for every $n \geq 0$, where $sMTT$ is the class of *stay-macro tree transformations* of [EM03]. These decompositions were obtained in Theorems 10 and 35, and Section 8 of [EM03] for the weak pebble handling case. However, we think that those proofs cannot be generalized for the strong pebble case because the mapping EncPeb appearing in the proof of Theorem 10 of [EM03] is strongly based on weak pebble handling.

Then we obtain the following applications of the above decomposition results.

In Lemma 27 of [EM03], it was proved that $sMTT \subseteq MON \circ MTT$, where MON is the class of monadic insertions and MTT is the class of macro tree transformations. Moreover, both the inverses of *monadic insertions* and of *macro tree transformations* preserve regularity of tree languages. These obviously imply that the inverses of stay-macro tree transformations also preserve regularity. Hence, by $n\text{-}PMTT \subseteq sMTT^{n+2}$, the inverses of (compositions of) pebble macro tree transformations preserve regularity (Theorem 10.4), and thus the domains of (compositions of) pebble macro tree transformations are regular (Corollary 10.5).

Next we obtain a type checking result for pebble macro tree transformations. Roughly speaking, the *type checking problem of XML transformations* is the question whether the results of an XML transformation of trees in an input DTD satisfy an output DTD. Formally, the type checking problem for pebble macro tree transducers [EM03, MBPS05a] is the following decision problem. Given two regular tree languages L_{in} and L_{out}, and a pebble macro tree transformation τ, we ask whether, for each input tree $s \in L_{in}$, the outputs of s translated by τ are in L_{out} or not (i.e., if it is true that $\tau(L_{in}) \subseteq L_{out}$). Now, we can conclude from the decomposition $n\text{-}PMTT \subseteq sMTT^{n+2}$ and the fact that the type checking problem for compositions of stay-macro tree transformations is decidable (see Corollary 44 of [EM03]) that the type checking problem for pebble macro tree transducers is decidable (Theorem 10.6).

Moreover, we obtain the following decidability results for the circularity problem of pebble macro tree transducers. Since the inverses of (compositions of) stay-macro tree transformations preserve regularity, it also follows from the decomposition result $n\text{-}PMTT \subseteq sMTT^{n+2}$ that the domains of pebble macro tree transformations are effectively regular (Corollary 10.5). This can be directly used to prove that weak circularity, circularity, and strong circularity are decidable for pebble macro tree transducers (Theorem 10.8, Theorem 10.9, and Corollary 10.11, respectively).

Finally, we consider domains of pebble tree transformations. In fact, we define the concept of an n-pebble alternating tree-walking automaton, which models the behaviour of an n-pebble tree transducer on its domain. It turns out that nonlooping n-pebble alternating tree-walking automata recognize the domains of not strongly circular pebble tree transformations. As the main result, we show that the domains of deterministic and not strongly circular n-pebble tree transformations form a proper hierarchy with respect to n (Theorem 10.25).

The thesis is organized as follows. In Section 2 we define the necessary basic concepts, then in Section 3 we introduce pebble macro tree transducers. In Section 4 we define and discuss the three concepts of circularity. In Section 5 we generalize the concept of the yield tree transformation introduced in [EV85] and show that every (deterministic) yield tree transformation can be computed by a (noncircular and deterministic) 0-pebble tree transducer. In Section 6 we show that the composition of an n-pebble tree transformations and a yield tree transformation can be computed by an n-pebble tree transducer. In Section 7 we present our yield-like decomposition result for (general) pebble macro tree transducers and in Section 8 for restricted pebble macro tree transducers. In Section 9 we show that each n-pebble tree transducer can be simulated by an $(n-1)$-pebble tree transducer. In Section 10 we list the corollaries and applications of the main result of Section 9. Finally, in Section 11 we conclude our results and give some future research topics.

The results of Sections 5, 6, and 8 are published in [FM08], the results of Sections 7, 9 in [FM09], and the results of Section 10, in [FM08, FM09], and [Muz08].

2 Definitions

2.1 Sets, relations, and strings

We denote the set of nonnegative integers by $\mathbb{N}$. For every $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. The empty set is denoted by $\emptyset$.

Sometimes we identify a singleton set $\{a\}$ with a. For a set A, $\mathcal{P}(A)$ denotes the power set of A. Moreover A^* denotes the set of *strings* over A; the *empty string* is denoted by ε. For a string $u \in A^*$, $|u|$ denotes its *length* and, for a symbol $a \in A$, $|u|_a$ denotes the number of occurrences of a in u. For every $n \geq 0$, we define $A^{\leq n} = \{u \in A^* \mid |u| \leq n\}$. For every $u \in A^*$, and $1 \leq l \leq |u|$, $u(l)$ denotes the l-th symbol in u.

Let $u, v, w \in A^*$ be strings. Then w *is a prefix of* v if there is a string $w' \in A^*$ such that $v = ww'$, note that ε and v are prefixes of v. A prefix w of v with $w \neq v$ is a *proper prefix of* v.

If A is an alphabet, i.e., a finite, nonempty set, then any subset $L \subseteq A^*$ is called a *language*. If L is a finite and nonempty language, then we write the strings of the language L^* in the form $[u_1; \ldots; u_l]$, where $l \geq 0$ and $u_1, \ldots, u_l \in L$. The empty string over L is denoted by $[\,]$.

Let $\rho \subseteq A \times B$ be a binary relation. The fact that $(a, b) \in \rho$ for some $a \in A$ and $b \in B$ is also denoted by $a\,\rho\,b$. For $A' \subseteq A$, we define $\rho(A') = \{b \in B \mid \exists a \in A' : a\,\rho\,b\}$. In case $A = B$, the lth power of ρ for $l \geq 0$, the transitive closure, the reflexive, transitive closure, and the inverse of ρ are denoted by ρ^l, ρ^+, ρ^*, and ρ^{-1} respectively.

The *composition* of $\rho \subseteq A \times B$ and $\tau \subseteq B \times C$ is the binary relation $\rho \circ \tau = \{(a, c) \subseteq A \times C \mid$ there is a $b \in B$ such that $a\,\rho\,b$ and $b\,\tau\,c\}$. Note that $(\rho \circ \tau)(a) = \tau(\rho(a))$. The notion of composition is extended to classes of relations. For two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of relations we define $\mathcal{C}_1 \circ \mathcal{C}_2 = \{\rho \circ \tau \mid \rho \in C_1$ and $\tau \in C_2\}$. For a class $\mathcal{C}$ and every $n \geq 1$ we define $\mathcal{C}^n = \mathcal{C}$ if $n = 1$ and $\mathcal{C}^n = \mathcal{C}^{n-1} \circ \mathcal{C}$ otherwise.

Let $\Rightarrow \subseteq A \times A$ be a binary relation. We say that $\Rightarrow$ is *terminating* if there is not an infinite sequence $a_1, a_2, \ldots$ of the elements of A such that $a_1 \Rightarrow a_2 \Rightarrow \ldots$. Moreover, $\Rightarrow$ is *locally confluent* if, for every $a, b, c \in A$, the transitions $a \Rightarrow b$ and $a \Rightarrow c$ imply that there is a $d \in A$ such that $b \Rightarrow^* d$ and $c \Rightarrow^* d$. For every $a, b \in A$, the element b is a *normal form* of a (with respect to $\Rightarrow$) if $a \Rightarrow^* b$ and there is no $c \in A$ such that $b \Rightarrow c$. If a has exactly one normal form, then we denote it by $nf(a, \Rightarrow)$ and say that $nf(a, \Rightarrow)$ *exists*. We will use the following fact, cf. [Hue80] or [Boo83].

Proposition 2.1 If a binary relation $\Rightarrow \subseteq A \times A$ is terminating and locally confluent, then $nf(a, \Rightarrow)$ exists for every $a \in A$.

2.2 Trees, tree languages, and tree transformations

A *ranked alphabet* is an ordered pair $(\Sigma, rank)$, where Σ is a finite set and $rank$ is a mapping of type $\Sigma \to \mathbb{N}$. For every $k \geq 0$, we define $\Sigma^{(k)} = \{\sigma \in \Sigma \mid rank(\sigma) = k\}$. Moreover, we denote by $maxr(\Sigma)$ the maximum of ranks of symbols of Σ, i.e.,

$maxr(\Sigma) = \max\{rank(\sigma) \mid \sigma \in \Sigma\}$. In the rest of the thesis, we drop *rank* and write just Σ for $(\Sigma, rank)$. Moreover, Σ and Δ will denote ranked alphabets. For a set A, we denote by $\langle \Sigma, A\rangle$ the ranked alphabet $\Sigma \times A$ with ranking $rank(\langle\sigma, a\rangle) = rank(\sigma)$ for every $\langle\sigma, a\rangle \in \Sigma \times A$.

The set of *trees over Σ and a set A*, denoted by $T_\Sigma(A)$, is the smallest set $T \subseteq (\Sigma \cup \{(,)\} \cup \{,\})^*$ such that $\Sigma^{(0)} \cup A \subseteq T$ and whenever $k \geq 1$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T$, then $\sigma(t_1, \ldots, t_k) \in T$. In case $A = \emptyset$, we write T_Σ for $T_\Sigma(A)$. Certainly, $T_\Sigma = \emptyset$ if and only if $\Sigma^{(0)} = \emptyset$.

For every tree $s \in T_\Sigma$, we define the set $pos(s) \subseteq \mathbb{N}^*$ of *the nodes of s* as follows. We let $pos(s) = \{\varepsilon\}$ if $s \in \Sigma^{(0)}$, and $pos(s) = \{\varepsilon\} \cup \{iu \mid 1 \leq i \leq k,\ u \in pos(s_i)\}$ if $s = \sigma(s_1, \ldots, s_k)$ for some $k \geq 1$, $\sigma \in \Sigma^{(k)}$ and $s_1, \ldots, s_k \in T_\Sigma$. Here ε corresponds to the root node labeled by σ and, for every $u \in pos(s)$ with l children, ui is the i-th child of u for all $1 \leq i \leq l$.

Now, for a tree $s \in T_\Sigma$ and a node $u \in pos(s)$, *the subtree of s at u*, denoted by s/u and *the label of s at node u*, denoted $lab(s, u) \in \Sigma$ are defined in a standard way. By the *root of s* we mean the label of s at node ε. Moreover, u is a *leaf* of s if, for every $i \in \mathbb{N}$, $ui \notin pos(s)$. Finally we define the parent of u, denoted by $parent(u)$ and the child number of u, denoted by $childno(u)$ as follows:

(i) if $u = \varepsilon$, then $childno(u) = 0$ and $parent(u)$ is undefined,

(ii) if $u = u'j$ for some $u' \in pos(s)$ and $j \in \mathbb{N}$, then $childno(u) = j$ and $parent(u) = u'$.

Any set $L \subseteq T_\Sigma$ is a *tree language* and any relation $\tau \subseteq T_\Sigma \times T_\Delta$ is a *tree transformation.* The *domain* of τ is $dom(\tau) = \{s \in T_\Sigma \mid \exists t \in T_\Delta : (s,t) \in \tau\}$. Let $\mathcal{C}$ be a tree transformation class. Then $dom(\mathcal{C}) = \{dom(\tau) \mid \tau \in \mathcal{C}\}$.

The *complement of* L is the tree language $\overline{L} = T_\Sigma - L$. If $\mathcal{L}$ is a class of tree languages, then *co*-$\mathcal{L} = \{\overline{L} \mid L \in \mathcal{L}\}$.

We will freely use the concepts of a *regular tree language* and a *(finite) tree automaton.* The unfamiliar reader can consult [GS84, GS97] for these concepts. We denote the class of regular tree languages by REG. We will need the following result.

Proposition 2.2 REG = *co*-REG.

The inverse of a tree transformation $\tau \subseteq T_\Sigma \times T_\Delta$ *preserves regularity* if, for each regular tree language $L \subseteq T_\Delta$, the tree language $\tau^{-1}(L)$ is regular.

2.3 Substitutions of strings and trees

Computation with trees is based on operations called *tree substitution.* In this subsection we introduce the ones which we will need in the thesis.

We maintain the fixed set $Y = \{y_1, y_2, \ldots\}$ of *variables.* For every $m \geq 0$, we let $Y_m = \{y_1, \ldots, y_m\}$. We will assume Y to be disjoint with each ranked alphabet considered in this thesis.

First we define string substitution. Let A be an alphabet, $n \geq 0$ an integer, $u, v_1, \ldots, v_n \in A^*$ strings, and $a_1, \ldots, a_n \in A$ pairwise different symbols (which are not necessarily in u). We denote by $u[a_1 \leftarrow v_1, \ldots, a_n \leftarrow v_n]$ the result of substituting $v_1, \ldots, v_n$ simultaneously for the occurrences of $a_1, \ldots, a_n$ in u, respectively. Note that $u[a_1 \leftarrow v_1, \ldots, a_n \leftarrow v_n] \in A^*$.

Now we introduce the concept of *string substitution defined by condition.* Let A be an alphabet and $P : A \times A^* \rightarrow \{\mathit{true}, \mathit{false}\}$ a binary predicate such that $\{(a, w) \in A \times A^* \mid P(a, w) = \mathit{true}\}$ is a partial function. Then, for every $u \in A^*$, we define

$$u[a \leftarrow w \mid P(a, w) = \mathit{true}] = u[a_1 \leftarrow w_1, \ldots, a_n \leftarrow w_n],$$

where $\{(a_1, w_1), \ldots, (a_n, w_n)\} = \{(a, w) \in A \times A^* \mid P(a, w) = \mathit{true}\}$.

Since trees are also strings, a special case of the string substitution is the *first-order tree substitution.* Let $n \geq m \geq 0$, $s \in T_\Sigma(Y_m)$ and $t_1, \ldots, t_n \in T_\Sigma(A)$. Then the string $s[y_1 \leftarrow t_1, \ldots, y_n \leftarrow t_n]$ is also a tree and it is in $T_\Sigma(A)$.

Next we define the *second-order tree substitution.* Let $s \in T_\Sigma$ be a tree, $n \geq 0$, $\sigma_1, \ldots, \sigma_n \in \Sigma$ different symbols of rank $k_1, \ldots, k_n$, respectively, and $t_1 \in T_\Sigma(Y_{k_1}), \ldots, t_n \in T_\Sigma(Y_{k_n})$ trees. The result of the second-order tree substitution of $\sigma_1, \ldots, \sigma_n$ by $t_1, \ldots, t_n$, respectively, in s, is the tree denoted by $s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!] \in T_\Sigma$ which is defined as follows.

(i) If $s = \alpha \in \Sigma^{(0)}$, then

- if $\alpha = \sigma_i$ for some $1 \leq i \leq n$, then $s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!] = t_i$,
- otherwise $s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!] = \alpha$.

(ii) If $s = \sigma(s_1, \ldots, s_k)$, for some $k \geq 1$, $\sigma \in \Sigma^{(k)}$ and $s_1, \ldots, s_k \in T_\Sigma$, then

- if $\sigma = \sigma_i$ for some $1 \leq i \leq n$ (and hence $k = k_i$), then $s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!] = t_i[y_1 \leftarrow s_1[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!], \ldots, y_{k_i} \leftarrow s_{k_i}[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!]]$,
- otherwise $s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!] =$
  $\sigma(s_1[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!], \ldots, s_k[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!])$.

The *second-order tree substitution defined by condition* is introduced as follows. Let $P : \Sigma \times T_\Sigma(Y) \rightarrow \{\mathit{true}, \mathit{false}\}$ be a binary predicate such that $\{(\sigma, t) \in \Sigma \times T_\Sigma(Y) \mid P(\sigma, t) = \mathit{true}\}$ is a partial function and, for every $\sigma \in \Sigma$ and $t \in T_\Sigma(Y)$, if $P(\sigma, t) = \mathit{true}$, then $t \in T_\Sigma(Y_{\mathit{rank}(\sigma)})$. For every tree $s \in T_\Sigma$,

$$s[\![\sigma \leftarrow t \mid P(\sigma, t) = \mathit{true}]\!] = s[\![\sigma_1 \leftarrow t_1, \ldots, \sigma_n \leftarrow t_n]\!],$$

where $\{(\sigma, t) \in \Sigma \times T_\Sigma(Y) \mid P(\sigma, t) = \mathit{true}\} = \{(\sigma_1, t_1), \ldots, (\sigma_n, t_n)\}$.

Next we define the first-order and the second-order substitution for a node of a tree.

Let $s \in T_\Sigma$, $t \in T_\Sigma(Y)$ and $u \in \mathit{pos}(s)$. The result of the *first-order substitution of* t *for* u *in* s is the tree $s[u \leftarrow t] \in T_\Sigma(Y)$ defined as follows.

(i) If $s = \alpha \in \Sigma^{(0)}$ (which implies $u = \varepsilon$), then $s[u \leftarrow t] = t$.

(ii) If $s = \sigma(s_1, \ldots, s_k)$ for some $k \geq 1$, $\sigma \in \Sigma^{(k)}$ and $s_1, \ldots, s_k \in T_\Sigma$, then

- if $u = \varepsilon$, then $s[u \leftarrow t] = t$;
- if $u = iv$, where $1 \leq i \leq k$ and $v \in \mathit{pos}(s_i)$, then

$$s[u \leftarrow t] = \sigma(s_1, \ldots, s_{i-1}, s_i[v \leftarrow t], s_{i+1}, \ldots, s_k).$$

The result of the *second-order substitution of t for u in s* is the tree $s[\![u \leftarrow t]\!] \in T_\Sigma(Y)$ defined as follows. Let $\mathit{lab}(s, u) = \sigma \in \Sigma^{(k)}$ for some $k \geq 0$. Then

$$s[\![u \leftarrow t]\!] = s[u \leftarrow t[y_1 \leftarrow s/u1, \ldots, y_k \leftarrow s/uk]].$$

Note that if $t \in T_\Sigma$ or $k = 0$, then $s[\![u \leftarrow t]\!] = s[u \leftarrow t]$.

Finally we define *OI* substitution of trees, see [ES77]. The term "OI" is an abbreviated form of "Outside-In". Let $s \in T_\Sigma(Y)$ be a tree, $n \geq 0$, and $L_1, \ldots, L_n \subseteq T_\Sigma(Y)$ tree languages. The result of the *OI substitution of* $L_1, \ldots, L_n$ *in* s is the tree language $s \overset{OI}{\leftarrow} (L_1, \ldots, L_n) \subseteq T_\Sigma(Y)$ defined by induction as follows.

(i) If $s \in Y$, then

$$s \overset{OI}{\leftarrow} (L_1, \ldots, L_n) = \begin{cases} L_i & \text{if } s = y_i \text{ for some } 1 \leq i \leq n \\ \{s\} & \text{otherwise.} \end{cases}$$

(ii) If $s = \sigma(s_1, \ldots, s_k)$ for some $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma(Y)$, then

$$s \overset{OI}{\leftarrow} (L_1, \ldots, L_n) = \{\sigma(t_1, \ldots, t_k) \mid \text{for every } 1 \leq i \leq k,\ t_i \in s_i \overset{OI}{\leftarrow} (L_1, \ldots, L_n)\}.$$

In case $n = 0$ we have $s \overset{OI}{\leftarrow} (\,) = \{s\}$. In particular, we write $\sigma(L_1, \ldots, L_n)$ for $\sigma(y_1, \ldots, y_n) \overset{OI}{\leftarrow} (L_1, \ldots, L_n)$.

The result of the *OI substitution of* $L_1, \ldots, L_n$ *in the tree language* $L \subseteq T_\Sigma(Y)$ is the tree language

$$L \overset{OI}{\leftarrow} (L_1, \ldots, L_n) = \bigcup_{s \in L} s \overset{OI}{\leftarrow} (L_1, \ldots, L_n).$$

Note that $\emptyset \overset{OI}{\leftarrow} (L_1, \ldots, L_n) = \emptyset$ and in case $n = 0$ we have $L \overset{OI}{\leftarrow} (\,) = L$.

2.4 Simultaneous induction

In proofs concerning pebble macro tree transducers we will apply the following version of *simultaneous induction*.

Let $\mathrm{K} : (\mathbb{N} - \{0\}) \to \{\mathit{true}, \mathit{false}\}$ and $\mathrm{L} : \mathbb{N} \to \{\mathit{true}, \mathit{false}\}$ be predicates. For every $l \geq 1$, we say that $\mathrm{K}[l]$ holds if $\mathrm{K}(1) = \mathit{true}, \ldots, \mathrm{K}(l) = \mathit{true}$ and we say that K holds if, for every $l \geq 1$, $\mathrm{K}[l]$ holds. We use the same terminology for $l \geq 0$, $\mathrm{L}[l]$, and L.

The simultaneous induction is a proof method which, in certain concrete instances of K and L, is suitable to prove that K and L hold. Moreover, it can be used for pebble macro tree transducers very well because statements concerning them can be described as instances of K and L.

Let K and L be predicates as above and consider the following three statements:

IB: L(0) holds.

IS1: For every $l \geq 0$, if L$[l]$ holds, then K$(l + 1)$ holds.

IS2: For every $l \geq 1$, if K$[l]$ holds, then L(l) holds.

The principle of Simultaneous induction is based on the fact that if statements IB, IS1, and IS2 hold, then also K and L hold. Here IB, IS1, and IS2 are the base of the induction, the induction step 1, and the induction step 2, respectively.

3 Pebble macro tree transducers

3.1 An informal introduction

We provide an intuitive introduction for pebble macro tree transducers, before defining their syntax in an exact way.

An n-pebble macro tree transducer M is a finite-state device that takes an input tree and generates an output tree. Each state has a rank, hence states are ranked symbols. M has finitely many rules of the form $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \rightarrow \zeta$.

On the left-hand side of the rule, q is a state of M, σ is an input symbol, b is a bit vector which can be used to test the presence of the pebbles at a node of the input tree, and j is a number to test the child number of a node of the input tree. Finally, the symbols $y_1, \ldots, y_m$ are so called parameter variables. The right-hand side ζ of the rule is a tree over the output symbols, the parameter variables $y_1, \ldots, y_m$ and ranked pairs of the form $\langle q, \varphi\rangle$, where q is a state and φ is an instruction. The tree ζ is built in the way as the right-hand side of a rule of a macro tree transducer in [EV85], i.e., it contains recursive calls of applications of other rules. Instruction can be moving instructions as *stay*, *up*, and $down_i$ and pebble instructions as *drop* and *lift*.

M takes an input tree s and makes a computation over sentential forms.

A general sentential form ξ is a tree over output symbols and configurations. A configuration is a pair $\langle q, h\rangle$, where q is a current state and $h = (u, [u_1; \ldots; u_l])$ is a pebble configuration, where u is a pointer to a current node of s and $[u_1; \ldots; u_l]$ is a vector of pointers to the nodes of s where pebbles $1, \ldots, l$ are placed, respectively.

At the beginning of the computation, the current state of M is its initial state and the current node is the root node of s. Moreover there are no pebbles at the nodes of s. Thus the initial sentential form is the configuration $\langle q_0, (\varepsilon, [\,])\rangle$, where q_0 is the initial state of M, ε is the pointer to the root of s, and $[\,]$ is an empty list of the pebble pointers. Then, M acts as follows. It takes a configuration node $\langle q, h\rangle$ with $h = (u, [u_1; \ldots; u_l])$ of the current sentential form ξ. Then M takes a rule $r : \langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \rightarrow \zeta$, if any, such that σ is the label of node u of s, the bit vector b fits the presence of pebbles at u, and j is the child number of node u. Now M applies the rule r to the sentential form ξ in the following way. Every instruction φ occurring in ζ is applied to the pebble configuration $(u, [u_1; \ldots; u_l])$. The result of the application of φ to $(u, [u_1; \ldots; u_l])$ is denoted by $\varphi((u, [u_1; \ldots; u_l]))$. For instance, if $\varphi = down_i$, then $\varphi((u, [u_1; \ldots; u_l])) = (ui, [u_1; \ldots; u_l])$ indicating that the pointed moves down to the ith son of the current node u. Furthermore, if $\varphi = drop$, then $\varphi((u, [u_1; \ldots; u_l])) = (u, [u_1; \ldots; u_l; u])$ indicating that the next pebble is dropped at node u. The pebbles are used in a stack-like fashion, i.e., if $l \leq n$ pebbles are on the tree s, then either the $(l + 1)$th pebble can be dropped (provided $l < n$) or the lth pebble can be lifted. The obtained pebble configuration $\varphi((u, [u_1; \ldots; u_l]))$ is substituted for φ in ζ. This process yields a tree ζ'. Finally ζ', which may contain the variables $y_1, \ldots, y_m$, is substituted for the configuration node $\langle q, h\rangle$ in a second-order way in ξ. The result of this substitution yields the next sentential form.

The computation may produce a sentential form which contains no configurations. Such a sentential form is the output of M to the input s. Note that a computation may never terminate due to circles in the computation. This problem will be discussed in Section 4.

The concept of n-pebble macro tree transducers of [EM03] was introduced with *weak pebble handling*. In [FM09] we generalized it by allowing *strong pebble handling* to the model, see [EH07, MSS06], however we left its original name unchanged. Roughly speaking, strong pebble handling of a tree-walking device generalizes weak pebble handling only in lifting of pebbles as follows:

Lifting pebbles in weak pebble handling: If $1 \leq l \leq n$ pebbles are placed on the input tree, then pebble l can be lifted provided that it is at the current node;

Lifting pebbles in strong pebble handling: If $1 \leq l \leq n$ pebbles are placed on the input tree, then pebble l can be lifted regardless of its position;

Next we define n-pebble macro tree transducers (with strong pebble handling) in an exact way. We follow [FM09] in elaborating the details.

3.2 Syntax of pebble macro tree transducers

We will need the concepts of the "instructions" of a pebble macro tree transducer.

Definition 3.1 For every integer $d \geq 0$ we define the set

$$I_d = \{\mathit{drop}, \mathit{lift}, \mathit{stay}, \mathit{up}, \mathit{down}_1, \mathit{down}_2, \ldots, \mathit{down}_d\}.$$

The elements of I_d are called *instructions*. For a symbol $\sigma \in \Sigma$, $n \geq 0$, bit vector $b \in \{0,1\}^{\leq n}$, and $j \in \{0, 1, \ldots, \mathit{maxr}(\Sigma)\}$, let $I_{\sigma,b,j,n} \subseteq I$ be the smallest subset of I_n satisfying that

(i) $\mathit{stay} \in I_{\sigma,b,j,n}$,

(ii) if $j \neq 0$, then $\mathit{up} \in I_{\sigma,b,j,n}$,

(iii) for every $1 \leq i \leq \mathit{rank}(\sigma)$ we have $\mathit{down}_i \in I_{\sigma,b,j,n}$,

(iv) if $|b| < n$, then $\mathit{drop} \in I_{\sigma,b,j,n}$, and

(v) if $b \neq \varepsilon$ then $\mathit{lift} \in I_{\sigma,b,j,n}$.

If n is clear from the context, we also write $I_{\sigma,b,j}$ for $I_{\sigma,b,j,n}$. $\diamond$

In the present thesis the bit vector $b \in \{0,1\}^{\leq n}$ will store the following information about a given node u of an input tree s: (1) There are $|b|$ pebbles placed at s and (2) pebble i is at u iff $b(i) = 1$. Note that, in the definition of instructions in [EM03], condition (v) requires that both $b \neq \varepsilon$ and $b(|b|) = 1$ hold. This assures the weak pebble handling, i.e., that the last pebble can be lifted only if it is on the position of the pointer.

Definition 3.2 For $n \geq 0$, an *n-pebble macro tree transducer* (shortly *n-pmtt*) is a system $M = (Q, \Sigma, \Delta, q_0, R)$, where

- Q is a ranked alphabet, the *set of states*,
- Σ and Δ are called the *input ranked alphabet* and the *output ranked alphabet*, respectively,
- $q_0 \in Q^{(0)}$ is a distinguished state of rank 0, the *initial state*, and
- R is a finite *set of rules* of the form $\langle q, \sigma, b, j\rangle(y_1, \dots, y_m) \to \zeta$, where $m \geq 0$, $q \in Q^{(m)}$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, $j \in \{0, 1, \dots, maxr(\Sigma)\}$ and $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_m)$. (Note that here the set $\langle Q, I_{\sigma,b,j}\rangle$ is a ranked alphabet and the rank of an element $\langle q, \varphi\rangle \in \langle Q, I_{\sigma,b,j}\rangle$ is equal to the rank of q.) $\diamond$

In order to make the right-hand sides of the rules of M well defined, from now on in this thesis we will assume that Δ is disjoint with $\langle Q, I_{maxr(\Sigma)}\rangle$. By a *pebble macro tree transducer (pmtt)* we mean an n-pmtt for some $n \geq 0$.

We will need the following syntactical restrictions for pmtts, which are similarly defined in [EM03].

Definition 3.3 An n-pmtt $M = (Q, \Sigma, \Delta, q_0, R)$ is called

- an *n-pebble tree transducer* (*n-ptt*) if each state in Q has rank zero;
- a *stay-macro tree transducer* (*smtt*) if $n = 0$ and there is no *up* instruction in the right-hand sides of the rules;
- a *macro tree transducer* (*mtt*) if it is an smtt and there is no *stay* instruction in the right-hand sides of the rules; $\diamond$

By a *pebble tree transducer* (*ptt*) we mean an n-ptt for some $n \geq 0$.

For every $q \in Q^{(m)}$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, 1, \dots, maxr(\Sigma)\}$, let $rhs_M(q, \sigma, b, j) = \{\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_m) \mid \langle q, \sigma, b, j\rangle(y_1, \dots, y_m) \to \zeta \in R\}$. If M is clear from the context, then we write $rhs(q, \sigma, b, j)$ for $rhs_M(q, \sigma, b, j)$.

Next we introduce other syntactic restrictions for pmtts.

Definition 3.4 The n-pmtt M is

- *deterministic* (resp. *total*), if, for every $q \in Q^{(m)}$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, 1, \dots, maxr(\Sigma)\}$, we have $|rhs(q, \sigma, b, j)| \leq 1$ (resp. $|rhs(q, \sigma, b, j)| \geq 1$),
- *context-linear*, if, for every rule $\langle q, \sigma, b, j\rangle(y_1, \dots, y_m) \to \zeta \in R$ and $1 \leq i \leq m$, we have $|\zeta|_{y_i} \leq 1$. $\diamond$

We call the variables $y_1, \dots, y_m$ occurring in the specification of a rule of a pmtt *parameter variables* or just *parameters*. A tree $s \in T_\Sigma$ is called an *input tree to M* or just an input tree.

3.3 Semantics of pebble macro tree transducers

Next we make some preparations for defining the semantics of a pmtt. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an n-pmtt.

For an input tree $s \in T_\Sigma$, an *n-pebble configuration* (or: *pebble configuration*) over s and M is a pair $h = (u, \pi)$, where $u \in pos(s)$ is a node of s and $\pi \in (pos(s))^{\leq n}$, i.e., π is a string over $pos(s)$ of length at most n. The set of pebble configurations over s and M is denoted by $PC_{M,s}$.

A pebble configuration $h = (u, \pi) \in PC_{M,s}$, with $\pi = [u_1; \ldots ; u_l]$ contains the information that the node being scanned by M (the current node) of the input tree s is u and M put $l = |\pi|$ pebbles on the nodes $u_1, \ldots , u_l$ of s.

A pebble configuration will be a part of a configuration of M. Further, configurations of M will occur in sentential forms computed by M. The pmtt M computes the next sentential form by applying a rule to a configuration. However, a rule can be applied to that configuration only if its left hand-side fits to the result of a certain *test* of h, where h is the pebble configuration in the involved configuration of M. The result of this test is defined as follows.

Let $s \in T_\Sigma$ and $h = (u, [u_1; \ldots ; u_l]) \in PC_{M,s}$. Then $test(h) = (\sigma, b, j)$, where $\sigma = lab(s, u)$, $j = childno(u)$, and $b \in \{0, 1\}^*$ is a string (bit vector) of length l such that, for every $1 \leq i \leq l$, and $b(i) = 1$ if $u_i = u$ and $b(i) = 0$ otherwise.

Let $test(h) = (\sigma, b, j)$ and take an instruction $\varphi \in I_{\sigma,b,j}$. The *execution of φ on h* is the pebble configuration $\varphi(h)$ defined in the following way.

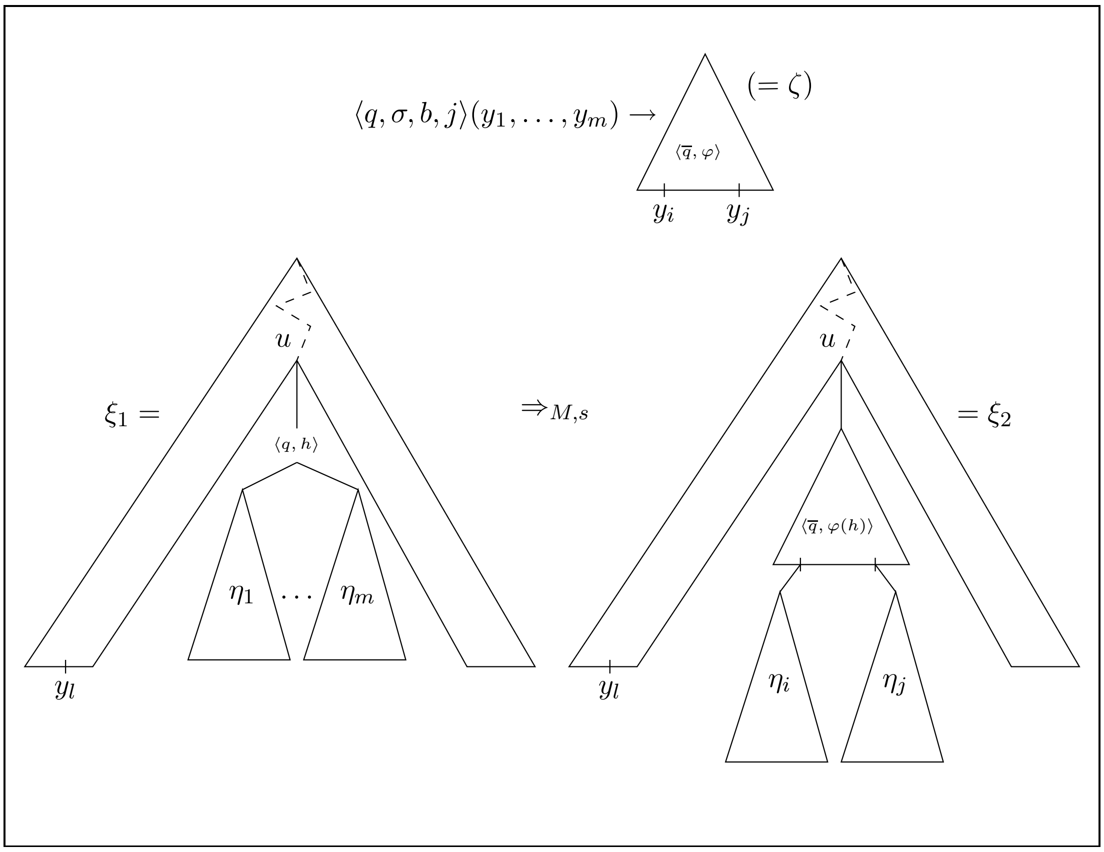
$$\varphi(h) = \begin{cases} (u, [u_1; \ldots ; u_l]) & \text{if } \varphi = stay, \\ (parent(u), [u_1; \ldots ; u_l]) & \text{if } \varphi = up, \\ (ui, [u_1; \ldots ; u_l]) & \text{if } \varphi = down_i, \\ (u, [u_1; \ldots ; u_l; u]) & \text{if } \varphi = drop, \\ (u, [u_1; \ldots ; u_{l-1}]) & \text{if } \varphi = lift. \end{cases}$$

Note that in case $\varphi \in \{stay, up, down_i \mid i \geq 1\}$, $\varphi(h)$ does not have effect on $[u_1; \ldots ; u_l]$. Hence we might write $(\varphi(u), [u_1; \ldots ; u_l])$ instead of $\varphi(h)$.

Definition 3.5 A *configuration of M* (over s) is a pair $\langle q, h\rangle$, where $q \in Q$ and $h \in PC_{M,s}$. $\diamond$

The set of configurations of M over s is denoted by $C_{M,s}$. Note that the set $C_{M,s}$ is a ranked alphabet with $rank(\langle q, h\rangle) = rank(q)$ for every $\langle q, h\rangle \in C_{M,s}$. The set $T_{\Delta\cup C_{M,s}}$ is the set of *sentential forms of M* (over s). Let us note that, if M is an n-ptt, then the rank of each configuration in $C_{M,s}$ is 0. Hence, a configuration may occur only at a leaf of a sentential form $\xi \in T_{\Delta\cup C_{M,s}}$. In order to make the sentential forms of M well defined, from now on in this thesis we will assume that Δ is disjoint with $C_{M,s}$ for each $s \in T_\Sigma$.

We introduce the computation relation of M as a binary relation over sentential forms. For technical reasons, we define the computation relation over the larger set $T_{\Delta\cup C_{M,s}}(Y)$. We also call the elements of $T_{\Delta\cup C_{M,s}}(Y)$ sentential forms.



$\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to$
$(= \zeta)$
$\langle \overline{q}, \varphi\rangle$
y_i
y_j
$\xi_1 =$
u
$\langle q, h\rangle$
η_1
$\ldots$
η_m
y_l
$\Rightarrow_{M,s}$
$= \xi_2$
$\langle \overline{q}, \varphi(h)\rangle$
η_i
η_j

Figure 1: For $\xi_1, \xi_2 \in T_{\Delta \cup C_{M,s}}(Y)$, the step $\xi_1 \Rightarrow_{M,s} \xi_2$ with rule $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$.

A step of the computation is a substitution of a tree for a configuration $\langle q, h\rangle$ occurring in a sentential form ξ. Following [EM03] we allow only *outside-in substitution*, i.e., OI-semantics, cf. [EV85, Fis68], which means that the configuration $\langle q, h\rangle$, occurring at node $u \in pos(s)$, is substituted in ξ only if no proper ancestor of u is labeled by a configuration in $C_{M,s}$.

The intuition behind the idiom outside-in is the following. If we consider a tree as a hierarchic term (with parentheses), then the inner part of it is the one which is deeply inside this hierarchy. Therefore the outside-in substitution described above proceeds from outer part to the inner part of the tree.

Formally, we define the following. A node $u \in pos(\xi)$ of a sentential form ξ of M is an *outside active node* if $lab(\xi, u) \in C_{M,s}$ and, for every proper prefix u' of u, it holds $lab(\xi, u') \in \Delta$. Let us observe that if M is a ptt, then, due to the above note, each node $u \in pos(\xi)$ satisfying $lab(\xi, u) \in C_{M,s}$ is an outside active node.

Let $s \in T_\Sigma$ be an input tree. The *computation relation of M on s*, denoted by $\Rightarrow_{M,s}$, is a binary relation over $T_{\Delta \cup C_{M,s}}(Y)$ defined as follows. For every $\xi_1, \xi_2 \in T_{\Delta \cup C_{M,s}}(Y)$ we have $\xi_1 \Rightarrow_{M,s} \xi_2$ if and only if

1. there is an outside-active node $u \in pos(\xi_1)$ such that $lab(\xi_1, u) = \langle q, h\rangle$ with $\langle q, h\rangle \in C^{(m)}_{M,s}$ for some $m \geq 0$,
2. there is a rule $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$ in R such that $test(h) = (\sigma, b, j)$ and

$$\xi_2 = \xi_1[\![u \leftarrow \zeta[\![\langle \overline{q}, \varphi\rangle \leftarrow \langle \overline{q}, \varphi(h)\rangle(y_1, \ldots, y_{rank(\overline{q})}) \mid \overline{q} \in Q,\ \varphi \in I_{\sigma,b,j}]\!]]\!].$$

We demonstrate one step of the derivation $\Rightarrow_{M,s}$ in Figure 1.

The *tree transformation computed by* M is the relation

$$\tau_M = \{(s, t) \in T_\Sigma \times T_\Delta \mid \langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} t\}.$$

The classes of tree transformations computed by n-pmtts, n-ptts, smtts, and mtts are denoted by n-$PMTT$, n-PTT, $sMTT$, and MTT, respectively. The deterministic subclass of each above class is denoted by prefixing the class with d. For instance, n-$dPMTT$ stands for the class of tree transformations computed by deterministic n-pebble macro tree transducers.

3.4 Examples of pmtts

In this subsection we give an example of a 1-ptt and examples of two 1-pmtts.

Example 3.6 Let $M_1 = (Q, \Sigma, \Delta, q_0, R)$ be the 1-ptt defined as follows.

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$
- $\Delta = \{1^{(1)}, 2^{(1)}, \#^{(1)}, \alpha^{(0)}\}$
- R consists of the rules:

(r_1): $\langle q_0, \sigma, \varepsilon, 0\rangle \to \langle q_0, drop\rangle$	
(r_2): $\langle q_0, \sigma, b, j\rangle \to 1(\langle q_0, down_1\rangle)$	$(b \in \{0,1\},\ j \in \{0,1,2\})$
(r_3): $\langle q_0, \sigma, b, j\rangle \to 2(\langle q_0, down_2\rangle)$	$(b \in \{0,1\},\ j \in \{0,1,2\})$
(r_4): $\langle q_0, \sigma, 0, j\rangle \to \alpha$	$(j \in \{0,1,2\})$
(r_5): $\langle q_0, \alpha, 0, j\rangle \to \alpha$	$(j \in \{0,1,2\})$
(r_6): $\langle q_0, \sigma, 0, j\rangle \to \#(\langle q_1, lift\rangle)$	$(j \in \{0,1,2\})$
(r_7): $\langle q_0, \alpha, 0, j\rangle \to \#(\langle q_1, lift\rangle)$	$(j \in \{0,1,2\})$
(r_8): $\langle q_1, \sigma, \varepsilon, j\rangle \to \langle q_2, drop\rangle$	$(j \in \{0,1,2\})$
(r_9): $\langle q_1, \alpha, \varepsilon, j\rangle \to \langle q_2, drop\rangle$	$(j \in \{0,1,2\})$
(r_{10}): $\langle q_2, \sigma, b, j\rangle \to \langle q_2, up\rangle$	$(b \in \{0,1\},\ j \in \{1,2\})$
(r_{11}): $\langle q_2, \alpha, b, j\rangle \to \langle q_2, up\rangle$	$(b \in \{0,1\},\ j \in \{1,2\})$
(r_{12}): $\langle q_2, \sigma, b, 0\rangle \to \langle q_0, stay\rangle$	$(b \in \{0,1\})$

(r_{13}): $\langle q_2, \alpha, b, 0\rangle \to \langle q_0, stay\rangle$ $\qquad (b \in \{0,1\})$

Intuitively, M_1 works on an input tree $s \in T_\Sigma$ in the following way.

(1) First M_1 drops the pebble at the root of s, see rule (r_1).

(2) The iteration step. Assume that the pointer points at node $u \in pos(s)$. Then M_1 chooses nondeterministically among the following activities.

(a) If $lab(s,u) = \sigma$, then M_1 moves down to the ith child of u and writes i to the output for some $i \in \{1,2\}$, see rules (r_2)-(r_3). Then the iteration starts again.

(b) If the pebble is not at u, then M_1 writes a separator # to the output, replaces the pebble to the current node u (by applying *lift* and *drop*, note that M_1 has a strong pebble), finally M_1 moves *up* to the root node, see rules (r_6)-(r_{13}). Then the iteration starts again.

(c) If the pebble is not at u, then M_1 writes α to the output and terminates, see rules (r_4)-(r_5).

It is easy to see that, taking s as input, M_1 outputs all monadic trees of the form $u_1\#u_2\#\ldots u_{k-1}\#u_k\alpha$, where

- $u_1, \ldots, u_k \in pos(s)$,
- $k \geq 1$ and $u_1 \neq \varepsilon$, and
- for each $2 \leq i \leq k$ we have $u_{i-1} \neq u_i$.

Observe that we omitted the braces in monadic trees of T_Δ. The second condition holds, because M_1 cannot terminate without writing out at least one position of s, due to the fact that initially the pebble is placed at the root of s. Finally, the third condition holds, because, while computing u_i, the pebble is kept at the position u_{i-1}. On the other hand, a # or an α can be written out at u_i only if the pebble is not there, see (b) and (c).

Hence, the tree transformation induced by M_1 is

$$\tau_{M_1} = \{(s, u_1\#u_2\#\ldots u_{k-1}\#u_k\alpha) \mid s \in T_\Sigma,\ k \geq 1,\ u_1, \ldots, u_k \in pos(s),\\ u_1 \neq \varepsilon, \text{ and for each } 2 \leq i \leq k,\ u_{i-1} \neq u_i\}.$$

We note that the tree transformation τ_{M_1} can also be computed by a 1-ptt with weak pebble handling. However, those which we know do this in a more complicated way with more states and rules. Therefore, we think the above example demonstrates the advantage of the strong pebble handling well. $\diamond$

Example 3.7 Let us fix the ranked alphabet $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$. We will use the following concepts concerning a tree $s \in T_\Sigma$.

We call s a *right-α-tree* if its rightmost leaf is labeled by α. For example the trees α, $\sigma(\alpha,\alpha)$, and $\sigma(\sigma(\beta,\beta),\alpha)$ are right-α-trees, while β and $\sigma(\alpha,\beta)$ are not.

Assume that there are k leaf nodes of s. Then $bin(s)$ is the full binary tree over $\{\sigma, \alpha\}$ of height $k+1$. For example, $bin(\alpha) = bin(\beta) = \sigma(\alpha, \alpha)$ and $bin(\sigma(\beta, \alpha)) = \sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha))$.

The *leftmost path of s* is the path from the root of s to its leftmost leaf node.

Now we describe intuitively, how our 1-pmtt M_2 works. It processes an input tree $s \in T_\Sigma$ in three phases. The first two phases are sequential (pebble tree-walking) computations, while the last phase is a computation with branchings and pure macro calls. These three phases are as follows.

Phase 1: Initially, the pointer of M_2 is at the root of s. Descending on the leftmost path of s, M_2 moves its pointer to the topmost node u on that leftmost path which is the root of a right-α-tree. Then M_2 marks u with its pebble and continues in Phase 2. If such a u does not exist, then M_2 terminates with no output, cf. the rules of R_1 below.

Phase 2: Starting in the node pebbled in Phase 1, by a recursive descending search, M_2 finds the downmost node u' on the leftmost path of s, such that u' is the root of a right-α-tree. During the search, the root of the current candidate is marked by the pebble. Once another node is found on the leftmost path below the current candidate which is the root of a right-α-tree, by applying *lift* and *drop* instructions and its strong pebble capability, M_2 replaces the pebble on this node. Note that, due to strong pebble handling, this pebble actualization can be realized easily.

Phase 3: Let s' be the right-α-tree found in Phase 2. Using macro calls, M_2 computes $bin(s')$ for the s', cf. the rules of R_3.

Let $M_2 = (Q, \Sigma, \{\sigma, \alpha\}, q_0, R)$, where Q and R are defined as follows.

- $Q = Q^{(1)} \cup Q^{(0)}$, where $Q^{(1)} = \{q_{bin}\}$ and $Q^{(0)} = \{q_0, q_1, q_2, q_3, q_\alpha, q_\beta\}$.
- $R = R_1 \cup R_2 \cup R_3$, where

  R_1 consists of the rules:

  (r_1): $\langle q_0, \sigma, \varepsilon, j\rangle \to \langle q_0, down_2\rangle$, $(j \in \{0,1,2\})$
  (r_2): $\langle q_0, \alpha, \varepsilon, 2\rangle \to \langle q_\alpha, up\rangle$,
  (r_3): $\langle q_0, \beta, \varepsilon, 2\rangle \to \langle q_\beta, up\rangle$,
  (r_4): $\langle q_\alpha, \sigma, \varepsilon, 2\rangle \to \langle q_\alpha, up\rangle$,
  (r_5): $\langle q_\alpha, \sigma, \varepsilon, j\rangle \to \langle q_1, drop\rangle$, $(j \in \{0,1\})$
  (r_6): $\langle q_\beta, \sigma, \varepsilon, 2\rangle \to \langle q_\beta, up\rangle$,
  (r_7): $\langle q_\beta, \sigma, \varepsilon, j\rangle \to \langle q_0, down_1\rangle$, $(j \in \{0,1\})$
  (r_8): $\langle q_0, \alpha, \varepsilon, j\rangle \to \langle q_{bin}, stay\rangle(\alpha)$, $(j \in \{0,1\})$

  R_2 consists of the rules:

  (r_9): $\langle q_1, \sigma, 1, j\rangle \to \langle q_1, down_1\rangle$, $(j \in \{0,1\})$
  (r_{10}): $\langle q_1, \sigma, 0, j\rangle \to \langle q_1, down_2\rangle$, $(j \in \{1,2\})$
  (r_{11}): $\langle q_1, \alpha, 0, 2\rangle \to \langle q_\alpha, up\rangle$,

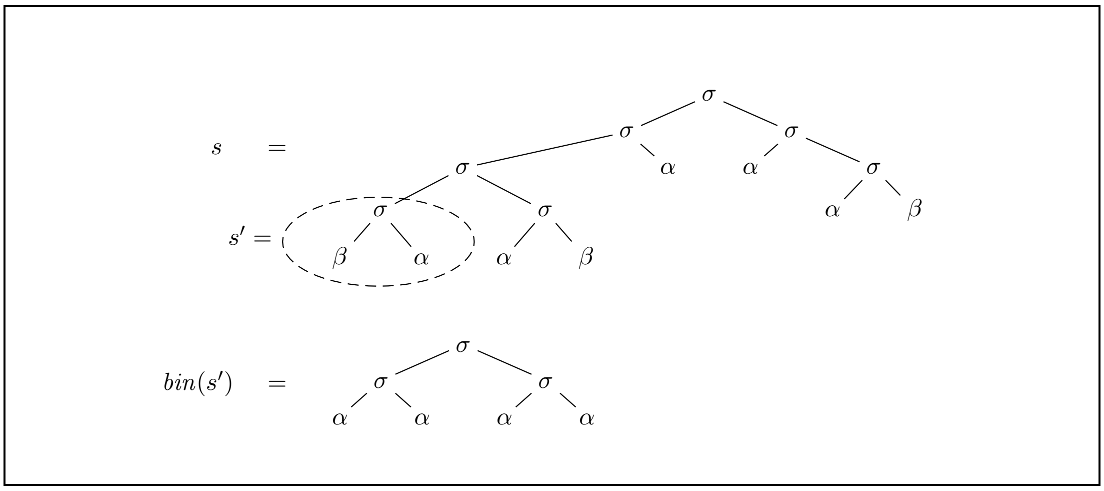


Figure 2: The trees s, s', and $bin(s')$.

(r_{12}): $\langle q_1, \beta, 0, 2\rangle \to \langle q_\beta, up\rangle$,
(r_{13}): $\langle q_\alpha, \sigma, 0, 2\rangle \to \langle q_\alpha, up\rangle$,
(r_{14}): $\langle q_\alpha, \sigma, 0, 1\rangle \to \langle q_2, \mathit{lift}\rangle$,
(r_{15}): $\langle q_2, \sigma, \varepsilon, 1\rangle \to \langle q_1, drop\rangle$,
(r_{16}): $\langle q_\beta, \sigma, 0, 2\rangle \to \langle q_\beta, up\rangle$,
(r_{17}): $\langle q_\beta, \sigma, 0, 1\rangle \to \langle q_1, down_1\rangle$,
(r_{18}): $\langle q_1, \alpha, 0, 1\rangle \to \langle q_{bin}, \mathit{lift}\rangle(\alpha)$,
(r_{19}): $\langle q_1, \beta, 0, 1\rangle \to \langle q_3, up\rangle$,
(r_{20}): $\langle q_3, \sigma, 0, 1\rangle \to \langle q_3, up\rangle$
(r_{21}): $\langle q_3, \sigma, 1, 1\rangle \to \langle q_{bin}, \mathit{lift}\rangle(\alpha)$,

and R_3 consists of the rules:

(r_{22}): $\langle q_{bin}, \sigma, \varepsilon, j\rangle(y_1) \to \langle q_{bin}, down_1\rangle(\langle q_{bin}, down_2\rangle(y_1))$, $(j \in \{0, 1, 2\})$
(r_{23}): $\langle q_{bin}, \alpha, \varepsilon, j\rangle(y_1) \to \sigma(y_1, y_1)$, and $(j \in \{0, 1, 2\})$
(r_{24}): $\langle q_{bin}, \beta, \varepsilon, j\rangle(y_1) \to \sigma(y_1, y_1)$. $(j \in \{0, 1, 2\})$

Note that M_2 is deterministic. As we saw, the tree transformation induced by M_2 is

$$\tau_{M_2} = \{(s, bin(s')) \mid s \in T_\Sigma, \text{ and } s' \text{ is the downmost right-}\alpha\text{-tree on the leftmost path of } s\}$$

We give an example of a computation of M_2. For this, let $s = \sigma(\sigma(\sigma(\sigma(\beta,\alpha),\sigma(\alpha,\beta)),\alpha),\sigma(\alpha,\sigma(\alpha,\beta)))$ be an input tree. The trees s, s', and $bin(s')$ can be seen in Fig. 2, where s', the tree marked by a dashed ellipse, is the downmost right-α-tree the root of which occurs in the leftmost path of s. Now M_2 works on s as follows.

Phase 1:

$$\begin{array}{llllllll}
\langle q_0,(\varepsilon,[\,])\rangle & \Rightarrow_{M_2} & \langle q_0,(2,[\,])\rangle & \Rightarrow_{M_2} & \langle q_0,(22,[\,])\rangle & \Rightarrow_{M_2} & \langle q_0,(222,[\,])\rangle \\
& \Rightarrow_{M_2} & \langle q_\beta,(22,[\,])\rangle & \Rightarrow_{M_2} & \langle q_\beta,(2,[\,])\rangle & \Rightarrow_{M_2} & \langle q_\beta,(\varepsilon,[\,])\rangle \\
& \Rightarrow_{M_2} & \langle q_0,(1,[\,])\rangle & \Rightarrow_{M_2} & \langle q_0,(12,[\,])\rangle & \Rightarrow_{M_2} & \langle q_\alpha,(1,[\,])\rangle \\
& \Rightarrow_{M_2} & \langle q_1,(1,[1])\rangle
\end{array}$$

Phase 2:

$$\begin{array}{llllll}
\langle q_1,(1,[1])\rangle & \Rightarrow_{M_2} & \langle q_1,(11,[1])\rangle & \Rightarrow_{M_2} & \langle q_1,(112,[1])\rangle \\
& \Rightarrow_{M_2} & \langle q_1,(1122,[1])\rangle & \Rightarrow_{M_2} & \langle q_\beta,(112,[1])\rangle \\
& \Rightarrow_{M_2} & \langle q_\beta,(11,[1])\rangle & \Rightarrow_{M_2} & \langle q_1,(111,[1])\rangle \\
& \Rightarrow_{M_2} & \langle q_1,(1112,[1])\rangle & \Rightarrow_{M_2} & \langle q_\alpha,(111,[1])\rangle \\
& \Rightarrow_{M_2} & \langle q_2,(111,[\,])\rangle & \Rightarrow_{M_2} & \langle q_1,(111,[111])\rangle \\
& \Rightarrow_{M_2} & \langle q_1,(1111,[111])\rangle & \Rightarrow_{M_2} & \langle q_3,(111,[111])\rangle \\
& \Rightarrow_{M_2} & \langle q_{bin},(111,[\,])\rangle(\alpha)
\end{array}$$

Phase 3:

$$\begin{array}{lll}
\langle q_{bin},(111,[\,])\rangle(\alpha) & \Rightarrow_{M_2} & \langle q_{bin},(1111,[\,])\rangle(\langle q_{bin},(1112,[\,])\rangle(\alpha)) \\
& \Rightarrow_{M_2} & \sigma(\langle q_{bin},(1112,[\,])\rangle(\alpha),\langle q_{bin},(1112,[\,])\rangle(\alpha)) \\
& \Rightarrow^2_{M_2} & \sigma(\sigma(\alpha,\alpha),\sigma(\alpha,\alpha))
\end{array}$$

Hence, $\tau_{M_2}(s) = \sigma(\sigma(\alpha,\alpha),\sigma(\alpha,\alpha))$. $\diamond$

Finally, we give an example of an 1-pmtt with weak pebble handling.

Example 3.8 Let $M_3 = (Q,\Sigma,\Delta,q_0,R)$ be a pebble macro tree transducer, where

- $Q = Q^{(0)} \cup Q^{(1)}$, $Q^{(0)} = \{q_0,q_1,q_2,q_3,q_0',q_1',q_0''\}$, and $Q^{(1)} = \{q_1''\}$,
- $\Sigma = \{\sigma^{(2)},\alpha^{(0)},\beta^{(0)}\}$,
- $\Delta = \{a^{(1)},e^{(0)}\}$,
- $R = R_1 \cup R_2 \cup R_3$ with R_1, R_2, and R_3 being the following sets of rules.
- $\underline{R_1}$:

  (r1): $\langle q_0,\alpha,\varepsilon,0\rangle \to \langle q_0'',\mathit{stay}\rangle$

  (r2): $\langle q_0,\sigma,\varepsilon,0\rangle \to \langle q_0,\mathit{down}_1\rangle$

  (r3): $\langle q_0,\gamma,\varepsilon,1\rangle \to \langle q_0',\mathit{drop}\rangle$, for every $\gamma \in \Sigma$

  (r4): $\langle q_1,\gamma,1,1\rangle \to \langle q_1,\mathit{lift}\rangle$, for every $\gamma \in \Sigma$

  (r5): $\langle q_1,\gamma,\varepsilon,1\rangle \to \langle q_2,\mathit{up}\rangle$, for every $\gamma \in \Sigma$

  (r6): $\langle q_2,\sigma,\varepsilon,j\rangle \to \langle q_0,\mathit{down}_2\rangle$, for every $j \in \{0,2\}$

  (r7): $\langle q_0,\alpha,\varepsilon,2\rangle \to \langle q_3,\mathit{up}\rangle$

  (r8): $\langle q_3,\sigma,\varepsilon,2\rangle \to \langle q_3,\mathit{up}\rangle$

  (r9): $\langle q_3,\sigma,\varepsilon,0\rangle \to \langle q_0'',\mathit{stay}\rangle$
- $\underline{R_2}$:

(r10): $\langle q'_0, \alpha, 1, 1\rangle \to \langle q_1, \mathit{stay}\rangle$

(r11): $\langle q'_0, \alpha, 0, j\rangle \to \langle q'_1, \mathit{up}\rangle$, for every $j \in \{1, 2\}$

(r12): $\langle q'_0, \sigma, b, j\rangle \to \langle q'_0, \mathit{down}_1\rangle$, for every $b \in \{0, 1\}$ and $j \in \{1, 2\}$

(r13): $\langle q'_0, \sigma, b, j\rangle \to \langle q'_0, \mathit{down}_2\rangle$, for every $b \in \{0, 1\}$, and $j \in \{1, 2\}$

(r14): $\langle q'_1, \sigma, 0, j\rangle \to \langle q'_1, \mathit{up}\rangle$, for every $j \in \{1, 2\}$

(r15): $\langle q'_1, \sigma, 1, 1\rangle \to \langle q_1, \mathit{stay}\rangle$

- $\underline{R_3}$:

(r16): $\langle q''_0, \sigma, \varepsilon, 0\rangle \to \langle q''_1, \mathit{down}_2\rangle(\langle q''_1, \mathit{down}_2\rangle(e))$

(r17): $\langle q''_0, \alpha, \varepsilon, 0\rangle \to a(e)$

(r18): $\langle q''_1, \sigma, \varepsilon, 2\rangle(y_1) \to \langle q''_1, \mathit{down}_2\rangle(\langle q''_1, \mathit{down}_2\rangle(y_1))$

(r19): $\langle q''_1, \alpha, \varepsilon, 2\rangle(y_1) \to a(y_1)$

Intuitively, M_3 works on an input tree s as follows. It checks the shape of s and it associates an output to s only if s has the shape $\sigma(s_1, \sigma(s_2, \ldots, \sigma(s_k, \alpha)\ldots))$ for some $k \geq 0$ and, for every $1 \leq i \leq k$, it holds that $|s_i|_\alpha \geq 1$. For this, M_3 first moves the reading head, with its rules in R_1, to the root of subtrees $s_1, \ldots, s_k$ in order. After finding the root of a subtree s_i, it puts a pebble there in order to find the way back to that position. Then it checks, with its rules in R_2, if the tree s_i contains at least one symbol α. Note that, during this latter activity, M_3 acts nondeterministically due to the rules (r12) and (r13). In case the test fails, M_3 halts because of the lack of an appropriate rule. However, if the check is successful, then M_3 computes the output tree with rules in R_3, which is a monadic tree of height $2^k + 1$. For this, it needs the macro rules with parameters (r18) and (r19), cf. Example 4.3 in [EV85].

In fact, M_3 computes the tree transformation

$$\tau_{M_3} = \{(\sigma(s_1, \sigma(s_2, \ldots, \sigma(s_k, \alpha)\ldots)),\ a^{2^k}(e)) \mid k \geq 0, |s_1|_\alpha \geq 1, \ldots, |s_k|_\alpha \geq 1\}.$$

Note that τ_{M_3} is a partial function.

As an example, we give the computation of M_3 on the input tree $s = \sigma(\sigma(\alpha, \beta), \alpha)$. For the sake of better readability we drop M_3 from $\Rightarrow_{M_3,s}$.

$$\left.\begin{array}{lll}\langle q_0, (\varepsilon, [\,])\rangle & \Rightarrow_s & \langle q_0, (1, [\,])\rangle \\ & \Rightarrow_s & \langle q'_0, (1, [1])\rangle\end{array}\right\} \text{rules (r2), (r3) in } R_1$$

$$\left.\begin{array}{lll}\langle q'_0, (1, [1])\rangle & \Rightarrow_s & \langle q'_0, (11, [1])\rangle \\ & \Rightarrow_s & \langle q'_1, (1, [1])\rangle \\ & \Rightarrow_s & \langle q_1, (1, [1])\rangle\end{array}\right\} \text{rules (r12), (r11), (r15) in } R_2$$

$$\left.\begin{array}{lll}\langle q_1, (1, [1])\rangle & \Rightarrow_s & \langle q_1, (1, [\,])\rangle \\ & \Rightarrow_s & \langle q_2, (\varepsilon, [\,])\rangle \\ & \Rightarrow_s & \langle q_0, (2, [\,])\rangle \\ & \Rightarrow_s & \langle q_3, (\varepsilon, [\,])\rangle \\ & \Rightarrow_s & \langle q''_0, (\varepsilon, [\,])\rangle\end{array}\right\} \text{rules (r4), (r5), (r6), (r7), (r9) in } R_1$$

$$\left.\begin{array}{lll}\langle q_0'', (\varepsilon, [\,])\rangle & \Rightarrow_s & \langle q_1'', (2, [\,])\rangle(\langle q_1'', (2, [\,])\rangle(e)) \\ & \Rightarrow_s & a(\langle q_1'', (2, [\,])\rangle(e)) \\ & \Rightarrow_s & a(a(e))\end{array}\right\}\text{ rules (r16), (r19), (r19) in } R_3$$

It is an exercise to show that τ_{M_3} can also be computed by a deterministic 1-pebble macro tree transducer. Indeed the test of the existence of a symbol α with rules in R_2 can also be performed by traversing the corresponding subtree, e.g., in preorder way. However, such a deterministic 1-pebble macro tree transducer has more rules than M_3 does. ◇

Further examples of relations computed by pebble tree transducers as well as pebble macro tree transducers can be found in [EM03].

4 The problem of circularity

In this section we discuss circularity properties of pmtts. A computation of a pmtt M may not terminate because it has a subcomputation which starts in a configuration $\langle q, h\rangle \in C_{M,s}$ being at an outside-active node of the current sentential form and the result of this subcomputation is a sentential form which also contains $\langle q, h\rangle$ at an outside-active node. Now after substituting the result of the subcomputation for $\langle q, h\rangle$, we get a sentential form which also has an outside-active node with label $\langle q, h\rangle$. Hence, the computation may lead to a circulus vitiosus. If M is deterministic, then it does. Since we face this phenomenon later, we introduce the following circularity concepts and discuss them. The results of this section can be found in Section 4.1 of [FM08].

Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an n-pmtt.

The first concept that we introduce, called weak circularity, seems to be unnatural, however we will need it in Section 8.3.

Definition 4.1 Let $s \in T_\Sigma$ be an input tree. We define the relation *one-step*$_{M,s} \subseteq C_{M,s} \times C_{M,s}$ as follows: $(\langle q, h\rangle, \langle p, f\rangle) \in$ *one-step*$_{M,s}$ if and only if there is a sentential form $\xi \in T_{\Delta\cup C_{M,s}}(Y)$ such that

- $\langle q, h\rangle(y_1, \ldots, y_{rank(q)}) \Rightarrow_{M,s} \xi$ and
- there is a (not necessarily outside-active) node $u \in pos(\xi)$ such that $lab(\xi, u) = \langle p, f\rangle$.

If M and s are clear from the context, then we write *one-step* for *one-step*$_{M,s}$. A configuration $\langle q, h\rangle \in C_{M,s}$ is weakly circular if $(\langle q, h\rangle, \langle q, h\rangle) \in$ *one-step*$^+$.

We say that M is *weakly circular* if there is an input tree $s \in T_\Sigma$ such that $C_{M,s}$ contains a weakly circular configuration. Otherwise M is *not weakly circular.* $\diamond$

Now we make steps to define the concepts of circularity and strong circularity.

Definition 4.2 Let $s \in T_\Sigma$ be an input tree. A configuration $\langle q, h\rangle \in C_{M,s}$ is *circular* if there is a sentential form $\xi \in T_{\Delta\cup C_{M,s}}(Y)$ such that

- $\langle q, h\rangle(y_1, \ldots, y_{rank(q)}) \Rightarrow^+_{M,s} \xi$ and
- there is an outside-active node $u \in pos(\xi)$ such that $lab(\xi, u) = \langle q, h\rangle$.

We say that M is

1) *circular* if there is an input tree s such that $C_{M,s}$ contains a circular configuration (cf. page 659 of [EM03]). Otherwise M is *noncircular.*

2) *strongly circular* if there is an input tree s, a sentential form $\xi \in T_{\Delta\cup C_{M,s}}(Y)$, such that $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} \xi$, and ξ contains a circular configuration in an outside-active node. Otherwise M is *not strongly circular.* $\diamond$

It should also be clear that a macro tree transducer (and thus also a top-down tree transducer) can be neither weakly circular, nor circular, nor strongly circular. In the rest of this subsection we compare strong circularity, circularity, and weak circularity.

Lemma 4.3 a) If M is strongly circular, then it is circular.

b) If M is circular, then it is weakly circular.

Proof. The proof of a) follows from the definition of strong circularity and circularity (Definition 4.2).

Let us prove b). If M is circular, then there is an input tree $s \in T_\Sigma$, a configuration $\langle q, h\rangle \in C_{M,s}$ and a sentential form $\xi \in T_{\Delta \cup C_{M,s}}(Y)$ such that $\langle q, h\rangle(y_1, \ldots, y_{rank(q)}) \Rightarrow^l_{M,s} \xi$ for some $l \geq 1$ and the configuration $\langle q, h\rangle$ occurs in ξ at an outside-active node. Then, obviously, $(\langle q, h\rangle, \langle q, h\rangle) \in \textit{one-step}^l \subseteq \textit{one-step}^+$, hence M is weakly circular. $\diamond$

The converse of statements a) and b) of Lemma 4.3 do not hold even for 0-ptts, as the following examples show.

Example 4.4 Let $M_1 = (\{q_0^{(0)}, q_1^{(0)}\}, \{\alpha^{(0)}, \gamma^{(1)}\}, \{\alpha^{(0)}\}, q_0, R)$ be a 0-ptt, where $R = \{\langle q_0, \alpha, 0\rangle \to \alpha, \langle q_0, \gamma, 0\rangle \to \alpha, \langle q_0, \alpha, 1\rangle \to \langle q_1, \textit{stay}\rangle, \langle q_1, \alpha, 1\rangle \to \langle q_0, \textit{stay}\rangle\}$.

It is easy to see that M_1 is deterministic and, for every input tree $s \in T_{\{\alpha,\gamma\}}$, we have $\langle q_0, \varepsilon\rangle \Rightarrow_{M_1,s} \alpha$. Hence, M_1 is not strongly circular. On the other hand, if we pick $s = \gamma(\alpha)$ to be an input tree and the configuration $\langle q_1, 1\rangle \in C_{M_1,s}$, then $\langle q_1, 1\rangle \Rightarrow^2_{M_1,s} \langle q_1, 1\rangle$ which means that M_1 is circular.

Let $M_2 = (\{q_0^{(0)}, q_1^{(1)}\}, \{\alpha^{(0)}\}, \{\alpha^{(0)}\}, q_0, R)$ be a 0-pmtt, where R consists of the following two rules:

- $\langle q_0, \alpha, 0\rangle \to \langle q_1, \textit{stay}\rangle(\langle q_0, \textit{stay}\rangle)$,
- $\langle q_1, \alpha, 0\rangle(y_1) \to \alpha$.

The only input tree to M_2 is $s = \alpha$ and there are only two configurations over s: $\langle q_0, \varepsilon\rangle$ and $\langle q_1, \varepsilon\rangle$. Moreover, since M_2 is deterministic, there is only one computation for each, namely

$$\langle q_0, \varepsilon\rangle \Rightarrow_{M_2,s} \langle q_1, \varepsilon\rangle(\langle q_0, \varepsilon\rangle) \Rightarrow_{M_2,s} \alpha \text{ and}$$

$$\langle q_1, \varepsilon\rangle(y_1) \Rightarrow_{M_2,s} \alpha.$$

Hence none of the two configurations is circular which proves that M_2 is noncircular.

On the other hand M_2 is weakly circular because, for $s = \alpha$, we have $(\langle q_0, \varepsilon\rangle, \langle q_0, \varepsilon\rangle) \in \textit{one-step}_{M_2,s}$. $\diamond$

It is easy to see that, for ptts the circularity and weak circularity concepts are equivalent.

Lemma 4.5 Let us assume that M is an n-ptt. Then M is circular if and only if it is weakly circular.

Proof. The part "$\Rightarrow$" follows from Lemma 4.3 thus we prove part "$\Leftarrow$" only.

We observe that, since M is a ptt, configurations occur only at leaves of a sentential form ζ, thus an application of a rule to a configuration does not delete other configurations from ζ. Now assume that M is weakly circular, i.e., there is an input tree $s \in T_\Sigma$ and a configuration $\langle q, h\rangle \in C_{M,s}$ with $(\langle q, h\rangle, \langle q, h\rangle) \in \textit{one-step}^+$. Then there is a sentential form ξ such that $\langle q, h\rangle \Rightarrow^+_{M,s} \xi$ and $\langle q, h\rangle$ occurs in ξ. Since $\langle q, h\rangle$ occurs at an outside-active node of ξ, see the note after Definition 3.5, we obtain that M is circular. $\diamond$

Let us note, that the circularity (and hence also weak circularity) of M may not have any effect on the computation of M. In fact, it may be that M is circular but for every input tree s, sentential form $\eta \in T_{\Delta \cup C_{M,s}}$ with $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} \eta$, and outside-active node $u \in pos(\eta)$ with $lab(\eta, u) = \langle q, h\rangle$, the configuration $\langle q, h\rangle$ is not circular. In other words, M may be circular but none of the circular configurations is accessible from the initial configuration. On the other hand, the strong circularity always has an effect on the computation of M.

We note that the circularity of attribute grammars [Knu68, Knu71] and of attributed tree transducers [Fül81, FV98] was also defined in the sense of point 1) of Definition 4.2. Informally speaking, an attribute grammar is circular if there is a derivation tree t of the underlying context-free grammar such that the dependency graph of t contains a cycle. However, if an attribute grammar is circular, then this does not mean that the computation of the value of the designated synthesized attribute at the root of t gets into that cycle.

In this paper, among others, we are interested in pmtts which are noncircular. We denote the tree transformation classes computed by not weakly circular, noncircular, and not strongly circular n-pebble (macro) tree transducers, by $n\text{-}P(M)TT_{nwc}$, $n\text{-}P(M)TT_{nc}$, and $n\text{-}P(M)TT_{nsc}$, respectively. Moreover, the prefixes d, t, and cl denote the deterministic, total, context-linear subclasses of these classes.

We will need the following proposition in Section 8.2.

Proposition 4.6 Assume that M is an n-ptt. If M is deterministic, total, and noncircular, then, for every input tree $s \in T_\Sigma$ and sentential form $\xi \in T_{\Delta \cup C_{M,s}}$, $nf(\xi, \Rightarrow_{M,s})$ exists and $nf(\xi, \Rightarrow_{M,s}) \in T_\Delta$.

Proof. (Sketch.) Let $s \in T_\Sigma$ be an input tree. The proposition follows from the following three statements.

(1) If M is deterministic, then the relation $\Rightarrow_{M,s}$ is locally confluent.

(2) If M is total, then, for every sentential form $\xi \in T_{\Delta \cup C_{M,s}}$, each normal form of ξ is in T_Δ.

(3) If M is noncircular, then the relation $\Rightarrow_{M,s}$ is terminating.

The proof of (1) is easy and analogous to the corresponding one for attributed tree transducers, cf. the proof Lemma 5.20 in [FV98]. Statement (2) follows from the fact that if M is total, then a sentential form containing a configuration cannot be a normal form. (3) can be proved by showing that the existence of an infinite sequence $\xi_1 \Rightarrow_{M,s} \xi_2 \Rightarrow_{M,s} \ldots$ would imply the existence of a circular configuration. See the corresponding proof for attributed tree transducers in Lemma 5.24 of [FV98]. $\diamond$

5 Generalized yield tree transformations

In this section we generalize the yield mappings of [Eng81, EV85]. In these papers the underlying mapping of a yield mapping is a total function, which associates with every nullary symbol a tree with variables. Here we generalize by permitting that the underlying mapping associates with every nullary symbol a finite (maybe empty) set of such trees. As a special case, we obtain partial yield tree transformations (cf. [MBPS05b]), which we will relate with 0-pmtts. The results of this section can be found in Section 5 of [FM08].

Definition 5.1 Let Σ and Δ be ranked alphabets, $m \geq 0$, and $g : \Sigma^{(0)} \to \mathcal{P}(T_\Delta(Y_m))$ a mapping. We call g *deterministic* (resp. *total*) if, for every $\alpha \in \Sigma^{(0)}$, we have $|g(\alpha)| \leq 1$ (resp. $|g(\alpha)| \geq 1$).

The tree transformation $\mathit{yield}_g : T_\Sigma \to \mathcal{P}(T_\Delta(Y_m))$ induced by g is defined as follows.

(i) For every $\alpha \in \Sigma^{(0)}$, let $\mathit{yield}_g(\alpha) = g(\alpha)$.

(ii) For every $\sigma(s_0, s_1, \ldots, s_k) \in T_\Sigma$, where $k \geq 0$, $\sigma \in \Sigma^{(k+1)}$, and $s_0, s_1, \ldots, s_k \in T_\Sigma$, let $\mathit{yield}_g(\sigma(s_0, s_1, \ldots, s_k)) = \mathit{yield}_g(s_0) \overset{OI}{\leftarrow} (\mathit{yield}_g(s_1), \ldots, \mathit{yield}_g(s_k))$. $\diamond$

If g is deterministic (total), then, for every $s \in T_\Sigma$, we have $|\mathit{yield}_g(s)| \leq 1$ ($|\mathit{yield}_g(s)| \geq 1$). A tree transformation defined in the above way is called a *yield tree transformation*. A yield tree transformation is deterministic (total) if the underlying mapping g is deterministic (total). We denote the class of yield tree transformations, and deterministic (total) yield tree transformations by *YIELD*, *dYIELD* (*tYIELD*), respectively, and combine the prefixes d and t in the usual way.

It should be clear that $\mathit{dtYIELD} \subseteq \mathit{dYIELD} \subseteq \mathit{YIELD}$ and $\mathit{dtYIELD} \subseteq \mathit{tYIELD} \subseteq \mathit{YIELD}$.

Let us note that the yield mapping introduced in Section 2.3 of [EV85] (cf. also Definition 4.30 in [FV98]) is, in our sense, a deterministic and total yield tree transformation. In that paper, the class of all yield mappings is denoted by *YIELD* (which is our *dtYIELD*). Hence we deviate from the conventional notation.

For every tree language $L \subseteq T_\Sigma$, we let $\mathit{yield}_g(L) = \bigcup\{\mathit{yield}_g(s) \mid s \in L\}$.

If a mapping $g : \Sigma^{(0)} \to \mathcal{P}(T_\Delta(Y_m))$ is deterministic, then we consider it as a partial mapping of type $\Sigma^{(0)} \to T_\Delta(Y_m)$. In this case yield_g is a partial mapping of type $T_\Sigma \to T_\Delta(Y_m)$.

Deterministic yield tree transformations can be expressed in terms of the first-order tree substitution defined by condition. It is an exercise to show that if the mapping g is deterministic, then, for every $s = \sigma(s_0, s_1, \ldots, s_k) \in T_\Sigma$ and $t \in T_\Delta(Y_m)$, the equation $\mathit{yield}_g(s) = t$ holds if and only if

- $\mathit{yield}_g(s_0) = \bar{t}$ for some $\bar{t} \in T_\Delta(Y_m)$ and
- for every $1 \leq j \leq k$, if $|\bar{t}|_{y_j} \geq 1$, then $\mathit{yield}_g(s_j) = t_j$ for some $t_j \in T_\Delta(Y_m)$ and

- $t = \overline{t}[y_j \leftarrow t_j \mid 1 \leq j \leq k$ and $|\overline{t}|_{y_j} \geq 1]$.

In Lemma 36 of [EM03] it was proved that each deterministic and total yield tree transformation $yield_g$ can be computed by a deterministic and total 0-ptt M. In the next lemma we show that the restriction total can be dropped. Moreover, we show that M is noncircular.

Theorem 5.2 (cf. Lemma 36 of [EM03]) $dYIELD \subseteq$ 0-$dPTT_{nc}$

Proof. Let Σ and Δ be ranked alphabets, $m \geq 0$, and $g : \Sigma^{(0)} \to T_\Delta(Y_m)$ a deterministic mapping. Following the proof of Lemma 36 of [EM03], we give a deterministic 0-ptt $M = (Q, \Sigma, \Gamma, q_0, R)$, and we show that for every $s \in T_\Sigma$, we have $yield_g(s) = \tau_M(s)$. For this, let

- $Q = \{q_0, q_1, \ldots, q_m, q'_1, \ldots, q'_m\}$,
- $\Gamma = \Delta \cup \{y_1^{(0)}, \ldots, y_m^{(0)}\}$,
- R is the smallest set satisfying the conditions (r1) – (r7), where $J = maxr(\Sigma)$.
  - (r1) for every $\sigma \in \Sigma^{(k)}$, $k \geq 1$ and $0 \leq j \leq J$, $\langle q_0, \sigma, j\rangle \to \langle q_0, down_1\rangle \in R$,
  - (r2) for every $\alpha \in \Sigma^{(0)}$ with $g(\alpha) \neq \emptyset$ and, for every $0 \leq j \leq J$, $\langle q_0, \alpha, j\rangle \to g(\alpha)[y_\mu \leftarrow \langle q_\mu, stay\rangle \mid \mu \in [m]] \in R$,
  - (r3) for every $\mu \in [m]$ and $\sigma \in \Sigma$, $\langle q_\mu, \sigma, 1\rangle \to \langle q'_\mu, up\rangle \in R$,
  - (r4) for every $\mu \in [m]$ and $2 \leq j \leq J$, $\langle q_\mu, \sigma, j\rangle \to \langle q_\mu, up\rangle \in R$,
  - (r5) for every $\mu \in [m]$ and $\sigma \in \Sigma$, $\langle q_\mu, \sigma, 0\rangle \to y_\mu \in R$,
  - (r6) for every $\mu \in [m]$, $k \geq \mu + 1$, $\sigma \in \Sigma^{(k)}$, and $0 \leq j \leq J$, $\langle q'_\mu, \sigma, j\rangle \to \langle q_0, down_{\mu+1}\rangle \in R$,
  - (r7) for every $\mu \in [m]$, $k < \mu + 1$, $\sigma \in \Sigma^{(k)}$, and $0 \leq j \leq J$, $\langle q'_\mu, \sigma, j\rangle \to \langle q_\mu, stay\rangle \in R$.

We note, the only difference between M and the 0-ptt given in [EM03] is that we define rules of type (r2) only for those α's which satisfy $g(\alpha) \neq \emptyset$.

It should be clear that M is deterministic.

Next we show that $yield_g = \tau_M$ by proving Statements 1 and 2.

<u>*Statement 1.*</u> For every input tree $s \in T_\Sigma$ and node $u \in pos(s)$ if $yield_g(s/u) \neq \emptyset$, then

$$\langle q_0, u\rangle \Rightarrow^*_{M,s} yield_g(s/u)[y_\mu \leftarrow \langle q_\mu, u\rangle \mid 1 \leq \mu \leq m].$$

<u>*Statement 2.*</u> For every input tree $s \in T_\Sigma$ and node $u \in pos(s)$ if $yield_g(s/u) = \emptyset$, then there exists no $t \in T_\Gamma$ such that $\langle q_0, u\rangle \Rightarrow^*_{M,s} t$.

Before proving these two statements, we show that they verify $yield_g = \tau_M$. Take an input tree $s \in T_\Sigma$. We distinguish two cases. First, let us assume that $yield_g(s) \neq \emptyset$. Then, by Statement 1 for $u = \varepsilon$, we get

$$\langle q_0, \varepsilon \rangle \Rightarrow^*_{M,s} \mathit{yield}_g(s)[y_\mu \leftarrow \langle q_\mu, \varepsilon \rangle \mid \mu \in [m]].$$

Moreover, applying rules of type (r5), we have

$$\mathit{yield}_g(s)[y_\mu \leftarrow \langle q_\mu, \varepsilon \rangle \mid \mu \in [m]] \Rightarrow^*_{M,s} \mathit{yield}_g(s),$$

hence $\tau_M(s) = \mathit{yield}_g(s)$.

Now assume $\mathit{yield}_g(s) = \emptyset$. Then, by Statement 2 for $u = \varepsilon$, there is no $t \in T_\Gamma$ such that $\langle q_0, \varepsilon \rangle \Rightarrow^*_{M,s} t$, which means $\tau_M(s) = \emptyset = \mathit{yield}_g(s)$.

Next we prove Statements 1 and 2.

Proof of Statement 1. The proof is the same as that of the corresponding part of Lemma 36 in [EM03]. In that lemma the condition $\mathit{yield}_g(s/u) \neq \emptyset$ holds because g is a total mapping.

Proof of Statement 2. We apply induction on the structure of the tree s/u.

(i) Let $s/u = \alpha \in \Sigma^{(0)}$. Since $\mathit{yield}_g(s/u) = \emptyset$, we have $g(\alpha) = \emptyset$. Hence, due to the definition of rules (r2) of M, there is no computation step from the configuration $\langle q_0, u \rangle$, which proves Statement 2.

(ii) Let $s/u = \sigma(s_0, \ldots, s_k)$ for some $k \geq 0$, $\sigma \in \Sigma^{(k+1)}$ and $s_0, s_1, \ldots, s_k \in T_\Sigma$. Then, by the characterization of deterministic yield mappings with first-order tree substitution, there are two cases.

Case 1: $\mathit{yield}_g(s_0) = \emptyset$. Then, due to rules (r1), we get

$$\langle q_0, u \rangle \Rightarrow_{M,s} \langle q_0, u1 \rangle.$$

Moreover, by the induction hypothesis, there exists no $t \in T_\Gamma$ such that $\langle q_0, u1 \rangle \Rightarrow^*_{M,s} t$. Since M is deterministic, there is no $t \in T_\Gamma$ such that $\langle q_0, u \rangle \Rightarrow^*_{M,s} t$.

Case 2: $\mathit{yield}_g(s_0) \neq \emptyset$, however there is a $1 \leq j \leq k$ such that $|\mathit{yield}_g(s_0)|_{y_j} \geq 1$ and $\mathit{yield}_g(s_j) = \emptyset$.

Now, due to rules (r1) and Statement 1, we obtain

$$\langle q_0, u \rangle \Rightarrow_{M,s} \langle q_0, u1 \rangle \Rightarrow^*_{M,s} \mathit{yield}_g(s_0)[y_\mu \leftarrow \langle q_\mu, u1 \rangle \mid \mu \in [m]] = \xi.$$

Since $|\mathit{yield}_g(s_0)|_{y_j} \geq 1$, the configuration $\langle q_j, u1 \rangle$ occurs in ξ. Moreover,

$$\begin{array}{lll} \langle q_j, u1 \rangle \Rightarrow_{M,s} & & \\ \quad \Rightarrow_{M,s} & \langle q'_j, u \rangle & \text{(rule (r3))} \\ \quad \Rightarrow_{M,s} & \langle q_0, u(j+1) \rangle & \text{(rule (r6)).} \end{array}$$

By the induction hypothesis, there exists no $t' \in T_\Gamma$ for which $\langle q_0, u(j+1) \rangle \Rightarrow^*_{M,s} t'$. Consequently, there exists no $t \in T_\Gamma$ such that $\xi \Rightarrow^*_{M,s} t$. Since M is deterministic, this finishes the proof of Statement 2.

Finally, we show that M is noncircular.

First ,,we make M total" by adding some dummy rules to R. More exactly, we introduce the 0-ptt $M' = (Q, \Sigma, \Gamma', q_0, R')$, where

- Q, Σ, and q_0 are the same as in M,

- $\Gamma' = \Gamma \cup \{nil^{(0)}\}$, where nil is a new nullary symbol,
- $R' = R \cup \{\langle q_0, \alpha, j\rangle \to nil \mid \alpha \in \Sigma^{(0)},\, 0 \leq j \leq J, rhs_M(q_0, \alpha, j) = \emptyset\}$.

It should be clear that M' is deterministic and total and that M' is noncircular if and only if M has this property. This equivalence follows from that the rules we added cannot cause a cycle in any computation because their right-hand side is a nullary symbol.

Hence it is sufficient to show that M' is noncircular. Therefore, we show that, for every input tree $s \in T_\Sigma$, $u \in pos(s)$, and $p \in Q$, the configuration $\langle p, u\rangle \in C_{M',s}$ is not circular because there exists an output tree $t \in T_{\Gamma'}$ such that $\langle p, u\rangle \Rightarrow^*_{M',s} t$. This will follow from Statements 4, 5, and 6 proved below. Statement 3 will be needed in the proof of Statements 4 and 5.

As a preparation, we define the set $L(u)$ of sentential forms as

$$L(u) = T_{\Gamma' \cup \{\langle q_\mu, u\rangle \mid \mu \in [m]\}}.$$

Statement 3. For every configuration $\langle q_0, u\rangle \in C_{M',s}$ there is a sentential form $\xi \in L(u)$ such that $\langle q_0, u\rangle \Rightarrow^*_{M',s} \xi$.

Proof of Statement 3. The proof is the same as the proof of the Claim of Lemma 36. of [EM03], since M' computes a total and deterministic yield tree transformation.

Statement 4. For every $\mu \in [m]$ and configuration $\langle q_\mu, u\rangle \in C_{M',s}$, there is an output tree $t \in T_{\Gamma'}$ such that $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} t$.

Proof of Statement 4. Induction on the length of u.

(i) If $u = \varepsilon$, then, by rules of type (r5), $\langle q_\mu, \varepsilon\rangle \Rightarrow_{M',s} y_\mu \in T_{\Gamma'}$.

(ii) Let $u = u'j$, for some $u' \in pos(s)$ and $1 \leq j \leq J$. We distinguish two cases.

Case 1: $j \geq 2$. Then, by rules of type (r4), $\langle q_\mu, u\rangle \Rightarrow_{M',s} \langle q_\mu, u'\rangle$. Moreover, by the induction hypothesis, there is a $t \in T_{\Gamma'}$ such that $\langle q_\mu, u'\rangle \Rightarrow^*_{M',s} t$. Thus we obtain $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} t$.

Case 2: $j = 1$. Then, by rules of type (r3), $\langle q_\mu, u\rangle \Rightarrow_{M',s} \langle q'_\mu, u'\rangle$ and we continue the proof by distinguishing two subcases.

The first subcase is when

$\langle q'_\mu, u'\rangle \Rightarrow_{M',s} \langle q_\mu, u'\rangle$ with rules of type (r7).

Then, by the induction hypothesis, there is a tree $t \in T_{\Gamma'}$ such that $\langle q_\mu, u'\rangle \Rightarrow^*_{M',s} t$, hence also $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} t$.

The second one is when

$\langle q'_\mu, u'\rangle \Rightarrow_{M',s} \langle q_0, u'(\mu+1)\rangle$ with rules of type (r6).

Now, by Statement 3, there is a sentential form $\xi \in L(u'(\mu+1))$ such that $\langle q_0, u'(\mu+1)\rangle \Rightarrow^*_{M',s} \xi$, i.e., such that $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} \xi$. Moreover, by Case 1, for every configuration $\langle q_\nu, u'(\mu+1)\rangle \in C_{M',s}$ which occurs in ξ, there is a tree $t' \in T_{\Gamma'}$ such that $\langle q_\nu, u'(\mu+1)\rangle \Rightarrow^*_{M',s} t'$. Consequently, there is a tree $t \in T_{\Gamma'}$ such that $\xi \Rightarrow^*_{M',s} t$. For this tree t

also $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} t$. This finishes the proof of Statement 4.

Statement 5. For every configuration $\langle q_0, u\rangle \in C_{M',s}$ there is a tree $t \in T_{\Gamma'}$ such that $\langle q_0, u\rangle \Rightarrow^*_{M',s} t$.

Proof of Statement 5. By Statement 3, there is a sentential form $\xi \in L(u)$ such that $\langle q_0, u\rangle \Rightarrow^*_{M',s} \xi$. Moreover, by Statement 4, for every configuration $\langle q_\mu, u\rangle \in C_{M',s}$ occurring in ξ, there is a tree $t' \in T_{\Gamma'}$ such that $\langle q_\mu, u\rangle \Rightarrow^*_{M',s} t'$. This implies that there exists a tree $t \in T_{\Gamma'}$ such that $\xi \Rightarrow^*_{M',s} t$. For this tree t also $\langle q_0, u\rangle \Rightarrow^*_{M',s} t$.

Statement 6. For every $\mu \in [m]$ and configuration $\langle q'_\mu, u\rangle \in C_{M',s}$, there is a tree $t \in T_{\Gamma'}$ such that $\langle q'_\mu, u\rangle \Rightarrow^*_{M',s} t$.

Proof of Statement 6. Then, either

$\langle q'_\mu, u\rangle \Rightarrow_{M',s} \langle q_0, u(\mu+1)\rangle$ with rules of type $(r6)$ or

$\langle q'_\mu, u\rangle \Rightarrow_{M',s} \langle q_\mu, u\rangle$ with rules of type $(r7)$.

In the first case it follows from Statement 5, in the second one it follows from Statement 4 that there is a tree $t \in T_{\Gamma'}$ such that $\langle q'_\mu, u\rangle \Rightarrow^*_{M',s} t$.

This finishes the proof of our lemma. $\diamond$

Later on we will use the following version of the inclusion result of Lemma 36 of [EM03] and Theorem 5.2.

Theorem 5.3 *YIELD* $\subseteq$ 0-*PTT*

Proof. The construction of the 0-ptt and the proof of its correctness are similar to the corresponding part of the proof of Theorem 5.2. $\diamond$

Since we use Theorem 5.3 in decompositions of arbitrary (maybe circular) pmtts, we do not consider circularity issues here. We just note that the proof of that M of Theorem 5.2 is deterministic does not apply here because in that proof we strongly used the fact that M is deterministic.

6 Composition of pebble and yield tree transformations

In this section we will prove that the composition of a pebble tree transformation and a yield tree transformation can be computed by a pebble macro tree transducer. The idea behind the construction is that we apply the yield tree transformation to the right-hand sides of the rules of the pebble tree transducer. Results of this section can be found in Section 7 of [FM08]. We note that the pebble macro tree transducer model of [FM08] works with weak pebble handling. However, this fact is not used there in the proof of the following lemma. Hence the composition result works also for pebble macro tree transducers of this thesis.

Lemma 6.1 For all $n \geq 0$ we have $n\text{-}PTT \circ YIELD \subseteq n\text{-}PMTT$.

Proof. Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be an n-ptt. Let Δ be a ranked alphabet, $m \geq 0$, and $g : \Gamma^{(0)} \to \mathcal{P}(T_\Delta(Y_m))$ a mapping. We construct an n-pebble macro tree transducer M', such that $\tau_M \circ yield_g = \tau_{M'}$. For this, let $M' = (Q', \Sigma, \Delta \cup \{y_1, \dots, y_m\}, q_{in}, R')$, where Q', q_{in}, and R' are given as follows.

Let $d = maxr(\Sigma)$.

- $Q' = \{q' \mid q \in Q\} \cup \{q_{in}\}$, where for every $q \in Q$, the state q' has rank m and q_{in} has rank 0.
- We extend g to $g' : \Gamma^{(0)} \cup \langle Q, I_d\rangle \to \mathcal{P}(T_{\Delta\cup\langle Q',I_d\rangle}(Y_m))$ by defining $g'(\langle q,\varphi\rangle) = \{\langle q',\varphi\rangle(y_1,\dots,y_m)\}$, for every state-instruction pair $\langle q,\varphi\rangle \in \langle Q, I_d\rangle$.
- Let R' be the smallest set of rules satisfying the following conditions.
  - For every $\sigma \in \Sigma$ the rule $\langle q_{in}, \sigma, \varepsilon, 0\rangle \to \langle q_0', stay\rangle(y_1,\dots,y_m)$ is in R'.
  - For every rule $\langle q, \sigma, b, j\rangle \to \zeta$ of R and $\zeta' \in yield_{g'}(\zeta)$, the rule $\langle q', \sigma, b, j\rangle(y_1,\dots,y_m) \to \zeta'$ is in R'.

In the rest of this proof let $s \in T_\Sigma$ be an input tree. For every pebble configuration $h \in IC_{M,s}$ with $test(h) = (\sigma, b, j)$, we introduce the following abbreviations for second-order substitutions:

$$[\![\dots]\!]_h = [\![\langle q,\varphi\rangle \leftarrow \langle q, \varphi(h)\rangle \mid q \in Q, \varphi \in I_{\sigma,b,j}]\!]$$

and

$$[\![\dots]\!]'_h = [\![\langle q',\varphi\rangle \leftarrow \langle q', \varphi(h)\rangle(y_1,\dots,y_m) \mid q' \in Q' - \{q_{in}\}, \varphi \in I_{\sigma,b,j}]\!].$$

The correctness of the construction is verified by proving the following two statements.

<u>*Statement 1:*</u>

K(l): For every $\langle q, h\rangle \in C_{M,s}$ and $t \in T_\Gamma$, if $\langle q,h\rangle \Rightarrow^l_{M,s} t$, then for each $t' \in yield_{g'}(t)$ we have $\langle q', h\rangle(y_1,\dots,y_m) \Rightarrow^*_{M',s} t'$.

L(l): For every $h \in IC_{M,s}$, where $test(h) = (\sigma, b, j)$, $\zeta \in T_{\Gamma \cup \langle Q, I_{\sigma,b,j}\rangle}$, and $t \in T_\Gamma$, if $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$, then for all $t' \in yield_{g'}(t)$, there is a $\zeta' \in yield_{g'}(\zeta)$ such that $\zeta'[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$.

<u>*Statement 2:*</u>

K(l): For every $\langle q', h\rangle \in C_{M',s}$ and $t' \in T_\Delta$, if $\langle q', h\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M',s} t'$, then there is a tree $t \in T_\Gamma$, where $\langle q, h\rangle \Rightarrow^*_{M,s} t$, and $t' \in yield_{g'}(t)$.

L(l): For every $h \in IC_{M,s}$, where $test(h) = (\sigma, b, j)$, $\zeta \in T_{\Gamma \cup \langle Q, I_{\sigma,b,j}\rangle}$, $\zeta' \in yield_{g'}(\zeta)$, and $t' \in T_\Delta(Y_m)$, if $\zeta'[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$, then there is a tree $t \in T_\Gamma$, such that $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$ and $t' \in yield_{g'}(t)$.

We note that for each tree $t \in T_\Gamma$ we have $yield_{g'}(t) = yield_g(t)$.

The inclusion $\tau_M \circ yield_g \subseteq \tau_{M'}$ follows from Statement 1 due to the following. Let $(s, t') \in \tau_M \circ yield_g$. Then there is a tree $t \in T_\Gamma$ such that $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^l_{M,s} t$ for some $l \geq 1$ (meaning that $(s, t) \in \tau_M$) and $t' \in yield_g(t)$. By predicate K(l) of Statement 1 we obtain that $\langle q'_0, (\varepsilon, [\,])\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M',s} t'$, moreover, since $\langle q_{in}, (\varepsilon, [\,])\rangle \Rightarrow_{M',s} \langle q'_0, (\varepsilon, [\,])\rangle(y_1, \ldots, y_m)$, we also get that $\langle q_{in}, (\varepsilon, [\,])\rangle \Rightarrow^*_{M',s} t'$, hence $(s, t') \in \tau_{M'}$.

The inclusion $\tau_{M'} \subseteq \tau_M \circ yield_g$ follows from Statement 2 due to the following. Let $(s, t') \in \tau_{M'}$. Then $\langle q_{in}, (\varepsilon, [\,])\rangle \Rightarrow_{M',s} \langle q'_0, (\varepsilon, [\,])\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M',s} t'$ for some $l \geq 1$. Hence, by predicate K(l) of Statement 2 we obtain that there is a tree $t \in T_\Gamma$ such that $t' \in yield_g(t)$ and $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} t$, which means that $(s, t') \in \tau_M \circ yield_g$.

Now we have left to prove Statements 1 and 2.

<u>*Proof of Statement 1:*</u>

IB: (proof of L(0))

Let us assume that $\zeta[\![\ldots]\!]_h \Rightarrow^0_{M,s} t$. It follows $t = \zeta$, hence $yield_{g'}(\zeta) = yield_{g'}(t)$. Thus for each $t' \in yield_{g'}(\zeta)$ we can pick $\zeta' = t'$ which gives the transition $\zeta'[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$.

IS1: (proof of L[l] $\Rightarrow$ K($l+1$))

Assume that $\langle q, h\rangle \Rightarrow^{l+1}_{M,s} t$ holds. Then there exists a sentential form $\xi \in T_{\Gamma \cup C_{M,s}}$ such that $\langle q, h\rangle \Rightarrow_{M,s} \xi \Rightarrow^l_{M,s} t$. Thus there is a rule $\langle q, \sigma, b, j\rangle \to \zeta \in R$, where $test(h) = (\sigma, b, j)$ and $\zeta[\![\ldots]\!]_h = \xi$.

Let $t' \in yield_{g'}(t)$ be an arbitrary tree. By the induction hypothesis L(l), there is a tree $\zeta' \in yield_{g'}(\zeta)$, such that $\zeta'[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$.

Moreover, by the definition of M', the rule $\langle q', \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta'$ is in R', hence $\langle q', h\rangle(y_1, \ldots, y_m) \Rightarrow_{M',s} \zeta'[\![\ldots]\!]'_h$, which concludes that $\langle q', h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M',s} t'$.

IS2: (proof of K[l] $\Rightarrow$ L(l))

Assume that $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$. We continue the proof by induction on ζ. The case $\zeta = \alpha \in \Gamma^{(0)}$ is not possible because $l \geq 1$. Hence we have cases (i) and (ii).

(i) If $\zeta = \langle q, \varphi\rangle \in \langle Q, I_d\rangle$, then $yield_{g'}(\zeta)$ is singleton and is equal to $\langle q', \varphi\rangle(y_1, \ldots, y_m) = \zeta'$. By the induction hypothesis K[l] we get that $\zeta'[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$ for all $t' \in yield_{g'}(t)$.

(ii) Let $\zeta = \delta(\zeta_0, \ldots, \zeta_k)$ for some $k \geq 0$, $\delta \in \Gamma^{(k)}$, and $\zeta_0, \ldots, \zeta_k \in T_{\Gamma \cup \langle Q, I_{\sigma,b,j}\rangle}$.

We make the following observations.

a) There are integers $l_0, \ldots, l_k \leq l$ and output trees $t_0, \ldots, t_k \in T_\Gamma$ such that

$$\zeta_0[\![\ldots]\!]_h \Rightarrow^{l_0}_{M,s} t_0, \ldots, \zeta_k[\![\ldots]\!]_h \Rightarrow^{l_k}_{M,s} t_k$$

and $t = \delta(t_0, \ldots, t_k)$.

b) Let $t' \in yield_{g'}(t)$. By Definition 5.1

$$t' = t'_0[y_1 \leftarrow t'_1, \ldots, y_k \leftarrow t'_k],$$

where $t'_0 \in yield_{g'}(t_0), \ldots, t'_k \in yield_{g'}(t_k)$.

c) By the induction hypothesis on ζ, there are trees $\zeta'_0 \in yield_{g'}(\zeta_0), \ldots, \zeta'_k \in yield_{g'}(\zeta_k)$, such that

$$\zeta'_0[\![\ldots]\!]'_h, \Rightarrow^*_{M',s} t'_0, \ldots, \zeta'_k[\![\ldots]\!]'_h, \Rightarrow^*_{M',s} t'_k.$$

d) Let us pick ζ' to be the tree $\zeta'_0[y_1 \leftarrow \zeta'_1, \ldots, y_k \leftarrow \zeta'_k]$

e) By Definition 5.1 $\zeta' \in yield_{g'}(\zeta)$.

f) Hence,

$$\begin{array}{rcl} \zeta'[\![\ldots]\!]'_h & = & \zeta'_0[\![\ldots]\!]'_h[y_1 \leftarrow \zeta'_1[\![\ldots]\!]'_h, \ldots, y_k \leftarrow \zeta'_k[\![\ldots]\!]'_h] \\ & \Rightarrow^*_{M',s} & t'_0[y_1 \leftarrow \zeta'_1[\![\ldots]\!]'_h, \ldots, y_k \leftarrow \zeta'_k[\![\ldots]\!]'_h] \\ & \Rightarrow^*_{M',s} & t'_0[y_1 \leftarrow t'_1, \ldots, y_k \leftarrow t'_k] \\ & = & t'. \end{array}$$

This ends the proof of Statement 1.

Proof of Statement 2:

IB: (proof of L(0))

Assume that $\zeta'[\![\ldots]\!]'_h \Rightarrow^0_{M',s} t'$. Then obviously $\zeta' = t'$. Let us choose t to be equal to ζ. Then it follows easily that $t' \in yield_{g'}(t)$, and $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$.

IS1: (proof of L[l] $\Rightarrow$ K($l+1$))

Assume that $\langle q', h\rangle(y_1, \ldots, y_m) \Rightarrow^{l+1}_{M',s} t'$. Then there is a sentential form $\xi' \in T_{\Delta \cup C_{M's}}(Y_m)$, where $\langle q', h\rangle(y_1, \ldots, y_m) \Rightarrow_{M',s} \xi' \Rightarrow^l_{M',s} t'$. Hence, there is a rule $\langle q', \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta' \in R'$, such that $test(h) = (\sigma, b, j)$, and $\xi' = \zeta'[\![\ldots]\!]'_h$. By the construction of M', there is a rule $\langle q, \sigma, b, j\rangle \to \zeta \in R$, such that $\zeta' \in yield_{g'}(\zeta)$. These prove the following.

- By the induction hypothesis L(l), there is a tree $t \in T_\Gamma$ such that $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$ and $t' \in yield_{g'}(t)$.
- $\langle q, h\rangle \Rightarrow_{M,s} \zeta[\![\ldots]\!]_h$.

Thus we obtain that there is a tree $t \in T_\Gamma$ such that $\langle q, h\rangle \Rightarrow^*_{M,s} t$ and $t' \in \mathit{yield}_{g'}(t)$.

IS2: (proof of K[l] $\Rightarrow$ L(l))

Assume that $\zeta' [\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$. (Recall that $\zeta' \in \mathit{yield}_{g'}(\zeta)$ for a tree $\zeta \in T_{\Gamma \cup \langle Q, I_{\sigma,b,j}\rangle}$ and $t' \in T_\Delta(Y_m)$.) We prove by induction on ζ. The case $\zeta = \alpha \in \Gamma^{(0)}$ is not possible because $l \geq 1$. Hence we have cases (i) and (ii).

(i) Let $\zeta = \langle q, \varphi\rangle \in \langle Q, I_d\rangle$. Then $\mathit{yield}_{g'}(\zeta)$ is singleton and ζ' can only be $\langle q', \varphi\rangle(y_1, \ldots, y_m)$. Thus, by the induction hypothesis K[l], there is a tree $t \in T_\Gamma$, such that $\langle q, \varphi\rangle [\![\ldots]\!]_h \Rightarrow^*_{M,s} t$, and $t' \in \mathit{yield}_{g'}(t)$.

(ii) Let $\zeta = \delta(\zeta_0, \ldots, \zeta_k)$, for $k \geq 0$ and $\zeta_0, \ldots, \zeta_k \in T_{\Gamma \cup \langle Q, I_{\sigma,b,j}\rangle}$. We make the following observations.

a) By Definition 5.1 $\zeta' = \zeta'_0[y_1 \leftarrow \zeta'_1, \ldots, y_k \leftarrow \zeta'_k]$, where $\zeta'_0 \in \mathit{yield}_{g'}(\zeta_0), \ldots, \zeta'_k \in \mathit{yield}_{g'}(\zeta_k)$

b) It follows from the derivation $\zeta' [\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ and a) that there are numbers $l_0, \ldots, l_k \leq l$, $t'_0, \ldots, t'_k \in T_\Delta(Y_m)$ such that $\zeta'_0 [\![\ldots]\!]'_h \Rightarrow^{l_0}_{M',s} t'_0, \ldots, \zeta'_k [\![\ldots]\!]'_h \Rightarrow^{l_k}_{M',s} t'_k$, moreover $t' = t'_0[y_1 \leftarrow t'_1, \ldots, y_k \leftarrow t'_k]$.

c) By the induction hypothesis on ζ there are trees $t_0, \ldots, t_k \in T_\Gamma$ where $\zeta_0 [\![\ldots]\!]_h \Rightarrow^*_{M,s} t_0, \ldots, \zeta_k [\![\ldots]\!]_h \Rightarrow^*_{M,s} t_k$ and $t'_0 \in \mathit{yield}_{g'}(t_0), \ldots, t'_k \in \mathit{yield}_{g'}(t_k)$.

Let us choose $t = \delta(t_0, \ldots, t_k)$. We are left to prove that $t' \in \mathit{yield}_{g'}(t)$, and $\zeta [\![\ldots]\!]_h \Rightarrow^*_{M,s} t$. Hence

$$\begin{array}{rcl} t' & = & t'_0[y_1 \leftarrow t'_1, \ldots, y_k \leftarrow t'_k] \\ & \in & t'_0 \overset{OI}{\leftarrow} (\mathit{yield}_{g'}(t_1), \ldots, \mathit{yield}_{g'}(t_k)) \\ & \subseteq & \mathit{yield}_{g'}(t_0) \overset{OI}{\leftarrow} (\mathit{yield}_{g'}(t_1), \ldots, \mathit{yield}_{g'}(t_k)) \\ & = & \mathit{yield}_{g'}(t). \end{array}$$

On the other hand

$$\begin{array}{rcl} \zeta [\![\ldots]\!]_h & = & \delta(\zeta_0 [\![\ldots]\!]_h, \ldots, \zeta_k [\![\ldots]\!]_h) \\ & \Rightarrow^*_{M,s} & \delta(t_0 \ldots, t_k) \\ & = & t. \end{array}$$

This completes the proof of Statement 2 and this lemma. $\diamond$

Let us observe that the construction of M' from M in the proof of Lemma 6.1 preserves noncircularity and in case yield_g is deterministic, the construction also preserves determinism. (On the contrary, M' is not total, even if M and g are total, since there is no computation from state q_{in} if the pointer points to a non-root node or there are pebbles on the input tree.) Moreover, if M is a top-down tree transducer, then M' is a macro tree transducer. Thus we obtain the following corollaries.

Corollary 6.2 For each $n \geq 0$ we have

- $n\text{-}dPTT \circ dYIELD \subseteq n\text{-}dPMTT$,
- $n\text{-}PTT_{nc} \circ YIELD \subseteq n\text{-}PMTT_{nc}$,

- $n\text{-}dPTT_{nc} \circ dYIELD \subseteq n\text{-}dPMTT_{nc}$,
- $T \circ YIELD \subseteq MTT$, and
- $dT \circ dYIELD \subseteq dMTT$. $\diamond$

7 Decomposition and characterization results for pmtts

In this section, if not specified otherwise, $M = (Q, \Sigma, \Delta, q_0, R)$ stands for an arbitrary n-pmtt. We associate with M a total n-ptt M' and a yield tree transformation yield_g. Then we prove that $\tau_M = \tau_{M'} \circ \mathit{yield}_g$ holds. Using Lemma 6.1 and this decomposition, we give a characterization of tree transformations computed by n-pmtts in terms of the compositions of tree transformations computed by n-ptts and yield transformations. The results of this section can be found in Section 4 of [FM09].

7.1 Associating an n-ptt and a yield tree transformation with an n-pmtt

Definition 7.1 Let $d = 1 + \max\{\mathit{maxr}(Q), \mathit{maxr}(\Sigma), \mathit{maxr}(\Delta), 2\}$. The n-ptt *associated with* M is $M' = (Q', \Sigma, \Gamma, q'_0, R')$, where

- $Q' = Q'^{(0)} = \{q' \mid q \in Q\}$,
- Γ is the smallest ranked alphabet defined by
  - $\Gamma^{(0)} = \{\delta' \mid \delta \in \Delta\} \cup \{\alpha_1, \ldots, \alpha_d\} \cup \{\mathit{nil}\} \cup \{\#\}$, where δ', $\alpha_1, \ldots, \alpha_d$, nil, and $\#$ are new symbols,
  - $\Gamma^{(i)} = \{c_i\}$ for every $1 \leq i \leq d$.
- In order to give the rule set R', we need the mapping $\mathit{comb} : T_{\Delta \cup \langle Q, I_d \rangle}(Y_d) \to T_{\Gamma \cup \langle Q', I_d \rangle}$, cf. Lemma 5.5 in [EV85] and also Lemma 4.34 in [FV98]. For each $\zeta \in T_{\Delta \cup \langle Q, I_d \rangle}(Y_d)$, the tree $\mathit{comb}(\zeta)$ is defined by induction on ζ as follows.

  (i) If $\zeta = y_i$ for some $1 \leq i \leq d$, then $\mathit{comb}(\zeta) = \alpha_i$.

  (ii) Assume that $\zeta = \langle q, \varphi \rangle(\zeta_1, \ldots, \zeta_k)$, where $k \geq 0$, $q \in Q^{(k)}$, $\varphi \in I_d$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_d \rangle}(Y_d)$. Then

  $$\mathit{comb}(\zeta) = c_{k+1}(\langle q', \varphi \rangle, \mathit{comb}(\zeta_1), \ldots, \mathit{comb}(\zeta_k)).$$

  (iii) Assume that $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$, where $k \geq 0$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_d \rangle}(Y_d)$. Then

  $$\mathit{comb}(\zeta) = c_{k+1}(\delta', \mathit{comb}(\zeta_1), \ldots, \mathit{comb}(\zeta_k)).$$

  In Fig. 3 we give an example of $\mathit{comb}(\zeta)$.

  Now, for each $q \in Q, \sigma \in \Sigma, b \in \{0,1\}^{\leq n}$, and $0 \leq j \leq \mathit{maxr}(\Sigma)$, let

  $$\begin{array}{rcll} R_{q,\sigma,b,j} & = & \{\langle q', \sigma, b, j \rangle \to \mathit{nil}\} & (1) \\ & \cup & \{\langle q', \sigma, b, j \rangle \to \mathit{comb}(\zeta) \mid \zeta \in \mathit{rhs}_M(q, \sigma, b, j)\} & (2) \\ & \cup & \{\langle q', \sigma, b, j \rangle \to c_3(\#, \langle q', \mathit{stay} \rangle, \langle q', \mathit{stay} \rangle)\} & (3) \end{array}$$

  Then we define

  $$R' = \bigcup_{\substack{q \in Q, \sigma \in \Sigma, b \in \{0,1\}^{\leq n}, \\ 0 \leq j \leq \mathit{maxr}(\Sigma)}} R_{q,\sigma,b,j}.$$

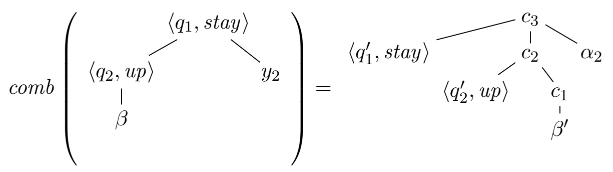


Figure 3: An example of $comb(\zeta)$.

Let $yield_g$ be the tree transformation induced by the mapping $g : \Gamma^{(0)} \to \mathcal{P}(T_\Delta(Y_d))$ defined in the following way.

- For every $k \geq 0$, $\delta \in \Delta^{(k)}$, let $g(\delta') = \{\delta(y_1, \ldots, y_k)\}$,
- for every $1 \leq i \leq d$, let $g(\alpha_i) = \{y_i\}$,
- let $g(\#) = \{y_1, y_2\}$, and
- let $g(nil) = \emptyset$. $\diamond$

Next we demonstrate the construction applied in Definition 7.1.

Example 7.2 Consider M_2 of Example 3.7. Then

$$M_2' = (\underbrace{\{q_0', q_1', q_2', q_3', q_\alpha', q_\beta'\}}_{Q'}, \underbrace{\{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\}}_{\Sigma}, \underbrace{\{c_1^{(1)}, c_2^{(2)}, c_3^{(3)}, \sigma'^{(0)}, \alpha'^{(0)}\}}_{\Gamma}, q_0', R'),$$

where R' contains the rules (1) and (3) of Definition 7.1 as well as the following rules:

- $\langle q_0', \sigma, \varepsilon, j\rangle \to c_1(\langle q_0', down_2\rangle)$ $(j \in \{0,1,2\})$
- $\langle q_0', \alpha, \varepsilon, 2\rangle \to c_1(\langle q_\alpha', up\rangle)$
- $\langle q_0', \beta, \varepsilon, 2\rangle \to c_1(\langle q_\beta', up\rangle)$
- $\langle q_\alpha', \sigma, \varepsilon, 2\rangle \to c_1(\langle q_\alpha', up\rangle)$
- $\langle q_\alpha', \sigma, \varepsilon, j\rangle \to c_1(\langle q_1', drop\rangle)$ $(j \in \{0,1\})$
- $\langle q_\beta', \sigma, \varepsilon, 2\rangle \to c_1(\langle q_\beta', up\rangle)$
- $\langle q_\beta', \sigma, \varepsilon, j\rangle \to c_1(\langle q_0', down_1\rangle)$ $(j \in \{0,1\})$
- $\langle q_0', \alpha, \varepsilon, j\rangle \to c_2(\langle q_{bin}', stay\rangle, c_1(\alpha'))$ $(j \in \{0,1\})$
- $\langle q_1', \sigma, 1, j\rangle \to c_1(\langle q_1', down_1\rangle)$ $(j \in \{0,1\})$
- $\langle q_1', \sigma, 0, j\rangle \to c_1(\langle q_1', down_2\rangle)$ $(j \in \{1,2\})$
- $\langle q_1', \alpha, 0, 2\rangle \to c_1(\langle q_\alpha', up\rangle)$
- $\langle q_1', \beta, 0, 2\rangle \to c_1(\langle q_\beta', up\rangle)$

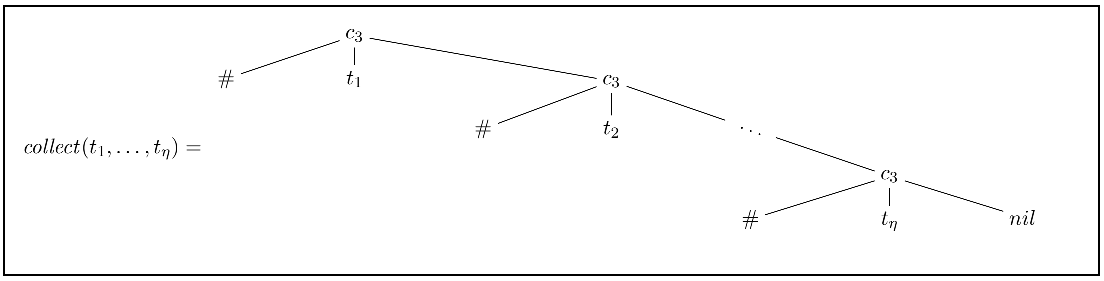


Figure 4: The tree $collect(t_1, \ldots, t_\eta)$ for some trees $t_1, \ldots, t_\eta \in T_\Gamma$.

- $\langle q'_\alpha, \sigma, 0, 2\rangle \to c_1(\langle q'_\alpha, up\rangle)$
- $\langle q'_\alpha, \sigma, 0, 1\rangle \to c_1(\langle q'_2, lift\rangle)$
- $\langle q'_2, \sigma, \varepsilon, 1\rangle \to c_1(\langle q'_1, drop\rangle)$
- $\langle q'_\beta, \sigma, 0, 2\rangle \to c_1(\langle q'_\beta, up\rangle)$
- $\langle q_\beta, \sigma, 0, 1\rangle \to c_1(\langle q'_1, down_1\rangle)$
- $\langle q'_1, \alpha, 0, 1\rangle \to c_2(\langle q'_{bin}, lift\rangle, c_1(\alpha'))$
- $\langle q'_1, \beta, 0, 1\rangle \to c_1(\langle q'_3, up\rangle)$
- $\langle q'_3, \sigma, 0, 1\rangle \to c_1(\langle q'_3, up\rangle)$
- $\langle q'_3, \sigma, 1, 1\rangle \to c_2(\langle q'_{bin}, lift\rangle, c_1(\alpha'))$
- $\langle q'_{bin}, \sigma, \varepsilon, j\rangle \to c_2(\langle q'_{bin}, down_1\rangle, c_2(\langle q'_{bin}, down_2\rangle, \alpha_1))$ $(j \in \{0, 1, 2\})$
- $\langle q'_{bin}, \alpha, \varepsilon, j\rangle \to c_3(\sigma', \alpha_1, \alpha_1)$ $(j \in \{0, 1, 2\})$
- $\langle q'_{bin}, \beta, \varepsilon, j\rangle \to c_3(\sigma, \alpha_1, \alpha_1)$ $(j \in \{0, 1, 2\})$

Moreover, $g(\sigma') = \sigma(y_1, y_2)$, $g(\alpha') = \alpha$, and $g(\alpha_1) = y_1$. $\diamond$

Now we introduce a technical notation, the concept of "collecting" trees of T_Γ, which will be used in Lemma 7.7. In fact, we collect finitely many trees of T_Γ in a certain manner in order to produce a new tree in T_Γ.

Definition 7.3 Let $t_1, \ldots, t_\eta \in T_\Gamma$ for some $\eta \geq 0$. Then $collect(t_1, \ldots, t_\eta)$ is a tree in T_Γ, defined by induction on η as follows.

(i) If $\eta = 0$ then $collect() = nil$.

(ii) If $\eta > 0$ then $collect(t_1, \ldots, t_\eta) = c_3(\#, t_1, collect(t_2, \ldots, t_\eta))$. $\diamond$

Fig. 4 illustrates $collect(t_1, \ldots, t_\eta)$. The following two lemmas show useful properties of collecting trees.

Lemma 7.4 Let $t_1, \ldots, t_\eta \in T_\Gamma$ for some $\eta \geq 0$. Then $\mathit{yield}_g(\mathit{collect}(t_1, \ldots, t_\eta)) = \mathit{yield}_g(t_1) \cup \ldots \cup \mathit{yield}_g(t_\eta)$.

Proof. We prove by induction on η.

(i) If $\eta = 0$ then obviously, $\mathit{yield}_g(\mathit{collect}(t_1, \ldots, t_\eta)) = \mathit{yield}_g(\mathit{collect}()) = \mathit{yield}_g(\mathit{nil}) = \emptyset$. Moreover, $\mathit{yield}_g(\mathit{collect}(t_1)) \cup \ldots \cup \mathit{yield}_g(\mathit{collect}(t_\eta)) = \emptyset$, since it is the empty union.

(ii) Let $\eta > 0$. Then

$$\begin{array}{ll}
 & \mathit{yield}_g(\mathit{collect}(t_1, \ldots, t_\eta)) \\
= & \mathit{yield}_g(c_3(\#, t_1, \mathit{collect}(t_2, \ldots, t_\eta))) \\
= & \mathit{yield}_g(\#) \overset{OI}{\leftarrow} (\mathit{yield}_g(t_1), \mathit{yield}_g(\mathit{collect}(t_2, \ldots, t_\eta))) \\
= & \{y_1, y_2\} \overset{OI}{\leftarrow} (\mathit{yield}_g(t_1), \mathit{yield}_g(\mathit{collect}(t_2, \ldots, t_\eta))) \\
= & \mathit{yield}_g(t_1) \cup \mathit{yield}_g(\mathit{collect}(t_2, \ldots, t_\eta)) \\
= & \mathit{yield}_g(t_1) \cup \mathit{yield}_g(t_2) \cup \ldots \cup \mathit{yield}_g(t_\eta) \\
 & \text{(by the induction hypothesis).}
\end{array}$$

◇

Lemma 7.5 Let $s \in T_\Sigma$, and $\langle q', h \rangle \in C_{M',s}$. Moreover, assume that there are $\eta \geq 0$ and $t_1, \ldots, t_\eta \in T_\Gamma$ such that $\langle q', h \rangle \Rightarrow^*_{M',s} t_i$ for every $1 \leq i \leq \eta$. Then $\langle q', h \rangle \Rightarrow^*_{M',s} \mathit{collect}(t_1, \ldots, t_\eta)$.

Proof. By induction on η. Assume that $\mathit{test}(h) = (\sigma, b, j)$.

(i) If $\eta = 0$ then, by (1) of Definition 7.1, the rule $\langle q', \sigma, b, j \rangle \to \mathit{nil}$ is in R', hence we obtain that $\langle q', h \rangle \Rightarrow_{M',s} \mathit{nil} = \mathit{collect}()$.

(ii) Let $\eta > 0$. Then it follows that

$$\begin{array}{llll}
\langle q', h \rangle & \Rightarrow_{M',s} & c_3(\#, \langle q', h \rangle, \langle q', h \rangle) & \text{(by (3) of Definition 7.1)} \\
 & \Rightarrow^*_{M',s} & c_3(\#, t_1, \langle q', h \rangle) & \text{(since } \langle q', h \rangle \Rightarrow^*_{M',s} t_1\text{)} \\
 & \Rightarrow^*_{M',s} & c_3(\#, t_1, \mathit{collect}(t_2, \ldots, t_\eta)) & \text{(by the induction hypothesis)} \\
 & = & \mathit{collect}(t_1, \ldots, t_\eta). & \text{(by Definition 7.3)}
\end{array}$$

◇

We will also need the following technical lemma.

Lemma 7.6 For every tree $\zeta \in T_\Delta(Y_d)$, we have $\zeta = \mathit{yield}_g(\mathit{comb}(\zeta))$.

Proof. The proof is performed by structural induction on ζ.

(i) If $\zeta = y_i \in Y_d$, then $\mathit{yield}_g(\mathit{comb}(\zeta)) = \mathit{yield}_g(\alpha_i) = y_i = \zeta$.

(ii) If $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 0$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_\Delta(Y_d)$, then

$$\begin{aligned}
& \mathit{yield}_g(\mathit{comb}(\zeta)) \\
= \; & \mathit{yield}_g(c_{k+1}(\delta', \mathit{comb}(\zeta_1), \ldots, \mathit{comb}(\zeta_k))) \\
= \; & \mathit{yield}_g(\delta') \overset{OI}{\leftarrow} (\mathit{yield}_g(\mathit{comb}(\zeta_1)), \ldots, \mathit{yield}_g(\mathit{comb}(\zeta_k))) \\
= \; & \mathit{yield}_g(\delta') \overset{OI}{\leftarrow} (\zeta_1, \ldots, \zeta_k) \\
& \text{(by the induction hypothesis)} \\
= \; & \delta(y_1, \ldots, y_k) \overset{OI}{\leftarrow} (\zeta_1, \ldots, \zeta_k) \\
= \; & \delta(\zeta_1, \ldots, \zeta_k) \\
= \; & \zeta.
\end{aligned}$$

◇

7.2 Proving $\tau_M = \tau_{M'} \circ \mathit{yield}_g$

We will need the following concept. For every $s \in T_\Sigma$, $\xi \in T_{\Delta \cup C_{M,s}}(Y)$, and $l \geq 0$ we define

$$M_s(\xi, l) = \{t \in T_\Delta(Y) \mid \exists l' \leq l : \xi \Rightarrow^{l'}_{M,s} t\}.$$

Note that $\tau_M(s) = \bigcup_{l \geq 0} M_s(\langle q_0, (\varepsilon, [\,])\rangle, l)$. Now we prove our main decomposition result.

Lemma 7.7 $\tau_M = \tau_{M'} \circ \mathit{yield}_g$.

Proof. Let us fix an arbitrary input tree $s \in T_\Sigma$. For every pebble configuration $h \in PC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, we introduce the abbreviations for the following second-order and first-order substitutions:

$$[\![\ldots]\!]_h = [\![\langle p, \varphi\rangle \leftarrow \langle p, \varphi(h)\rangle(y_1, \ldots, y_{\mathit{rank}(p)}) \mid p \in Q, \varphi \in I_{\sigma,b,j}]\!]$$

and

$$[\![\ldots]\!]'_h = [\![\langle p', \varphi\rangle \leftarrow \langle p', \varphi(h)\rangle \mid p' \in Q', \varphi \in I_{\sigma,b,j}]\!].$$

First we prove that $\tau_M(s) \subseteq \mathit{yield}_g(\tau_{M'}(s))$. In fact, we prove the following stronger statement by simultaneous induction.

Statement 1.

K(l): For every $m \geq 0$, $q \in Q^{(m)}$, $h \in PC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, and $t \in T_\Delta(Y_m)$, if $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M,s} t$, then there is a tree $t' \in T_\Gamma$ such that $\langle q', h\rangle \Rightarrow^*_{M',s} t'$ and $t \in \mathit{yield}_g(t')$.

L(l): Let $h \in PC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$, and $L \subseteq M_s(\zeta[\![\ldots]\!]_h, l)$ a finite, nonempty tree language. Then there is a tree $t' \in T_\Gamma$ such that $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$ and $L \subseteq \mathit{yield}_g(t')$.

The desired inclusion follows from K because for every output tree $t \in T_\Delta$, if $t \in \tau_M(s)$, i.e., $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^l_{M,s} t$ for some $l \geq 1$, then there is a tree $t' \in T_\Gamma$ such that $\langle q'_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M',s} t'$ and $t \in \mathit{yield}_g(t')$, i.e., $t \in \mathit{yield}_g(\tau_{M'}(s))$.

Proof of Statement 1.

IB: (proof of L(0))

Let us assume that $L \subseteq M_s(\zeta[\![\ldots]\!]_h, 0)$ and $L \neq \emptyset$. Then necessarily $\zeta \in T_\Delta(Y_d)$ and $L = \{\zeta\}$. Since $\mathit{yield}_g(\mathit{comb}(\zeta)) = \zeta$ (see Lemma 7.6), we obtain that $\mathit{comb}(\zeta) \in T_\Gamma$. Hence, if we choose $t' = \mathit{comb}(\zeta)$, then $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^0_{M',s} t'$ and $L \subseteq \mathit{yield}_g(t')$.

IS1: (proof of L$[l]$ ⇒ K$(l+1)$)

Assume that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^{l+1}_{M,s} t$. Then there exists a sentential form $\xi \in T_{\Delta \cup C_{M,s}}(Y_m)$ such that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow_{M,s} \xi \Rightarrow^l_{M,s} t$. Moreover, there is a tree $\zeta \in \mathit{rhs}_M(q, \sigma, b, j)$ such that $\zeta[\![\ldots]\!]_h = \xi$ and hence $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$.

Now let $L = \{t\} \subseteq M_s(\zeta[\![\ldots]\!]_h, l)$. By the induction hypothesis L$[l]$ we obtain that there is a tree $t' \in T_\Gamma$ such that $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$ and $t \in \mathit{yield}_g(t')$ (i.e., $L \subseteq \mathit{yield}_g(t')$). Finally, by (2) of Definition 7.1, the rule $\langle q', \sigma, b, j\rangle \to \mathit{comb}(\zeta)$ is in R' and hence the derivation $\langle q', h\rangle \Rightarrow_{M',s} \mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$ holds. This proves K$[l+1]$.

IS2: (proof of K$[l]$ ⇒ L(l))

Let $L \subseteq M_s(\zeta[\![\ldots]\!]_h, l)$ be an arbitrary finite, nonempty set of output trees. We prove by induction on ζ. Note that neither $\zeta \in \Delta^{(0)}$ nor $\zeta \in Y_d$ is possible because $l \geq 1$. Hence we have cases (i) and (ii).

(i) Let $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 1$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. We make the following observations.

(a) It is straightforward that there are finite and nonempty tree languages $L_1 \subseteq M_s(\zeta_1[\![\ldots]\!]_h, l), \ldots, L_k \subseteq M_s(\zeta_k[\![\ldots]\!]_h, l)$ such that $L \subseteq \delta(L_1, \ldots, L_k)$.

(b) By induction hypothesis on ζ, there are trees $t'_1, \ldots, t'_k \in T_\Gamma$ such that $\mathit{comb}(\zeta_1)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'_1, \ldots, \mathit{comb}(\zeta_k)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'_k$ and $L_1 \subseteq \mathit{yield}_g(t'_1), \ldots, L_k \subseteq \mathit{yield}_g(t'_k)$.

Let us choose $t' = c_{k+1}(\delta', t'_1, \ldots, t'_k)$. It follows from (a) and (b) that $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$. Moreover, we have

$$\begin{array}{rcl} L & \subseteq & \delta(L_1, \ldots, L_k) \\ & = & \delta(y_1, \ldots, y_k) \overset{OI}{\leftarrow} (L_1, \ldots, L_k) \\ & \subseteq & \mathit{yield}_g(\delta') \overset{OI}{\leftarrow} (\mathit{yield}_g(t'_1), \ldots, \mathit{yield}_g(t'_k)) \\ & = & \mathit{yield}_g(c_{k+1}(\delta', t'_1, \ldots, t'_k)) \\ & = & \mathit{yield}_g(t'). \end{array}$$

This proves case (i). Note that we did not use induction hypothesis K$[l]$ in this case.

(ii) Let $\zeta = \langle p, \varphi\rangle(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 0$, $\langle p, \varphi\rangle \in \langle Q, I_d\rangle^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$.

(a) Since L is finite and nonempty, there are finite tree languages

$$\begin{aligned}&\hat{L} \subseteq M_s(\langle p, \varphi(h)\rangle(y_1, \ldots, y_k), l)\\ &L_1 \subseteq M_s(\zeta_1[\![\ldots]\!]_h, l-1)\\ &\quad\vdots\\ &L_k \subseteq M_s(\zeta_k[\![\ldots]\!]_h, l-1)\end{aligned}$$

such that $L \subseteq \hat{L} \overset{OI}{\leftarrow} (L_1, \ldots, L_k)$. Moreover, $L \neq \emptyset$ implies that $\hat{L} \neq \emptyset$. On the other hand, it may be possible that $L_\mu = \emptyset$ for some $1 \leq \mu \leq k$. Let $\{i_1, \ldots, i_j\}$ be the set of all indexes $1 \leq \mu \leq k$, for which $L_\mu \neq \emptyset$.

(b) By induction hypothesis on ζ, there are trees $t'_{i_1}, \ldots, t'_{i_j} \in T_\Gamma$ such that $comb(\zeta_{i_1})[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'_{i_1}, \ldots, comb(\zeta_{i_j})[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'_{i_j}$ and $L_{i_1} \subseteq yield_g(t'_{i_1}), \ldots, L_{i_j} \subseteq yield_g(t'_{i_j})$.

(c) For each $1 \leq \mu \leq k$ where $\mu \notin \{i_1, \ldots, i_j\}$, let t'_μ be an arbitrary tree in T_Γ such that $comb(\zeta_\mu)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'_\mu$. Note that t'_μ exists since, by the existence of rules (1) of Definition 7.1, M' can output symbol *nil* for each configuration. Hence M' can give output from each sentential form.

(d) Assume that $\hat{L} = \{\hat{t}_1, \ldots, \hat{t}_\eta\}$. Since $\hat{L} \subseteq M_s(\langle p, \varphi(h)\rangle(y_1, \ldots, y_k), l)$, it follows from K[$l$] that there are trees $\hat{t}'_1, \ldots, \hat{t}'_\eta \in T_\Gamma$ such that $\langle p', \varphi(h)\rangle \Rightarrow^*_{M',s} \hat{t}'_1, \ldots, \langle p', \varphi(h)\rangle \Rightarrow^*_{M',s} \hat{t}'_\eta$ and $\hat{t}_1 \in yield_g(\hat{t}'_1), \ldots, \hat{t}_\eta \in yield_g(\hat{t}'_\eta)$.

(e) It follows from Lemma 7.5 that $\langle p', \varphi(h)\rangle \Rightarrow^*_{M',s} collect(\hat{t}'_1, \ldots, \hat{t}'_\eta)$. Moreover,

$$\begin{aligned}\hat{L} &= \{\hat{t}_1, \ldots, \hat{t}_\eta\}\\ &\subseteq yield_g(\hat{t}'_1) \cup \ldots \cup yield_g(\hat{t}'_\eta)\\ &= yield_g(collect(\hat{t}'_1, \ldots, \hat{t}'_\eta)) \qquad \text{(by Lemma 7.4).}\end{aligned}$$

(f) Let us choose t' to be the tree $c_{k+1}(collect(\hat{t}'_1, \ldots, \hat{t}'_\eta), t'_1, \ldots, t'_k)$.

Now, by (b)-(f) we get that $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$. Moreover,

$$\begin{aligned}L &\subseteq \hat{L} \overset{OI}{\leftarrow} (L_1, \ldots, L_k)\\ &\subseteq yield_g(collect(\hat{t}'_1, \ldots, \hat{t}'_\eta)) \overset{OI}{\leftarrow} (yield_g(t'_1), \ldots, yield_g(t'_k))\\ &= yield_g(c_{k+1}(collect(\hat{t}'_1, \ldots, \hat{t}'_\eta), t'_1, \ldots, t'_k))\\ &= yield_g(t').\end{aligned}$$

This finishes the proof of Statement 1.

Now we prove that $yield_g(\tau_{M'}(s)) \subseteq \tau_M(s)$. We prove the following stronger statement by (a variant of the) simultaneous induction.

<u>*Statement 2.*</u>

K(0): *true*

K(l): For every $m \geq 0$, $\langle q, h\rangle \in C^{(m)}_{M,s}$, $t' \in T_\Gamma$, and $t \in T_\Delta(Y_m)$, if $\langle q', h\rangle \Rightarrow^l_{M',s} t'$ and $t \in yield_g(t')$, then $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$.

L(l): For every $h \in PC_{M,s}$ with $test(h) = (\sigma, b, j)$, $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j} \rangle}(Y_d)$, $t' \in T_\Gamma$, and $t \in T_\Delta(Y_d)$, if $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ and $t \in yield_g(t')$ then $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$.

The required inclusion follows from K because for every output tree $t \in T_\Delta$, if $t \in yield_g(\tau_{M'}(s))$, i.e., there is a $t' \in T_\Gamma$ such that $\langle q'_0, (\varepsilon, [\,])\rangle \Rightarrow^l_{M',s} t'$ for some $l \geq 1$ and $t \in yield_g(t')$, then $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} t$, i.e., $t \in \tau_M(s)$.

Proof of Statement 2.

IB: (proof of L(0))

Assume that $t \in yield_g(t')$ and $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^0_{M',s} t'$. Then $t' = comb(\zeta)$, which implies $\zeta \in T_\Delta(Y_d)$. By Lemma 7.6 we obtain $t = yield_g(t') = yield_g(comb(\zeta)) = \zeta$. Since $\zeta \in T_\Delta(Y_d)$, we also get $\zeta[\![\ldots]\!]_h = \zeta \Rightarrow^0_{M,s} t$.

IS1: (proof of $K[l] \wedge L[l] \Rightarrow K(l+1)$)

Let us assume that $\langle q', h\rangle \Rightarrow^{l+1}_{M',s} t'$ and $t \in yield_g(t')$. Then there is a sentential form $\xi' \in T_{\Gamma \cup C_{M',s}}$ such that $\langle q', h\rangle \Rightarrow_{M',s} \xi' \Rightarrow^l_{M',s} t'$ and thus there is a rule $\langle q', \sigma, b, j\rangle \to \zeta' \in R'$ such that $\zeta'[\![\ldots]\!]'_h = \xi'$. We distinguish the following cases, according to (1), (2), and (3) of Definition 7.1.

Case 1: $\zeta' = nil$. Now $l = 0$, $yield_g(t') = \emptyset$ and thus $t \notin yield_g(t')$. Hence the condition $K[l+1]$ holds.

Case 2: $\zeta' = comb(\zeta)$ for some $\zeta \in rhs_M(q, \sigma, b, j)$. Since $\zeta \in rhs_M(q, \sigma, b, j)$, we obtain that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow_{M,s} \zeta[\![\ldots]\!]_h$. Moreover, since $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ and $t \in yield_g(t')$, by the induction hypothesis $L[l]$ we get $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$ and, hence $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$ holds.

Case 3: $\zeta' = c_3(\#, \langle q', stay\rangle, \langle q', stay\rangle)$. Then there are $1 \leq l_1, l_2 < l$ and $t'_1, t'_2 \in T_\Gamma$ such that $\langle q', h\rangle \Rightarrow^{l_1}_{M',s} t'_1$, $\langle q', h\rangle \Rightarrow^{l_2}_{M',s} t'_2$, and $t' = c_3(\#, t'_1, t'_2)$. Since $t \in yield_g(t')$ and $yield_g(t') = yield_g(t'_1) \cup yield_g(t'_2)$, we have $t \in yield_g(t'_1)$ or $t \in yield_g(t'_2)$. Then, by the induction hypothesis $K[l_1]$ or $K[l_2]$, we get that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$.

IS2: (proof of $K[l] \Rightarrow L(l)$)

Assume that $t \in yield_g(t')$ and $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$. We finish the proof by induction on ζ. Neither $\zeta = \alpha \in \Delta^{(0)}$ nor $\zeta = y_i$ because $l \geq 1$. Hence, we have cases (i) and (ii).

(i) $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 1$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$.

Then $comb(\zeta) = c_{k+1}(\delta', comb(\zeta_1), \ldots, comb(\zeta_k))$. Moreover, since $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$, there are integers $l_1, \ldots, l_k \leq l$ and trees $t'_1, \ldots, t'_k \in T_\Gamma$ such that $comb(\zeta_1)[\![\ldots]\!]'_h \Rightarrow^{l_1}_{M',s} t'_1, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h \Rightarrow^{l_k}_{M',s} t'_k$ and $t' = c_{k+1}(\delta', t'_1, \ldots, t'_k)$.

Since $t \in yield_g(t')$, we get that $t = \delta(t_1, \ldots, t_k)$ with $t_1 \in yield_g(t'_1), \ldots, t_k \in yield_g(t'_k)$.

By the induction hypothesis on ζ, we obtain that $\zeta_1[\![\ldots]\!]_h \Rightarrow^*_{M,s} t_1, \ldots, \zeta_k[\![\ldots]\!]_h \Rightarrow^*_{M,s} t_k$. This concludes that $\zeta[\![\ldots]\!]_h \Rightarrow^* t$.

(ii) Assume that $\zeta = \langle p, \varphi\rangle(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 0$, $\langle p, \varphi\rangle \in \langle Q, I_d\rangle^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. Then, obviously, $comb(\zeta) =$

$c_{k+1}(\langle p', \varphi\rangle, comb(\zeta_1), \ldots, comb(\zeta_k))$. We make the following observations.

(a) There are $l' \leq l$, $l_1, \ldots, l_k < l$ and $\overline{t}', t'_1, \ldots, t'_k \in T_\Gamma$, such that $\langle p', \varphi(h)\rangle \Rightarrow^{l'}_{M',s} \overline{t}'$, $comb(\zeta_1)[\![\ldots]\!]'_h \Rightarrow^{l_1}_{M',s} t'_1, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h \Rightarrow^{l_k}_{M',s} t'_k$, and $t' = c_{k+1}(\overline{t}', t'_1, \ldots, t'_k)$.

(b) Since $t \in yield_g(t')$, there is a tree $\overline{t} \in T_\Delta(Y_d)$ such that $\overline{t} \in yield_g(\overline{t}')$ and $t \in \overline{t} \overset{OI}{\leftarrow} (yield_g(t'_1), \ldots, yield_g(t'_k))$.

(c) Recall that by $|\overline{t}|_{y_j}$ we denote the number of occurrences of y_j in $\overline{t}$. Assume that $\{i_1, \ldots, i_j\}$ is the index set $\{1 \leq \mu \leq d \mid |\overline{t}|_{y_j} \geq 1\}$. Moreover, let $\eta_{i_1} = |\overline{t}|_{y_{i_1}}, \ldots, \eta_{i_j} = |\overline{t}|_{y_{i_j}}$, and let $u^{[1]}_{i_1}, \ldots, u^{[\eta_{i_1}]}_{i_1}, \ldots, u^{[1]}_{i_j}, \ldots, u^{[\eta_{i_j}]}_{i_j} \in pos(\overline{t})$ be the pairwise different nodes of $\overline{t}$ such that $lab(\overline{t}, u^{[1]}_{i_1}) = \ldots = lab(\overline{t}, u^{[\eta_{i_1}]}_{i_1}) = y_{i_1}, \ldots, lab(\overline{t}, u^{[1]}_{i_j}) = \ldots = lab(\overline{t}, u^{[\eta_{i_j}]}_{i_j}) = y_{i_j}$.

(d) By the definition of $yield_g(t')$, there are trees

$$t_{i_1}{}^{[1]}, \ldots, t_{i_1}{}^{[\eta_{i_1}]}, \ldots, t_{i_j}{}^{[1]}, \ldots, t_{i_j}{}^{[\eta_{i_j}]} \in T_\Delta(Y_d)$$

such that $t = \overline{t}[u^{[\kappa]}_{i_\nu} \leftarrow t^{[\kappa]}_{i_\nu} \mid 1 \leq \nu \leq j,\ 1 \leq \kappa \leq i_\nu]$.

(e) Since $\langle p', \varphi(h)\rangle \Rightarrow^{l'}_{M',s} \overline{t}'$ for some $l' \leq l$, by the induction hypothesis K[l], we have $\langle p, \varphi(h)\rangle(y_1, \ldots, y_k) \Rightarrow^*_{M,s} \overline{t}$.

(f) By the induction hypothesis on ζ, we obtain that $\zeta_{i_\nu}[\![\ldots]\!]_h \Rightarrow^*_{M,s} t^{[\kappa]}_{i_\nu}$ for $1 \leq \nu \leq j$ and $1 \leq \kappa \leq i_\nu$.

Consequently, we obtain

$$\begin{array}{rl} \zeta[\![\ldots]\!]_h & = \quad \langle p, \varphi(h)\rangle(\zeta_1[\![\ldots]\!]_h, \ldots, \zeta_k[\![\ldots]\!]_h) \\ & \Rightarrow^*_{M,s} \quad \overline{t}[y_{i_1} \leftarrow \zeta_{i_1}[\![\ldots]\!]_h, \ldots, y_{i_j} \leftarrow \zeta_{i_j}[\![\ldots]\!]_h] \\ & \Rightarrow^*_{M,s} \quad \overline{t}[u^{[\kappa]}_{i_\nu} \leftarrow t^{[\kappa]}_{j_\nu} \mid 1 \leq \nu \leq j,\ 1 \leq \kappa \leq i_\nu] \\ & = \quad t, \end{array}$$

which proves IS2. This finishes also the proof of Statement 2 and of Lemma 7.7. ◇

Now we demonstrate how the above decomposition works by our running example.

Example 7.8 Let us consider the 1-pmtt M_2 and input tree s of Example 3.7, and the 1-ptt M'_2 mapping g of Example 7.2. Then, it is easy to see that $\tau_{M_2}(s) = \sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha))$. Moreover, if we let

$$t' = \overbrace{c_1(\ldots c_1(}^{23\ c_1} c_2(c_2(c_3(\sigma', \alpha_1, \alpha_1), c_2(c_3(\sigma', \alpha_1, \alpha_1), \alpha_1)), c_1(\alpha'))) \ldots),$$

and $t'' = \#(t', t', nil)$, then straightforwardly $(s, t'') \in \tau_{M'_2}$ and

$$yield_g(t'') = yield_g(t') \cup yield_g(t') \cup yield_g(nil) = \{\sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha))\}.$$

◇

Since M is an arbitrary n-pmtt, we obtain the following result.

Corollary 7.9 For each $n \geq 0$, the inclusion $n\text{-}PMTT \subseteq n\text{-}PTT \circ YIELD$ holds. $\diamond$

The next characterization result follows from Lemma 7.9 and from the converse inclusion $n\text{-}PTT \circ YIELD \subseteq n\text{-}PMTT$ of Lemma 6.1, where $n \geq 0$ is arbitrary.

Theorem 7.10 For each $n \geq 0$, we have $n\text{-}PMTT = n\text{-}PTT \circ YIELD$. $\diamond$

Let us observe that the proofs of Lemmas 7.7 and 6.1 do not depend on the type of pebble handling either. Hence Corollary 7.9 and Theorem 7.10 also hold for the weak pebble handling.

7.3 Some remarks on $n\text{-}PMTT = n\text{-}PTT \circ YIELD$

(A) Unfortunately, the n-ptt M' constructed in Definition 7.1 is nondeterministic and strongly circular (even if the n-pmtt M is deterministic and noncircular). To discuss why, let us consider rules of R' of type (1), (2), and (3) in Definition 7.1.

First let us observe that rules of type (3) make M' strongly circular immediately. However, we need these rules in the case that M is nondeterministic for the following purpose. The n-pmtt M can make several copies of a parameter ξ of a macro call, and then, due to nondeterminism, can process those copies in different ways. Since M' has no macro capability, in lack of rules of type (3), it keeps one copy of ξ and then the yield mapping will substitute the result of the (nondeterministic) processing of ξ for different occurrences of a parameter variable. Therefore the same tree will be substituted for different occurrences of a parameter variable and we have not simulated the work of M. Now, by rules of type (3), M' can compute a tree of which the image under the yield mapping is the set of all trees that M can compute from ξ. Then this set will be OI-substituted for different occurrences of a parameter variable, and thus we achieve the same result as M did.

Note that if M is deterministic, then the rules of type (3) can be omitted from M'. However, even in this case M' may be strongly circular, even if M is noncircular, see Example 8.2 later in this thesis.

The nondeterminism of M' is due to the rules of type (1), which we need in any case for the following reason. It is possible that a macro call occurs nested in the same macro call in a sentential form of M. Still, the computation of M terminates because the nested macro call will be deleted during the computation. However, M' cannot simulate this deletion because it has no macro capability, hence its corresponding computation does not terminate and the simulation of M is not guaranteed. Now, by applying the dummy rule (1), we can force that computation of M' to terminate and then the yield mapping will compute the result of the computation of M.

We can summarize by saying that the price of shifting the yield-like decomposition of macro tree transformations to pebble macro tree transformations is that, unfortunately, we lose determinism and noncircularity.

(B) Using some results of [EV86] concerning regular tree grammars and context-free tree grammars with storage, we can provide an alternative proof of Corollary 7.9. To see this we first recall some necessary concepts.

The reader is assumed to be familiar with *top-down tree transducers*, see [Rou70], [Eng75], or [FV98]. A top-down tree transducer is linear (nondeleting) if, for every rule r, it holds that each variable in the left-hand side of r occurs at most once (at least once) in the right-hand side of r. We denote the class of linear, nondeleting and deterministic top-down tree transformations by $lndT$. Moreover we need from [EV85] the tree transformation $set_{\Sigma,+,\theta} : T_{\Sigma\cup\{+,\theta\}} \to \mathcal{P}(T_\Sigma)$, which is defined as follows. The symbols $+ \notin \Sigma$ and $\theta \notin \Sigma$ are new, with rank 2 and 0, respectively. Moreover, (i) $set_{\Sigma,+,\theta}(\theta) = \emptyset$, (ii) $set_{\Sigma,+,\theta}(\sigma(s_1,\dots,s_k)) = \{\sigma(t_1,\dots,t_k) \mid t_1 \in set_{\Sigma,+,\theta}(s_1),\dots,t_k \in set_{\Sigma,+,\theta}(s_k)\}$, and (iii) $set_{\Sigma,+,\theta}(+(s_1,s_2)) = set_{\Sigma,+,\theta}(s_1) \cup set_{\Sigma,+,\theta}(s_2)$, where $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1,\dots,s_k \in T_{\Sigma\cup\{+,\theta\}}$. The class of all tree transformations $set_{\Sigma,+,\theta}$ is denoted by SET.

We also need the concept of a *regular tree grammar* and of a *context-free tree grammar*, cf. [GS84, GS97] and [ES78], respectively. Both regular tree grammars and context-free tree grammars can be extended with a storage type S, see [EV86]. We will abbreviate these concepts as RT(S)-grammars and CFT(S)-grammars, respectively. These devices take an input element of the storage type S and produce an output tree, i.e., both RT(S)-grammars and CFT(S)-grammars are storage-to-tree transducers. Let $\mathcal{RT}(S)$ and $\mathcal{CFT}(S)$ denote the class of transformations induced by RT(S)-grammars and CFT(S)-grammars.

Particular storage types are e.g. “Tree-walk”, “Tree-walk with n weak pebbles”, and “Tree-walk with n (strong) pebbles” for $n \geq 0$. Now, if we take $S =$“Tree-walk”, then RT(S)-grammars precisely correspond to 0-ptts, see Section 3.3 of [EM03], and similarly, CFT(S)-grammars correspond to 0-pmtts. Analogously, in case $S =$“Tree-walk with n pebbles”, grammars RT(S) and CFT(S) are the same as n-ptts and n-pmtts, respectively.

Next, we recall some necessary results based on the above concepts.

(a) By a careful reading of the proof of Theorem 3.26 of [EV86] we see that Corollary 3.27 of the same article states the inclusion $\mathcal{CFT}(S) \subseteq \mathcal{RT}(S) \circ lnMTT$, where l and n denote *linear* and *nondeleting*, respectively, and mean the same as for top-down tree transducers. Now letting S be the storage type “Tree-walk with n pebbles” we also obtain that $n\text{-}PMTT \subseteq n\text{-}PTT \circ lnMTT$.

(b) By Theorems 4.8 and 6.10 of [EV85] we have $lndtMTT \subseteq lndtT \circ dtYIELD$ and $lnMTT \subseteq lndtMTT \circ SET$, respectively.

(c) The equality $n\text{-}PTT \circ lndtT = n\text{-}PTT$ can be proved by the usual product construction, as it was done, e.g., for top-down tree transducers in Theorem 1 of [Bak79].

(d) It is easy to see that $YIELD = dtYIELD \circ SET$.

Now we are able to give the alternative proof of Corollary 7.9 we mentioned above.

$$\begin{array}{rcll}
\mathit{n\text{-}PMTT} & \subseteq & \mathit{n\text{-}PTT} \circ \mathit{lnMTT} & \text{(by (a))} \\
 & \subseteq & \mathit{n\text{-}PTT} \circ \mathit{lndtMTT} \circ \mathit{SET} & \text{(by (b))} \\
 & \subseteq & \mathit{n\text{-}PTT} \circ \mathit{lndtT} \circ \mathit{dtYIELD} \circ \mathit{SET} & \text{(by (b))} \\
 & = & \mathit{n\text{-}PTT} \circ \mathit{dtYIELD} \circ \mathit{SET} & \text{(by (c))} \\
 & = & \mathit{n\text{-}PTT} \circ \mathit{YIELD} & \text{(by (d)).}
\end{array}$$

(C) Finally we mention that Theorem 7.10 can be generalized to regular tree grammars and context-free tree grammars with an arbitrary storage type as follows. In fact, we can straightforwardly extend both Corollary 7.9 and Lemma 6.1 to CFT(S) and RT(S) grammars, and obtain the decomposition $\mathcal{CFT}(S) \subseteq \mathcal{RT}(S) \circ \mathit{YIELD}$ and composition $\mathcal{RT}(S) \circ \mathit{YIELD} \subseteq \mathcal{CFT}(S)$, respectively. Hence we get the following theorem.

Theorem 7.11 For each storage type S we have $\mathcal{CFT}(S) = \mathcal{RT}(S) \circ \mathit{YIELD}$. $\diamond$

Finally we note that, by Definition 7.1, M' and yield_g can be obtained from M in polynomial time. On the contrary, Theorem 3.26 of [EV86] needs exponential time with respect the size of M. Fore more complexity issues of the characterization $\mathit{n\text{-}PMTT} = \mathit{n\text{-}PTT} \circ \mathit{YIELD}$, see [FM09].

8 Decomposition result for restricted pmtts

In this section we give the decomposition of a tree transformation computed by a n-pmtt M into the composition of an n-ptt tree transformation and a yield tree transformation in an alternative way. We do this in order to attain that some good properties (noncircularity, determinism) of M are preserved for the pmtt. Again, we associate with M a total n-ptt M' and a yield tree transformation $yield_g$ and discuss some properties of M'. Then we prove that $\tau_M = \tau_{M'} \circ yield_g$ holds provided M is deterministic (or context-linear) and M' is noncircular. Finally, we conclude by giving sufficient conditions for M which guarantees that M' is noncircular. The results of this section can be found in Section 6 of [FM08]. Again, it is irrelevant that the pebble macro tree transducer model of [FM08] works with weak pebble handling.

In this section let $M = (Q, \Sigma, \Delta, q_0, R)$ be an n-pmtt and $d = maxr(Q \cup \Sigma \cup \Delta)$.

8.1 Associating an n-ptt and a yield tree transformation with an n-pmtt

Definition 8.1 We define the <u>total</u> n-ptt M' and the yield tree transformation $yield_g$ associated with M as follows.

The n-ptt associated with M is $M' = (Q', \Sigma, \Gamma, q'_0, R')$, where

- $Q' = Q'^{(0)} = \{q' \mid q \in Q\}$,
- Γ is the ranked alphabet defined by
  - $\Gamma^{(0)} = \{\delta' \mid \delta \in \Delta\} \cup \{\alpha_1, \ldots, \alpha_d\} \cup \{nil\}$,
  - for every $i \geq 1$,
  $$\Gamma^{(i)} = \begin{cases} \{c_i\} & \text{if } Q^{(i-1)} \cup \Delta^{(i-1)} \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$$
- In order to give the rule set R', we need the mapping $comb : T_{\Delta \cup \langle Q, I_d \rangle}(Y_d) \to T_{\Gamma \cup \langle Q', I_d \rangle}$ that we define as follows, cf. Lemma 5.5 in [EV85] and Lemma 4.34 in [FV98].

  (i) For every $1 \leq i \leq d$, let $comb(y_i) = \alpha_i$,

  (ii) for every $k \geq 0$, state $q \in Q^{(k)}$, instruction $\varphi \in I_d$, and trees $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_d \rangle}(Y_d)$ let
  $$comb(\langle q, \varphi \rangle(\zeta_1, \ldots, \zeta_k)) = c_{k+1}(\langle q', \varphi \rangle, comb(\zeta_1), \ldots, comb(\zeta_k)),$$

  (iii) for every symbol $\delta \in \Delta^{(k)}$ with $k \geq 0$ and trees $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_d \rangle}(Y_d)$, let
  $$comb(\delta(\zeta_1, \ldots, \zeta_k)) = c_{k+1}(\delta', comb(\zeta_1), \ldots, comb(\zeta_k)).$$

  Now let $R' = R_1 \cup R_2$ with
  $$R_1 = \{\langle q', \sigma, b, j \rangle \to comb(\zeta) \mid \langle q, \sigma, b, j \rangle(y_1, \ldots, y_m) \to \zeta \in R\}$$

and

$$R_2 = \{\langle q', \sigma, b, j\rangle \to nil \mid m \geq 0,\ q \in Q^{(m)},\ \sigma \in \Sigma,\ b \in \{0,1\}^{\leq n},$$
$$0 \leq j \leq maxr(\Sigma),\ \text{and } rhs_M(q, \sigma, b, j) = \emptyset\}$$

Moreover, let $yield_g$ be the tree transformation induced by the deterministic and partial mapping $g : \Gamma^{(0)} \to T_\Delta(Y_d)$ defined in the following way.

- For every $k \geq 0$, symbol $\delta \in \Delta^{(k)}$, let $g(\delta') = \delta(y_1, \ldots, y_k)$,
- for every $1 \leq i \leq d$, let $g(\alpha_i) = y_i$, and
- let $g(nil) = \emptyset$. $\diamond$

It is easy to show that M' is total and that if M is deterministic, then so is M'. In particular, if M is a macro tree transducer, then M' is a total top-down tree transducer, cf. the construction described in Lemma 5.5 of [EV85] and Lemma 4.34 of [FV98].

We made M' total because of a phenomenon, which shows up already in case that M is a macro tree transducer and which can be described as follows. Note that we are going to achieve that $\tau_M = \tau_{M'} \circ yield_g$. Since we work with OI derivation mode, M may compute a sentential form ξ from an input tree s which contains a configuration $\langle q, u\rangle$ in a parameter position for which there is no applicable rule of M. Then, in a later step of its computation, M deletes that configuration thus it eventually produces an output tree for s. But, since there is no rule of M for $\langle q, u\rangle$, the top-down tree transducer M' will not have a rule for its configuration corresponding to $\langle q, u\rangle$. Moreover, that configuration of M' cannot be in a parameter position, because M' is a top-down tree transducer, hence it will not be deleted. This means M' cannot produce an output tree for s, hence $\tau_M = \tau_{M'} \circ yield_g$ will not hold. We manage this problem by making M' total by providing it with extra rules with a nullary symbol nil on their right-hand side. Thus M' produces output for every input tree s. Then $yield_g$, being partial, simulates the deletion of M by deleting the occurrences of nil from its input tree.

Another problem is the circularity. It may happen that, while M is noncircular, M' is strongly circular. This is demonstrated in the following example.

Example 8.2 Let M_2 be the 0-pmtt of Example 4.4 (which is noncircular).

The total 0-ptt associated with M_2 is

$$M_2' = (\{q_0', q_1'\}, \{\alpha\}, \{\alpha'^{(0)}, \alpha_1^{(0)}, nil^{(0)}, c_1^{(1)}, c_2^{(2)}\}, q_0', R'),$$

where R' consists of the rules

- $\langle q_0', \alpha, 0\rangle \to c_2(\langle q_1', stay\rangle, c_1(\langle q_0', stay\rangle))$,
- $\langle q_1', \alpha, 0\rangle \to c_1(\alpha')$.

The configuration $\langle q_0', \varepsilon\rangle$ is circular because

$$\langle q_0', \varepsilon\rangle \Rightarrow_{M_2',\alpha} c_2(\langle q_1', \varepsilon\rangle, c_1(\langle q_0', \varepsilon\rangle)),$$

and $\langle q_0', \varepsilon\rangle$ labels an outside-active node in the latter sentential form, hence M_2' is strongly circular. $\diamond$

Hence, M' cannot produce output for an input tree s, for which M can and again $\tau_M = \tau_{M'} \circ \mathit{yield}_g$ does not hold. We handle this problem by requiring that M' is noncircular, see Lemma 8.4. Moreover, we give a condition for M which guarantees that M' is noncircular, hence the decomposition works. This condition will be that M is not weakly circular, cf. Lemma 8.9.

8.2 Proving $\tau_M = \tau_{M'} \circ \mathit{yield}_g$

In this subsection $\mathit{comb} : T_{\Delta\cup\langle Q,I_d\rangle}(Y_d) \to T_{\Gamma\cup\langle Q',I_d\rangle}$ is the mapping introduced in Definition 8.1, $M' = (Q', \Sigma, \Gamma, q_0', R')$ and yield_g stand for the total n-ptt and the yield tree transformation associated with M in the sense of Definition 8.1.

We will show that in certain cases $\tau_M = \tau_{M'} \circ \mathit{yield}_g$ holds. We need the following technical lemma. For first reading its proof can be skipped.

Lemma 8.3 For every tree $\zeta \in T_\Delta(Y_d)$, we have $\zeta = \mathit{yield}_g(\mathit{comb}(\zeta))$.

Proof. The proof is performed by structural induction on ζ.

(i) If $\zeta = y_i \in Y_d$, then $\mathit{yield}_g(\mathit{comb}(\zeta)) = \mathit{yield}_g(\alpha_i) = y_i = \zeta$.

(ii) If $\zeta = \delta(\zeta_1, \dots, \zeta_k)$ for some $k \geq 0$, $\delta \in \Delta^{(k)}$ and $\zeta_1, \dots, \zeta_k \in T_\Delta(Y_d)$, then

$$\begin{aligned}
& \mathit{yield}_g(\mathit{comb}(\zeta)) \\
= \; & \mathit{yield}_g(c_{k+1}(\delta', \mathit{comb}(\zeta_1), \dots, \mathit{comb}(\zeta_k))) \\
= \; & \mathit{yield}_g(\delta')[y_1 \leftarrow \mathit{yield}_g(\mathit{comb}(\zeta_1)), \dots, y_k \leftarrow \mathit{yield}_g(\mathit{comb}(\zeta_k))] \\
= \; & \delta(y_1, \dots, y_k)[y_1 \leftarrow \mathit{yield}_g(\mathit{comb}(\zeta_1)), \dots, y_k \leftarrow \mathit{yield}_g(\mathit{comb}(\zeta_k))] \\
= \; & \delta(\mathit{yield}_g(\mathit{comb}(\zeta_1)), \dots, \mathit{yield}_g(\mathit{comb}(\zeta_k))) \\
= \; & \delta(\zeta_1, \dots, \zeta_k) \text{ (by induction hypothesis)} \\
= \; & \zeta.
\end{aligned}$$

$\diamond$

The first decomposition result we prove is the following. Recall that if M' is noncircular, then M is also noncircular, but the reverse direction is not always true.

Lemma 8.4 If M is deterministic and M' is noncircular, then $\tau_M = \tau_{M'} \circ \mathit{yield}_g$.

Proof. Let $s \in T_\Sigma$ be an arbitrary input tree. First let us note, that M' is deterministic, total, and noncircular. Hence, by Proposition 4.6, for every sentential form $\xi \in T_{\Gamma\cup C_{M',s}}$, the normal form $\mathit{nf}(\xi, \Rightarrow_{M',s})$ exists and it is in T_Γ.

For every pebble configuration $h \in IC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, we introduce the following abbreviations for second-order substitutions:

$$[\![\dots]\!]_h = [\![\langle p, \varphi\rangle \leftarrow \langle p, \varphi(h)\rangle(y_1, \dots, y_{\mathit{rank}(p)}) \mid p \in Q, \varphi \in I_{\sigma,b,j}]\!]$$

and

$$[\![\ldots]\!]'_h = [\![\langle p', \varphi\rangle \leftarrow \langle p', \varphi(h)\rangle \mid p' \in Q', \varphi \in I_{\sigma,b,j}]\!].$$

Now we prove two statements from which the equation $\tau_M = \tau_{M'} \circ yield_g$ follows. Both statements are in fact two families of predicates which can be proved by simultaneous induction, see Section 2.4.

Statement 1.

K(l): For every $m \geq 0$, configuration $\langle q, h\rangle \in C^{(m)}_{M,s}$, and output tree $t \in T_\Delta(Y_m)$, if $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M,s} t$, then $yield_g(nf(\langle q', h\rangle, \Rightarrow_{M',s})) = t$.

L(l): For every pebble configuration $h \in IC_{M,s}$ with $test(h) = (\sigma, b, j)$, tree $\zeta \in T_{\Delta\cup\langle Q, I_{\sigma,b,j}\rangle}(Y_d)$, and $t \in T_\Delta(Y_d)$, if $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$, then $yield_g(nf(comb(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) = t$.

Proof of Statement 1.

IB: (proof of L(0))

Let us assume that $\zeta[\![\ldots]\!]_h \Rightarrow^0_{M,s} t$. It follows $t = \zeta$, hence also $\zeta \in T_\Delta(Y_d)$. Since $comb(\zeta) \in T_\Gamma$, we have $comb(\zeta)[\![\ldots]\!]'_h = comb(\zeta)$ and thus $nf(comb(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s}) = comb(\zeta)$. Now, by Lemma 8.3, $yield_g(comb(\zeta)) = \zeta = t$.

IS1: (proof of L[l] $\Rightarrow$ K($l+1$))

Assume that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^{l+1}_{M,s} t$ holds. Then there exists a sentential form $\xi \in T_{\Delta\cup C_{M,s}}(Y_m)$ such that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow_{M,s} \xi \Rightarrow^l_{M,s} t$. Thus there is a rule $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta \in R$ satisfying $\zeta[\![\ldots]\!]_h = \xi$.

Now, by the definition of M', the rule $\langle q', \sigma, b, j\rangle \to comb(\zeta)$ is in R', hence $\langle q', h\rangle \Rightarrow_{M',s} comb(\zeta)[\![\ldots]\!]'_h$ and thus $nf(\langle q', h\rangle, \Rightarrow_{M',s}) nf(comb(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s})$. On the other hand, by the induction hypothesis L[l], we obtain $yield_g(nf(comb(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) = t$.

IS2: (proof of K[l] $\Rightarrow$ L(l))

Assume that $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$. We continue the proof by induction on ζ. The cases $\zeta = \alpha \in \Delta^{(0)}$ and $\zeta = y_i \in Y_d$ are not possible because $l \geq 1$. Hence we have cases (i) and (ii).

(i) Let $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 1$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta\cup\langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. There are integers $l_1, \ldots, l_k \leq l$ and output trees $t_1, \ldots, t_k \in T_\Delta(Y_d)$ such that

$$\zeta_1[\![\ldots]\!]_h \Rightarrow^{l_1}_{M,s} t_1, \ldots, \zeta_k[\![\ldots]\!]_h \Rightarrow^{l_k}_{M,s} t_k$$

and $\delta(t_1, \ldots, t_k) = t$. By induction hypothesis on ζ,

$$yield_g(nf(comb(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) = t_1, \ldots, yield_g(nf(comb(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) = t_k.$$

Since

$$\begin{array}{rl} & comb(\zeta)[\![\ldots]\!]'_h \\ = & c_{k+1}(\delta', comb(\zeta_1), \ldots, comb(\zeta_k))[\![\ldots]\!]'_h \\ = & c_{k+1}(\delta', comb(\zeta_1)[\![\ldots]\!]'_h, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h), \end{array}$$

we obtain

$$\begin{array}{rl} & yield_g(nf(comb(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) \\ = & yield_g(nf(c_{k+1}(\delta', comb(\zeta_1)[\![\ldots]\!]'_h, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h), \Rightarrow_{M',s})) \\ = & yield_g(c_{k+1}(\delta', nf(comb(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s}), \ldots, nf(comb(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s}))) \\ = & \delta(y_1, \ldots, y_k)[y_1 \leftarrow yield_g(nf(comb(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s})), \ldots, \\ & \qquad y_k \leftarrow yield_g(nf(comb(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s}))] \\ = & \delta(y_1, \ldots, y_k)[y_1 \leftarrow t_1, \ldots, y_k \leftarrow t_k] \\ = & \delta(t_1, \ldots, t_k) \\ = & t. \end{array}$$

This proves case (i). Note that we did not use the induction hypothesis K$[l]$ in this case.

(ii) Let $\zeta = \langle p, \varphi\rangle(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 0$, $\langle p, \varphi\rangle \in \langle Q, I_d\rangle^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. We make the following observations (a)-(d).

(a) By our assumption in $L[l]$, there are $l' \leq l$ and $\bar{t} \in T_\Delta(Y_k)$ such that $\langle p, \varphi(h)\rangle(y_1, \ldots, y_k) \Rightarrow^{l'}_{M,s} \bar{t}$. Moreover, for every $1 \leq j \leq k$ with $|\bar{t}|_{y_j} \geq 1$, there are a $l_j < l$ and $t_j \in T_\Delta(Y_d)$ satisfying

$$\zeta_j[\![\ldots]\!]_h \Rightarrow^{l_j}_{M,s} t_j$$

Moreover, $\bar{t}[y_j \leftarrow t_j \mid 1 \leq j \leq k, |\bar{t}|_{y_j} \geq 1] = t$.

Here we used that M is deterministic: the same tree t_j can be substituted for different occurrences of y_j.

(b) By induction hypothesis K$[l]$, we have $yield_g(nf(\langle p', \varphi(h)\rangle, \Rightarrow_{M',s})) = \bar{t}$.

(c) Since M' is deterministic, total, and noncircular,

$$nf(comb(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s}), \ldots, nf(comb(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s})$$

exist and they are in T_Γ, cf. Proposition 4.6.

(d) By the induction hypothesis on ζ, for every $1 \leq j \leq k$ with $|\bar{t}|_{y_j} \geq 1$, we have $yield_g(nf(comb(\zeta_j)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) = t_j$.

Since

$$\begin{array}{rcl} comb(\zeta)[\![\ldots]\!]'_h & = & c_{k+1}(\langle p', \varphi\rangle, comb(\zeta_1), \ldots, comb(\zeta_k))[\![\ldots]\!]'_h \\ & = & c_{k+1}(\langle p', \varphi(h)\rangle, comb(\zeta_1)[\![\ldots]\!]'_h, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h) \end{array}$$

we get

$$\begin{aligned}
& \mathit{yield}_g(\mathit{nf}(\mathit{comb}(\zeta)[\![\ldots]\!]'_h, \Rightarrow_{M',s})) \\
= \; & \mathit{yield}_g(\mathit{nf}(c_{k+1}(\langle p', \varphi(h)\rangle, \mathit{comb}(\zeta_1)[\![\ldots]\!]'_h, \ldots, \mathit{comb}(\zeta_k)[\![\ldots]\!]'_h), \Rightarrow_{M',s})) \\
= \; & \mathit{yield}_g(c_{k+1}(\mathit{nf}(\langle p', \varphi(h)\rangle, \Rightarrow_{M',s}), \mathit{nf}(\mathit{comb}(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s}), \ldots, \\
& \qquad\qquad \mathit{nf}(\mathit{comb}(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s}))) \\
= \; & \mathit{yield}_g(\mathit{nf}(\langle p', \varphi(h)\rangle, \Rightarrow_{M',s})) \overset{OI}{\leftarrow} (\mathit{yield}_g(\mathit{nf}(\mathit{comb}(\zeta_1)[\![\ldots]\!]'_h, \Rightarrow_{M',s})), \ldots, \\
& \qquad\qquad \mathit{yield}_g(\mathit{nf}(\mathit{comb}(\zeta_k)[\![\ldots]\!]'_h, \Rightarrow_{M',s}))) \\
= \; & \bar{t} \overset{OI}{\leftarrow} (t_1, \ldots, t_k) \\
= \; & \bar{t}[y_j \leftarrow t_j \mid 1 \le j \le k, |\bar{t}|_{y_j} \ge 1] \\
= \; & t.
\end{aligned}$$

This finishes the proof of Statement 1. Now we form and prove Statement 2.

Statement 2.

K(l): For every $m \ge 0$, configuration $\langle q, h\rangle \in C^{(m)}_{M,s}$, trees $t' \in T_\Gamma$ and $t \in T_\Delta(Y_m)$, if $\langle q', h\rangle \Rightarrow^l_{M',s} t'$ and $\mathit{yield}_g(t') = t$, then $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$.

L(l): For every pebble configuration $h \in IC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, tree $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$, trees $t' \in T_\Gamma$ and $t \in T_\Delta(Y_d)$, if $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ and $\mathit{yield}_g(t') = t$, then $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$.

Proof of Statement 2.

IB: (proof of L(0))

Assume $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^0_{M',s} t'$ and $\mathit{yield}_g(t') = t$. Then $t' = \mathit{comb}(\zeta)[\![\ldots]\!]'_h$, which implies $t' = \mathit{comb}(\zeta)$ and $\zeta \in T_\Delta(Y_d)$. By Lemma 8.3 we obtain $t = \mathit{yield}_g(t') = \mathit{yield}_g(\mathit{comb}(\zeta)) = \zeta$. Since $\zeta \in T_\Delta(Y_d)$, we also get $\zeta[\![\ldots]\!]_h = \zeta \Rightarrow^0_{M,s} t$.

IS1: (proof of $L[l] \Rightarrow$ K($l+1$))

Let us assume that $\langle q', h\rangle \Rightarrow^{l+1}_{M',s} t'$ and $\mathit{yield}_g(t') = t$. Then there is a $\xi' \in T_{\Gamma \cup C_{M',s}}$ such that $\langle q', h\rangle \Rightarrow_{M',s} \xi' \Rightarrow^l_{M',s} t'$. Note that $\xi' \ne \mathit{nil}$, otherwise $l = 0$, $t' = \mathit{nil}$, and $t \ne \mathit{yield}_g(t')$. Hence there is a rule $\langle q', \sigma, b, j\rangle \to \mathit{comb}(\zeta) \in R'$ with $\mathit{comb}(\zeta)[\![\ldots]\!]'_h = \xi'$ applicable for $\langle q', h\rangle$ for some $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}$. By the construction of R', we have $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta \in R$ and thus $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow_{M,s} \zeta[\![\ldots]\!]_h$. Moreover, by the induction hypothesis $L[l]$, we have $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$. Then it follows that $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$.

IS2: (proof of K$[l] \Rightarrow$ L(l))

Now assume that $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ for some $l \ge 1$ and $\mathit{yield}_g(t') = t$. We finish the proof by induction on ζ. The cases $\zeta = \alpha \in \Delta^{(0)}$ and $\zeta = y_i \in Y_d$ are not possible because $l \ge 1$.

(i) Let $\zeta = \delta(\zeta_1, \ldots, \zeta_k)$ for some $k \ge 1$, $\delta \in \Delta^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. Then, obviously, $\mathit{comb}(\zeta) = c_{k+1}(\delta', \mathit{comb}(\zeta_1), \ldots, \mathit{comb}(\zeta_k))$ and thus there are integers $l_1, \ldots, l_k \le l$ and trees $t'_1, \ldots, t'_k \in T_\Gamma$ such that

$$\mathit{comb}(\zeta_1)[\![\ldots]\!]'_h \Rightarrow^{l_1}_{M',s} t'_1, \ldots, \mathit{comb}(\zeta_k)[\![\ldots]\!]'_h \Rightarrow^{l_k}_{M',s} t'_k$$

and $c_{k+1}(\delta', t'_1, \ldots, t'_k) = t'$.

Since $yield_g(t') = t$ and $yield_g(t') = \delta(y_1, \ldots, y_k)[y_1 \leftarrow yield_g(t'_1), \ldots, y_k \leftarrow yield_g(t'_k)]$, there are output trees $t_1, \ldots, t_k \in T_\Delta(Y_d)$ such that $yield_g(t'_1) = t_1, \ldots, yield_g(t'_k) = t_k$ and $t = \delta(t_1, \ldots, t_k)$.

Now, by induction hypothesis on ζ, we have $\zeta_1[\![\ldots]\!]_h \Rightarrow^*_{M,s} t_1, \ldots, \zeta_k[\![\ldots]\!]_h \Rightarrow^*_{M,s} t_k$. Consequently, $\zeta[\![\ldots]\!]_h = \delta(\zeta_1[\![\ldots]\!]_h, \ldots, \zeta_k[\![\ldots]\!]_h) \Rightarrow^*_{M,s} \delta(t_1, \ldots, t_k) = t$.

(ii) Let $\zeta = \langle p, \varphi\rangle(\zeta_1, \ldots, \zeta_k)$ for some $k \geq 0$, $\langle p, \varphi\rangle \in \langle Q, I_d\rangle^{(k)}$, and $\zeta_1, \ldots, \zeta_k \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$. Then, obviously, $comb(\zeta) = c_{k+1}(\langle p', \varphi\rangle, comb(\zeta_1), \ldots, comb(\zeta_k))$. We make the following observations.

(a) There are $l' \leq l$, $l_1, \ldots, l_k < l$, and $\overline{t}', t'_1, \ldots, t'_k \in T_\Gamma$, such that $\langle p', \varphi(h)\rangle \Rightarrow^{l'}_{M',s} \overline{t}'$, $comb(\zeta_1)[\![\ldots]\!]'_h \Rightarrow^{l_1}_{M',s} t'_1, \ldots, comb(\zeta_k)[\![\ldots]\!]'_h \Rightarrow^{l_k}_{M',s} t'_k$,

and $t' = c_{k+1}(\overline{t}', t'_1, \ldots, t'_k)$.

(b) Since $yield_g(t') = t$,

- there is a tree $\overline{t} \in T_\Delta(Y_d)$ such that $yield_g(\overline{t}') = \overline{t}$,
- for every $1 \leq j \leq k$ for which $|\overline{t}|_{y_j} \geq 1$, there is a tree $t_j \in T_\Delta(Y_d)$, such that $yield_g(t'_j) = t_j$.

Moreover, we have

$$\begin{aligned}
yield_g(t') &= yield_g(\overline{t}') \overset{OI}{\leftarrow} (yield_g(t'_1), \ldots, yield_g(t'_k))\\
&= \overline{t}[y_j \leftarrow yield_g(t'_j) \mid 1 \leq j \leq k,\, |\overline{t}|_{y_j} \geq 1]\\
&\ \text{(because } g \text{ is deterministic)}\\
&= \overline{t}[y_j \leftarrow t_j \mid 1 \leq j \leq k,\, |\overline{t}|_{y_j} \geq 1]\\
&= t.
\end{aligned}$$

(c) By induction hypothesis K[l], we have $\langle p, \varphi(h)\rangle(y_1, \ldots, y_k) \Rightarrow^*_{M,s} \overline{t}$.

(d) By induction hypothesis on ζ, for every j with $|\overline{t}|_{y_j} \geq 1$, we have $\zeta_j[\![\ldots]\!]_h \Rightarrow^*_{M,s} t_j$.

Consequently, we obtain

$$\begin{aligned}
\zeta[\![\ldots]\!]_h &= \langle p, \varphi(h)\rangle(\zeta_1[\![\ldots]\!]_h, \ldots, \zeta_k[\![\ldots]\!]_h)\\
&\Rightarrow^*_{M,s} \overline{t}[y_j \leftarrow \zeta_j[\![\ldots]\!]_h \mid 1 \leq j \leq d,\, |\overline{t}|_{y_j} \geq 1]\\
&\Rightarrow^*_{M,s} \overline{t}[y_j \leftarrow t_j \mid 1 \leq j \leq d,\, |\overline{t}|_{y_j} \geq 1]\\
&= t,
\end{aligned}$$

which proves IS2. This finishes also the proof of Statement 2.

Finally, we show that $\tau_M = \tau_{M'} \circ yield_g$, i.e., that, for every input tree $s \in T_\Sigma$, we have $\tau_M(s) = yield_g(\tau_{M'}(s))$.

The inclusion $\tau_M(s) \subseteq yield_g(\tau_{M'}(s))$ follows from K of Statement 1, because for every output tree $t \in T_\Delta$, if $t \in \tau_M(s)$, i.e., $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^l_{M,s} t$ for some $l \geq 1$, then $yield_g(\tau_{M'}(s)) = yield_g(nf(\langle q'_0, (\varepsilon, [\,])\rangle, \Rightarrow_{M',s})) = t$, i.e., $t \in yield_g(\tau_{M'}(s))$.

The inclusion $yield_g(\tau_{M'}(s)) \subseteq \tau_M(s)$ follows from K of Statement 2, because for every output tree $t \in T_\Delta$, if $t \in \mathit{yield}_g(\tau_{M'}(s))$, i.e., there is a $t' \in T_\Gamma$ such that $\langle q_0', (\varepsilon, [\,])\rangle \Rightarrow^l_{M',s} t'$ for some $l \geq 1$ and $\mathit{yield}_g(t') = t$, then $\langle q_0, (\varepsilon, [\,])\rangle \Rightarrow^*_{M,s} t$, i.e., $t \in \tau_M(s)$.

This finishes the proof of Lemma 8.4. $\diamond$

Now we state and prove a variant of Lemma 8.4 for context-linear pmtts. As a preparation for this, we show that the computation steps of a context-linear pmtt preserve the linearity of sentential forms. For (total, deterministic) nondeleting macro tree transducers, a similar preservation lemma is Lemma 6.7 in [EM99].

Lemma 8.5 Assume that the pmtt M is context-linear. For every input tree $s \in T_\Sigma$, $m \geq 0$, configuration $\langle q, h\rangle \in C^{(m)}_{M,s}$, and output tree $t \in T_\Delta(Y_m)$, if $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$, then t is linear in Y_m.

Proof. The proof can be performed by simultaneous induction. To form the predicates K and L, we introduce an abbreviation: for each pebble configuration $h \in IC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, let

$$[\![\ldots]\!]_h = [\![\langle p, \varphi\rangle \leftarrow \langle p, \varphi(h)\rangle(y_1, \ldots, y_{\mathit{rank}(p)}) \mid p \in Q, \varphi \in I_{\sigma,b,j}]\!].$$

Now let K and L be formed as follows.

K(l): For every $m \geq 0$, $q \in Q^{(m)}$, and $t \in T_\Delta(Y_m)$, if $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M,s} t$, then t is linear in Y_m.

L(l): For each tree $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_m)$ which is linear in Y_m, and $t \in T_\Delta(Y_m)$, if $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$, then t is linear in Y_m.

The proof is standard and our lemma follows from K. $\diamond$

Now we can state and prove our second main decomposition result.

Lemma 8.6 If M is context-linear and M' is noncircular, then $\tau_M = \tau_{M'} \circ \mathit{yield}_g$.

Proof. The proof is analogous to the proof of Lemma 8.4. We will use the abbreviations $[\![\ldots]\!]_h$ and $[\![\ldots]\!]'_h$ introduced in that proof.

The equality $\tau_M = \tau_{M'} \circ \mathit{yield}_g$ follows in a standard way from the following two statements. We note, that Statement 1. is formulated differently to the proof of Lemma 8.4 because of lack of determinism of M.

Statement 1.

K(l): For every $m \geq 0$, configuration $\langle q, h\rangle \in C^{(m)}_{M,s}$, and output tree $t \in T_\Delta(Y_m)$, if $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^l_{M,s} t$, then there is a tree $t' \in T_\Gamma$ such that $\langle q', h\rangle \Rightarrow^*_{M',s} t'$ and $\mathit{yield}_g(t') = t$.

L(l): For every pebble configuration $h \in IC_{M,s}$ with $\mathit{test}(h) = (\sigma, b, j)$, trees $\zeta \in T_{\Delta \cup \langle Q, I_{\sigma,b,j}\rangle}(Y_d)$ and $t \in T_\Delta(Y_d)$, if $\zeta[\![\ldots]\!]_h \Rightarrow^l_{M,s} t$, then there is a $t' \in T_\Gamma$, such that $\mathit{comb}(\zeta)[\![\ldots]\!]'_h \Rightarrow^*_{M',s} t'$ and $\mathit{yield}_g(t') = t$.

Statement 2.

K(l): For every $m \geq 0$, configuration $\langle q, h\rangle \in C^{(m)}_{M,s}$, and trees $t' \in T_\Gamma$ and $t \in T_\Delta(Y_m)$, if $\langle q', h\rangle \Rightarrow^l_{M',s} t'$ and $yield_g(t') = t$, then $\langle q, h\rangle(y_1, \ldots, y_m) \Rightarrow^*_{M,s} t$.

L(l): For every pebble configuration $h \in IC_{M,s}$ with $test(h) = (\sigma, b, j)$, and trees $\zeta \in T_{\Delta\cup\langle Q, I_{\sigma,b,j}\rangle}(Y_d)$, $t' \in T_\Gamma$ and $t \in T_\Delta(Y_d)$, if $comb(\zeta)[\![\ldots]\!]'_h \Rightarrow^l_{M',s} t'$ and $yield_g(t') = t$, then $\zeta[\![\ldots]\!]_h \Rightarrow^*_{M,s} t$.

The proof of Statement 1 and 2 is analogous to that of the corresponding two statements in Lemma 8.4. The only difference is the following. In the proof of case (ii) of IS2 of Statement 1. of Lemma 8.4, we used the fact that M is deterministic in decomposing the tree t: different occurrences of a variable y_j in the tree $\bar{t}$ can be substituted by the same tree t_j. Now, in the proof of the corresponding part of IS2 of Statement 1 of the present lemma, we can use that M is context-linear. In fact, by Lemma 8.5, the tree $\bar{t}$ will be linear in Y_k, hence there are no different occurrences of a variable y_j in $\bar{t}$. $\diamond$

Note that the ptt M' of Lemma 8.4 is restricted to be noncircular. If we allowed M' to be not strongly circular, then the proof of this Lemma 8.4 would fail, since in point (c) of IS2 of the Proof of Statement 1 we require that M' will terminate from each sentential form. The same is true for Lemma 8.6.

8.3 Sufficient conditions for M to guarantee that M' is noncircular

Let M' be the total n-pebble tree transducer and $yield_g$ the yield tree transformation which are associated with M (Definition 8.1). As we saw in the previous subsection, it is desirable that M' is noncircular because in that case, provided that M is deterministic or context-linear, the tree transformation τ_M can be decomposed into $\tau_{M'}$ and $yield_g$. We will see later (Corollary 10.10), that it is decidable whether a ptt is circular or not. However, there are some obvious conditions which guarantee that M' is noncircular. Such a condition is that M is a macro tree transducer. In this case we obtain two further variants of the yield decomposition theorems in [EV85].

Corollary 8.7 $dMTT \subseteq dtT \circ dYIELD$

Proof. If the pmtt appearing in Definition 8.1 is a (ordinary) deterministic macro tree transducer, then the pebble tree transducer M' associated with M will be a deterministic and total top-down tree transducer, cf. the note after Definition 8.1. Hence M' cannot be circular and thus, by Lemma 8.4, it holds that $\tau_M = \tau_{M'} \circ yield_g$. $\diamond$

Corollary 8.8 $clMTT \subseteq tT \circ dYIELD$

Proof. If the pmtt appearing in Definition 8.1 is a macro tree transducer, then the pebble tree transducer M' associated with M will be a total top-down tree transducer, cf. the note after Definition 8.1. Hence M' is noncircular. If we even assume that M is context-linear, then the requirements of Lemma 8.6 are fulfilled, hence $\tau_M = \tau_{M'} \circ yield_g$. $\diamond$

In the rest of this subsection, we give a condition for M which is equivalent to that M' is noncircular. Let $M' = (Q', \Sigma, \Gamma, q_0', R')$ be the n-pebble tree transducer associated with M.

Lemma 8.9 M is weakly circular if and only if M' is circular.

Proof. By Lemma 4.5, it is sufficient to show that M is weakly circular if and only if M' is weakly circular.

According to Definition 8.1, for each rule in R', either it has the form $\langle q', \sigma, b, j\rangle \to comb(\zeta)$, where $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta \in R$ or its right-hand side is a tree in T_Γ. Moreover, the state-instruction pairs which occur in $comb(\zeta)$ are $\langle q_1', \varphi_1\rangle, \ldots, \langle q_k', \varphi_k\rangle$ if and only if the ones which occur in ζ are $\langle q_1, \varphi_1\rangle, \ldots, \langle q_k, \varphi_k\rangle$. This implies that, for every input tree $s \in T_\Sigma$,

$$\textit{one-step}_{M',s} = \{(\langle q', h\rangle, \langle p', f\rangle) \mid (\langle q, h\rangle, \langle p, f\rangle) \in \textit{one-step}_{M,s}\}$$

and thus $\textit{one-step}^+_{M',s}$ is the "primed copy" of $\textit{one-step}^+_{M,s}$. $\diamond$

We will prove the decidability of the wc problem in Corollary 10.11.

Now we combine the above circularity results with the decomposition results obtained in Section 8.2. Let us denote the class of tree transformations computed by not weakly circular n-pmtts by $n\text{-}PMTT_{nwc}$. The prefixes d, t, and cl mean the same as before. Then we obtain the following results.

Corollary 8.10

$$n\text{-}dPMTT_{nwc} \subseteq n\text{-}dtPTT_{nc} \circ dYIELD \text{ and}$$

$$n\text{-}clPMTT_{nwc} \subseteq n\text{-}tPTT_{nc} \circ dYIELD.$$

Proof. It immediately follows from Lemmas 8.4, 8.6, and 8.9. $\diamond$

Theorem 8.11

$$n\text{-}dPMTT_{nwc} \subseteq n\text{-}dtPTT_{nc} \circ 0\text{-}dPTT_{nc} \text{ and}$$

$$n\text{-}clPMTT_{nwc} \subseteq n\text{-}tPTT_{nc} \circ 0\text{-}dPTT_{nc}.$$

Proof. It follows from Corollary 8.10 and Theorem 5.2. $\diamond$

In Section 8 of [EM03] it was conjectured that the composition closure of pebble macro tree transformations and of pebble tree transformations coincide, i.e., $PMTT^* = PTT^*$.

The following theorem says that (under certain circumstances) for the not weakly circular subclasses of $PMTT$ and PTT, the above conjecture holds.

Theorem 8.12 For every $n \geq 0$,

$$n\text{-}dPMTT^*_{nwc} = n\text{-}dPTT^*_{nc} \text{ and}$$

$$n\text{-}clPMTT^*_{nwc} = n\text{-}PTT^*_{nc}.$$

Proof. It follows from Lemma 4.5, Theorem 8.11, and the fact that every 0-ptt is an n-ptt. ◇

In [KV94], macro attributed tree transducers were introduced as a combination of macro tree transducers of [EV85] and of attributed tree transducers of [Fül81]. Moreover, it was shown that the class of tree transformations computed by (noncircular) macro attributed tree transducers is equal to the two-fold composition of the class of tree transformations computed by attributed tree transducers. In [EM03] they observed that this, for the pebble approach, suggests $0\text{-}dPMTT \subseteq (0\text{-}dPTT)^2$ and $0\text{-}PMTT \subseteq (0\text{-}PTT)^2$ and asked whether this holds or not. As consequence of Theorem 8.11, we obtain a partial solution of this open problem. In fact, we can decompose only not weakly circular deterministic 0-pmtts and not weakly circular context-linear (nondeterministic) 0-pmtts. However, we can state that the resulting ptts in both cases will be noncircular. We have summarized this in the following corollary.

Corollary 8.13

$$0\text{-}dPMTT_{nwc} \subseteq (0\text{-}dPTT_{nc})^2 \text{ and}$$

$$0\text{-}clPMTT_{nwc} \subseteq (0\text{-}PTT_{nc})^2.$$

Proof. Apply Theorem 8.11 for $n = 0$. ◇

Finally, combining Corollary 6.2 and the results of the present section, we can prove more characterization results for restricted pmtts.

Corollary 8.14 For each $n \geq 0$ we have that $n\text{-}dPMTT_{nwc} \subseteq n\text{-}dtPTT_{nc} \circ dYIELD \subseteq n\text{-}dPMTT_{nc}$.

Proof. The inclusion $n\text{-}dPMTT_{nwc} \subseteq n\text{-}dtPTT_{nc} \circ dYIELD$ follows from Corollary 8.10 and the inclusion $n\text{-}dtPTT_{nc} \circ dYIELD \subseteq n\text{-}dPMTT_{nc}$ follows from Corollary 6.2. ◇

As usual, putting together composition and decomposition results, one gets a characterization of a tree transformation class in terms of the composition of other, simpler ones. If we do that with the decomposition results of Section 7 and the composition results of this section, we obtain the following characterization of deterministic (OI) macro tree transformations.

Corollary 8.15 $dMTT = dtT \circ dYIELD$.

Proof. The inclusion $dMTT \subseteq dtT \circ dYIELD$ comes from Corollary 8.7, while the reverse inclusion can be obtained from Corollary 6.2. ◇

We note that the above equation is a slight generalization of the well know equality $dtMTT = dtT \circ dtYIELD$, which was proved in [EV85]. It is interesting that the decomposition and composition results concerning pmtts result only in a characterization of deterministic macro tree transformations.

9 Simulation of n-ptts by $(n-1)$-pmtts

In Section 8 of [EM03] the question was raised whether an n-ptt can be simulated by an $(n-1)$-pmtt (both with weak pebbles), i.e., whether the power of the last pebble can be replaced by macro calls. In this section we show that the answer is positive even in the strong pebble case, i.e., for each n-ptt M, we effectively construct an $(n-1)$-pmtt M' such that $\tau_M = \tau_{M'}$. Results of this section can be found in Section 5 of [FM09].

In order to make the proof simpler, we will assume that our n-ptt $M = (Q, \Sigma, \Delta, q_0, R)$ is in *normal form*, meaning that, for each rule $\langle q, \sigma, b, j\rangle \to \zeta \in R$, we have either $\zeta = \langle q', \varphi\rangle$ (pebble tree-walking rule) or $\zeta = \delta(\langle q_1, \mathit{stay}\rangle, \ldots, \langle q_k, \mathit{stay}\rangle)$ (output rule). In Lemma 2 of [EM03] it was proved that ptts and ptts in normal form, both with weak pebble handling, have the same computation power. The proof can obviously be adapted for strong pebble handling, hence the following holds.

Proposition 9.1 For each $n \geq 0$ and n-ptt M we can effectively construct an n-ptt M' in normal form such that $\tau_M = \tau_{M'}$. ◇

First we prove the simulation result for $n = 1$. To make the construction and the proof more readable, we drop $b = \varepsilon$ from the left-hand side of a rule of a 0-pmtt and write such a rule in the form $\langle q, \sigma, j\rangle(y_1, \ldots, y_m) \to \zeta$. Moreover, we drop the second component of every pebble configuration $(u, [\,])$ of a 0-pmtt and denote that pebble configuration just by u. Finally, we write the pebble configurations $(u, [\,])$ and $(u, [u'])$ of a 1-ptt as $(u, -)$ and (u, u'), respectively.

For a given 1-ptt $M = (Q, \Sigma, \Delta, q_0, R)$ being in normal form, we associate a 0-pmtt M', called the 0-pmtt associated with M, such that $\tau_M = \tau_{M'}$. In the following, we describe intuitively how M' works. Let us assume that $Q = \{q_0, q_1, \ldots, q_m\}$, where $m \geq 0$.

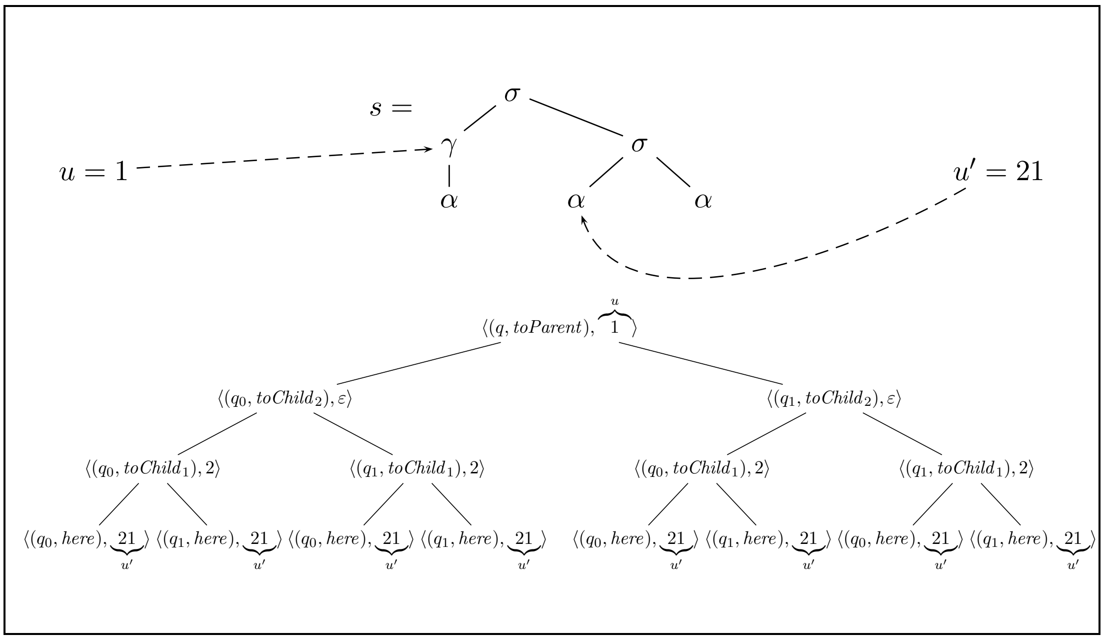
(A) Every state of M' has rank 0 or $m+1$. The states with rank 0 are the states in Q and the pairs (q, here), while the states with rank $m+1$ are the pairs $(q, \mathit{toParent})$ and $(q, \mathit{toChild}_\omega)$, where $q \in Q$ and $1 \leq \omega \leq \mathit{maxr}(\Sigma)$.

(B) For each configuration of the form $\langle q, (u, -)\rangle$ of M, $\langle q, u\rangle$ is a configuration also of M' and each pebble-free computation step of M is also made by M'.

(C) Whenever M is in configuration $\langle q, (u, u')\rangle$ (i.e., M is in state q, the pointer of M is at node u and the pebble is dropped at node u' of the current input tree), the pmtt M' realizes this configuration as a special sentential form which contains no output symbols. We call such a sentential form the *path tree of* $\langle q, (u, u')\rangle$ and denote it by $\mathit{path\_tree}(\langle q, (u, u')\rangle)$. The construction of $\mathit{path\_tree}(\langle q, (u, u')\rangle)$ follows the idea applied in the proof of Lemma 34 of [EM03], where 0-ptts were simulated by smtts.

In fact, $\mathit{path\_tree}(\langle q, (u, u')\rangle)$ has the following properties.

- It is a fully balanced $m+1$-ary tree of configurations of M', the height of which is equal to the number of the nodes in the shortest path from u to u'. Recall that $m+1$ is the number of the states of Q. Each node of $\mathit{path\_tree}(\langle q, (u, u')\rangle)$ is a configuration of the form $\langle(\overline{q}, d), \overline{u}\rangle$, where $\overline{q}$ is a state of M, $\overline{u}$ is a node in the path from u to u', and d is a *direction flag* detailed later.



$s =$
$u = 1$
$u' = 21$
σ
γ
σ
α
α
α
$\langle (q, \mathit{toParent}), \overbrace{1}^{u} \rangle$
$\langle (q_0, \mathit{toChild}_2), \varepsilon \rangle$
$\langle (q_1, \mathit{toChild}_2), \varepsilon \rangle$
$\langle (q_0, \mathit{toChild}_1), 2 \rangle$
$\langle (q_1, \mathit{toChild}_1), 2 \rangle$
$\langle (q_0, \mathit{here}), \underbrace{21}_{u'} \rangle$
$\langle (q_1, \mathit{here}), \underbrace{21}_{u'} \rangle$

Figure 5: An example of an input tree $s \in T_\Sigma$, a configuration $\langle q, (u, u') \rangle \in C_{M,s}$, and $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ where $Q = \{q_0, q_1\}$ (hence, $m = 1$).

- The configuration being at the root of $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ has the form $\langle (q, d), \overline{u} \rangle$.
- For each non-leaf configuration $\langle (\overline{q}, d), \overline{u} \rangle$ in $\mathit{path\_tree}(\langle q, (u, u') \rangle)$, the states of M being in the $m + 1$ children of $\langle (\overline{q}, d), \overline{u} \rangle$ are $q_0, q_1, \ldots, q_m$, respectively.
- The direction flag d of each configuration $\langle (\overline{q}, d), \overline{u} \rangle$ in $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ can be
  - *here*, if $\overline{u}$ is equal to u';
  - *toParent*, if the next node in the path from $\overline{u}$ to u' is the parent node of $\overline{u}$;
  - $\mathit{toChild}_\omega$ for some $1 \leq \omega \leq \mathit{maxr}(\Sigma)$, if the next node in the path from $\overline{u}$ to u' is the ω-th child of $\overline{u}$.
- For each $1 \leq i \leq \mathit{height}(\mathit{path\_tree}(\langle q, (u, u') \rangle))$, the node-components and the direction flags of each configuration on the i-th level of $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ are the same $\overline{u}$ and d, respectively, where $\overline{u}$ is the i-th node of the path from u to u' and d is the direction of the first step from $\overline{u}$ towards u'. Hence, at the root of $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ we have $\overline{u} = u$, while at the leaves of $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ have node $\overline{u} = u'$ and $d = \mathit{here}$.

An example of a configuration $\langle q, (u, u') \rangle \in C_{M,s}$ and of $\mathit{path\_tree}(\langle q, (u, u') \rangle)$ can be seen in Fig. 5.

(D) M' simulates each computation step $\langle q, h \rangle \Rightarrow_{M,s} \langle q', h' \rangle$ of M by computing $\mathit{path\_tree}(\langle q', h' \rangle)$ from $\mathit{path\_tree}(\langle q, h \rangle)$. Note that the path tree of a configuration with no pebble is the configuration itself.

Now we define M' formally.

Definition 9.2 Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a 1-ptt in normal form and $Q = \{q_0, \dots, q_m\}$. The *0-pmtt associated with* M is the 0-pmtt $M' = (Q', \Sigma, \Delta, q_0, R')$, where

$Q' =$
$Q \cup \{(q, \mathit{here})^{(0)}, (q, \mathit{toParent})^{(m+1)}, (q, \mathit{toChild}_\omega)^{(m+1)} \mid q \in Q,\ 1 \le \omega \le \mathit{maxr}(\Sigma)\}$

and R' is the smallest rule set satisfying the following conditions. Let $\sigma \in \Sigma$, $0 \le j \le \mathit{maxr}(\Sigma)$, $k \ge 0$, $q, p, p_1, \dots, p_k \in Q$, and $\delta \in \Delta^{(k)}$.

(r_1) If $\langle q, \sigma, \varepsilon, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \dots, \langle p_k, \mathit{stay}\rangle)$ is in R, then the rule

- $\langle q, \sigma, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \dots, \langle p_k, \mathit{stay}\rangle)$

is in R'. (If the pebble is not placed on the input and M outputs symbol, then so does M'.)

(r_2) If $\langle q, \sigma, \varepsilon, j\rangle \to \langle p, \varphi\rangle$ is in R for $\varphi \in \{\mathit{stay}, \mathit{up}, \mathit{down}_i \mid 1 \le i \le \mathit{maxr}(\Sigma)\}$, then the rule

- $\langle q, \sigma, j\rangle \to \langle p, \varphi\rangle$

is in R'. (If the pebble is not placed on the input and M moves the pointer to a direction, then so does M'.)

(r_3) If $\langle q, \sigma, \varepsilon, j\rangle \to \langle p, \mathit{drop}\rangle$ is in R, then the rule

- $\langle q, \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$

is in R'. (If M drops the pebble then M' drops a "virtual pebble" by labelling the current state by *here*.);

($r_4/1$) If $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{lift}\rangle$ is in R, then the rule

- $\langle (q, \mathit{here}), \sigma, j\rangle \to \langle p, \mathit{stay}\rangle$

is in R'.

($r_4/0$) If $\langle q, \sigma, 0, j\rangle \to \langle p, \mathit{lift}\rangle$ is in R, then the rules

- $\langle (q, d), \sigma, j\rangle(y_1, \dots, y_{m+1}) \to \langle p, \mathit{stay}\rangle$

are in R', for each direction $d \in \{\mathit{toParent}, \mathit{toChild}_\omega \mid 1 \le \omega \le \mathit{maxr}(\Sigma)\}$. (If M lifts the pebble, then M' lifts the "virtual pebble" by removing the direction flag d (or *here*) from the current state and, if any, deleting the subtrees of the current configuration.);

($r_5/1$) If $\langle q, \sigma, 1, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \dots, \langle p_k, \mathit{stay}\rangle)$ is in R, then the rule

- $\langle (q, \mathit{here}), \sigma, j\rangle \to \delta(\langle (p_1, \mathit{here}), \mathit{stay}\rangle, \dots, \langle (p_k, \mathit{here}), \mathit{stay}\rangle)$

is in R'.

$(r_5/0)$ If $\langle q, \sigma, 0, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \dots, \langle p_k, \mathit{stay}\rangle)$ is in R, then the rules

- $\langle (q, d), \sigma, j\rangle(y_1, \dots, y_{m+1}) \to$
  $\delta(\langle (p_1, d), \mathit{stay}\rangle(y_1, \dots, y_{m+1}), \dots, \langle (p_k, d), \mathit{stay}\rangle(y_1, \dots, y_{m+1}))$

are in R', for each direction $d \in \{\mathit{toParent}, \mathit{toChild}_\omega \mid 1 \le \omega \le \mathit{maxr}(\Sigma)\}$. (If the pebble is on the input and M writes an output symbol, then so does M'.);

$(r_6/1)$ If $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{stay}\rangle$ is in R, then the rule

- $\langle (q, \mathit{here}), \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$

is in R'.

$(r_6/0)$ If $\langle q, \sigma, 0, j\rangle \to \langle p, \mathit{stay}\rangle$ is in R, then the rules

- $\langle (q, d), \sigma, j\rangle(y_1, \dots, y_{m+1}) \to \langle (p, d), \mathit{stay}\rangle(y_1, \dots, y_{m+1})$

are in R', for each direction $d \in \{\mathit{toParent}, \mathit{toChild}_\omega \mid 1 \le \omega \le \mathit{maxr}(\Sigma)\}$. (If the pebble is on the input and M stays at the current node, then so does M' and the direction flag remains the same.)

$(r_7/1)$ If $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{up}\rangle$ is in R, then the rule

- $\langle (q, \mathit{here}), \sigma, j\rangle \to$
  $\langle (p, \mathit{toChild}_j), \mathit{up}\rangle(\langle (q_0, \mathit{here}), \mathit{stay}\rangle, \dots, \langle (q_m, \mathit{here}), \mathit{stay}\rangle)$

is in R'.

$(r_7/0)$ If $\langle q, \sigma, 0, j\rangle \to \langle p, \mathit{up}\rangle$ is in R, then the rules

- $\langle (q, \mathit{toParent}), \sigma, j\rangle(y_1, \dots, y_{m+1}) \to y_{\nu+1}$, where $p = q_\nu$
- $\langle (q, \mathit{toChild}_\omega), \sigma, j\rangle(y_1, \dots, y_{m+1}) \to \langle (p, \mathit{toChild}_j), \mathit{up}\rangle($
  $\langle (q_0, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \dots, y_{m+1}), \dots,$
  $\langle (q_m, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \dots, y_{m+1}))$

are in R', where $1 \le \omega \le \mathit{maxr}(\Sigma)$. (The pointer is at node u with child number j and M moves *up*. If the pebble is not upwards of u, then M' also moves *up*, annotates its state by direction $\mathit{toChild}_j$, and spawns $m + 1$ child configurations with node component u and with all the possible states labeled by the previous direction flag, for the case of returning to u. Otherwise, M' gets closer to the "virtual pebble" by returning the parameter with the appropriate state.)

$(r_8/1)$ If $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{down}_i\rangle$ is in R, then the rule

- $\langle (q, \mathit{here}), \sigma, j\rangle \to$
  $\langle (p, \mathit{toParent}), \mathit{stay}\rangle(\langle (q_0, \mathit{here}), \mathit{stay}\rangle, \dots, \langle (q_m, \mathit{here}), \mathit{stay}\rangle)$

is in R'.

$(r_8/0)$ If $\langle q, \sigma, 0, j\rangle \to \langle p, down_i\rangle$ is in R, then the rules

- $\langle (q, toParent), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to \langle (p, toParent), down_i\rangle($
  $\langle (q_0, toParent), stay\rangle(y_1, \ldots, y_{m+1}), \ldots, \langle (q_m, toParent), stay\rangle(y_1, \ldots, y_{m+1}))$
- $\langle (q, toChild_\omega), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to \langle (p, toParent), down_i\rangle($
  $\langle (q_0, toChild_\omega), stay\rangle(y_1, \ldots, y_{m+1}), \ldots, \langle (q_m, toChild_\omega), stay\rangle(y_1, \ldots, y_{m+1}))$

are in R' for every $1 \le \omega \le maxr(\Sigma)$ with $\omega \ne i$, moreover, the rule

- $\langle (q, toChild_i), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to y_{\nu+1}$, where $p = q_\nu$

is also in R'. (These two kinds of rules correspond to $(r_7/1)$ and $(r_7/0)$, respectively, in the case that M moves down.) ◇

We make the following observations concerning M' in Definition 9.2.

Observation 9.3 (a) The construction of M' preserves determinism. Moreover, (b) the construction of M' works for 1-ptts with weak pebble handling because these latters are special 1-ptts.

Before proving the main result of this section, we give the formal definition of *path_tree*.

Definition 9.4 Let $M = (Q, \Sigma, \Delta, R)$ be a 1-ptt in normal form, where $Q = \{q_0, \ldots, q_m\}$, $s \in T_\Sigma$ an input tree, and M' the 0-pmtt associated with M as in Definition 9.2. Then $path\_tree_{M,s} : C_{M,s} \to T_{C_{M',s}}$ is the mapping such that, for each state $q \in Q$ and nodes $u, u' \in pos(s)$,

(i) $path\_tree_{M,s}(\langle q, (u, -)\rangle) = \langle q, u\rangle$;

(ii) if $u = u'$, then $path\_tree_{M,s}(\langle q, (u, u')\rangle) = \langle (q, here), u\rangle$;

(iii) if $u' = u\omega u''$ for some $1 \le \omega \le maxr(\Sigma)$ and $u'' \in \mathbb{N}^*$, i.e., u' is a proper descendant of u and it is in the ω-th subtree of u, then

$$path\_tree_{M,s}(\langle q, (u, u')\rangle) = \langle (q, toChild_\omega), u\rangle(path\_tree_{M,s}(\langle q_0, (down_\omega(u), u')\rangle), \ldots, path\_tree_{M,s}(\langle q_m, (down_\omega(u), u')\rangle));$$

(iv) if the longest common prefix of u and u' is a proper prefix of u, i.e., the deepest common ancestor of u and u' is above u, then

$$path\_tree_{M,s}(\langle q, (u, u')\rangle) = \langle (q, toParent), u\rangle(path\_tree_{M,s}(\langle q_0, (up(u), u')\rangle), \ldots, path\_tree_{M,s}(\langle q_m, (up(u), u')\rangle)).$$

If M and s are clear from the context then we write $path\_tree(\langle q, (u, u')\rangle)$ rather than $path\_tree_{M,s}(\langle q, (u, u')\rangle)$. ◇

Now we can prove the following result.

Lemma 9.5 *1-PTT* $\subseteq$ *0-PMTT* and *1-dPTT* $\subseteq$ *0-dPMTT*.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a 1-ptt and $Q = \{q_0, \ldots, q_m\}$. By Proposition 9.1 we can assume that M is in normal form. Moreover, let $M' = (Q', \Sigma, \Delta, q_0, R')$ be the 0-pmtt associated with M. If M is deterministic, then M' is also deterministic by Observation 9.3(a). We show that $\tau_M = \tau_{M'}$. For this, we fix an input tree $s \in T_\Sigma$ and show that $\tau_M(s) = \tau_{M'}(s)$ holds. In fact, we prove the following two statements.

Statement 1. For each $l \geq 1$, $\langle q, h\rangle \in C_{M,s}$, and $t \in T_\Delta$, if $\langle q, h\rangle \Rightarrow^l_{M,s} t$ then $path\_tree(\langle q, h\rangle) \Rightarrow^*_{M',s} t$.

Statement 2. For each $l \geq 1$, $\langle q, h\rangle \in C_{M,s}$, and $t \in T_\Delta$, if $path\_tree(\langle q, h\rangle) \Rightarrow^l_{M',s} t$ then $\langle q, h\rangle \Rightarrow^*_{M,s} t$.

Then, the inclusion $\tau_M(s) \subseteq \tau_{M'}(s)$ can be obtained as follows. Let $t \in \tau_M(s)$, then there is an $l \geq 0$ such that $\langle q_0, (\varepsilon, -)\rangle \Rightarrow^l_{M,s} t$. Hence, by Definition 9.4(i) and Statement 1 we get $path\_tree(\langle q_0, (\varepsilon, -)\rangle) = \langle q_0, \varepsilon\rangle \Rightarrow^*_{M',s} t$ and thus, $t \in \tau_{M'}(s)$.

Similarly, we obtain the inclusion $\tau_{M'}(s) \subseteq \tau_M(s)$ from Statement 2.

In the rest of the proof we use the following notation and abbreviation. We denote by u and u' arbitrary nodes of s. Moreover, we refer shortly by (r_i), $1 \leq i \leq 4$, and (r_i/j), $4 \leq i \leq 8$ and $0 \leq j \leq 1$, to the corresponding part of Definition 9.2. Finally, we abbreviate the expression induction hypothesis by IH. We give the IH explicitly only in the proof of Subcase 1.1 of Statements 1 and 2. Later we use IH implicitly.

Proof of Statement 1. Assume that $\langle q, h\rangle \Rightarrow^l_{M,s} t$ and $test(h) = (\sigma, b, j)$. We prove by induction on l.

(i) Let $l = 1$. Since M is in normal form, there is an output rule $\langle q, \sigma, b, j\rangle \to \delta$ in R, such that $t = \delta$. We distinguish the following three cases, according to the shape of h.

Case 1: $h = (u, -)$. Then $b = \varepsilon$ and, by Definition 9.4(i), we have $path\_tree(\langle q, h\rangle) = \langle q, u\rangle$. Moreover, by (r_1), the rule $\langle q, \sigma, j\rangle \to \delta$ is in R', which concludes that $path\_tree(\langle q, h\rangle) \Rightarrow_{M',s} t$.

Case 2: $h = (u, u)$. Then $b = 1$ and, by Definition 9.4(ii), we have $path\_tree(\langle q, h\rangle) = \langle (q, here), u\rangle$. Moreover, by $(r_5/1)$ the rule $\langle (q, here), \sigma, j\rangle \to \delta$ is in R', which concludes that $path\_tree(\langle q, h\rangle) \Rightarrow_{M',s} t$.

Case 3: $h = (u, u')$ such that $u \neq u'$. Then $b = 0$ and, by Definition 9.4(iii)-(iv), the root of $path\_tree(\langle q, h\rangle)$ is $\langle (q, d), u\rangle$ for some direction $d \in \{toParent, toChild_\omega \mid 1 \leq \omega \leq maxr(\Sigma)\}$. Moreover, by $(r_5/0)$ the rule $\langle (q, d), \sigma, b, j\rangle(y_1, \ldots, y_{m+1}) \to \delta$ is in R', which concludes that $path\_tree(\langle q, h\rangle) \Rightarrow_{M',s} t$.

(ii) Assume that $l > 1$. We distinguish four cases, according to the shape of h.

Case 1: $h = (u, -)$. Then $b = \varepsilon$ and, by Definition 9.4(i), $path\_tree(\langle q, h\rangle) = \langle q, u\rangle$. Now we distinguish three subcases, according to the form of the rule applied in the first step of the computation of t.

Subcase 1.1: $\langle q, h\rangle \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle) \Rightarrow^{l-1}_{M,s} t$. We observe the following.

(a) The rule $\langle q, \sigma, \varepsilon, j\rangle \to \delta(\langle p_1, stay\rangle, \ldots, \langle p_k, stay\rangle)$ is in R. Hence, by (r_1), the rule $\langle q,\sigma,j\rangle \to \delta(\langle p_1, stay\rangle,\dots,\langle p_k, stay\rangle)$ is in R', which implies that $\langle q,u\rangle \Rightarrow_{M',s} \delta(\langle p_1,u\rangle,\dots,\langle p_k,u\rangle)$.

(b) There are integers $l_1,\dots,l_k \le l-1$ and trees $t_1,\dots,t_k \in T_\Delta$ such that, $t = \delta(t_1,\dots,t_k)$ and $\langle p_1,h\rangle \Rightarrow^{l_1}_{M,s} t_1,\dots,\langle p_k,h\rangle \Rightarrow^{l_k}_{M,s} t_k$.

(c) By Definition 9.4(i), $path\_tree(\langle p_1,h\rangle) = \langle p_1,u\rangle,\dots,path\_tree(\langle p_k,h\rangle) = \langle p_k,u\rangle$.

(d) By the induction hypothesis

$$path\_tree(\langle p_1,h\rangle) \Rightarrow^*_{M',s} t_1,\dots,path\_tree(\langle p_k,h\rangle) \Rightarrow^*_{M',s} t_k.$$

Hence, we conclude that

$$\begin{array}{lll}
path\_tree(\langle q,h\rangle) & & \\
\quad = & \langle q,u\rangle & \\
\quad \Rightarrow_{M',s} & \delta(\langle p_1,u\rangle,\dots,\langle p_k,u\rangle) & \text{(by (a))}\\
\quad = & \delta(path\_tree(\langle p_1,h\rangle),\dots,path\_tree(\langle p_k,h\rangle)) & \text{(by (c))}\\
\quad \Rightarrow^*_{M',s} & \delta(t_1,\dots,t_k) & \text{(by (d))}\\
\quad = & t. &
\end{array}$$

Subcase 1.2: $\langle q,h\rangle \Rightarrow_{M,s} \langle p,\varphi(h)\rangle \Rightarrow^{l-1}_{M,s} t$, where $\varphi \in \{stay, up, down_i \mid 1 \le i \le rank(\sigma)\}$.

(a) There is a rule $\langle q,\sigma,\varepsilon,j\rangle \to \langle p,\varphi\rangle$ in R. By (r_2), the rule $\langle q,\sigma,j\rangle \to \langle p,\varphi\rangle$ is in R', which implies $\langle q,u\rangle \Rightarrow_{M',s} \langle p,\varphi(u)\rangle$.

(b) It follows from Definition 9.4(i) that $path\_tree(\langle p,\varphi(h)\rangle) = \langle p,\varphi(u)\rangle$ (recall that $h=(u,-)$).

Thus we obtain

$$\begin{array}{rcll}
path\_tree(\langle q,h\rangle) & = & \langle q,u\rangle & \\
& \Rightarrow_{M',s} & \langle p,\varphi(u)\rangle & \text{(by (a))}\\
& = & path\_tree(\langle p,h\rangle) & \text{(by (b))}\\
& \Rightarrow^*_{M',s} & t. & \text{(by IH)}
\end{array}$$

Subcase 1.3: $\langle q,h\rangle \Rightarrow_{M,s} \langle p, drop(h)\rangle \Rightarrow^{l-1}_{M,s} t$. Note that $drop(h) = (u,u)$.

(a) The rule $\langle q,\sigma,\varepsilon,j\rangle \to \langle p, drop\rangle$ is in R. Then, by (r_3), the rule $\langle q,\sigma,j\rangle \to \langle (p,here), stay\rangle$ is in R', hence $\langle q,u\rangle \Rightarrow_{M',s} \langle (p,here),u\rangle$.

(b) By Definition 9.4(ii), $path\_tree(\langle p, drop(h)\rangle) = \langle (p,here),u\rangle$.

Hence we obtain

$$\begin{array}{rcll}
path\_tree(\langle q,h\rangle) & = & \langle q,u\rangle & \\
& \Rightarrow_{M',s} & \langle (p,here),u\rangle & \text{(by (a))}\\
& = & path\_tree(\langle p, drop(h)\rangle) & \text{(by (b))}\\
& \Rightarrow^*_{M',s} & t. & \text{(by IH)}
\end{array}$$

Case 2: $h=(u,u)$. Then necessarily $b=1$ and $path\_tree(\langle q,h\rangle) = \langle (q,here),u\rangle$. We distinguish five subcases, according to the form of the rule applied in the first step of the derivation.

Subcase 2.1: $\langle q, h\rangle \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \dots, \langle p_k, h\rangle) \Rightarrow^{l-1}_{M,s} t$.

(a) The rule $\langle q, \sigma, 1, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \dots, \langle p_k, \mathit{stay}\rangle)$ is in R. Hence, by $(r_5/1)$, the rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \delta(\langle (p_1, \mathit{here}), \mathit{stay}\rangle, \dots, \langle (p_k, \mathit{here}), \mathit{stay}\rangle)$ is in R', which implies $\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \delta(\langle (p_1, \mathit{here}), u\rangle, \dots, \langle (p_k, \mathit{here}), u\rangle)$.

(b) There are integers $l_1, \dots, l_k \leq l-1$ and trees $t_1, \dots, t_k \in T_\Sigma$ such that, $t = \delta(t_1, \dots, t_k)$ and $\langle p_1, h\rangle \Rightarrow^{l_1}_{M,s} t_1, \dots, \langle p_k, h\rangle \Rightarrow^{l_k}_{M,s} t_k$.

(c) By Definition 9.4(ii), $\mathit{path\_tree}(\langle p_1, h\rangle) = \langle (p_1, \mathit{here}), u\rangle, \dots, \mathit{path\_tree}(\langle p_k, h\rangle) = \langle (p_k, \mathit{here}), u\rangle$.

Hence it follows that

$$\begin{array}{lll}
\multicolumn{3}{l}{\mathit{path\_tree}(\langle q, h\rangle)} \\
= & \langle (q, \mathit{here}), u\rangle & \\
\Rightarrow_{M',s} & \delta(\langle (p_1, \mathit{here}), u\rangle, \dots, \langle (p_k, \mathit{here}), u\rangle) & \text{(by (a))} \\
= & \delta(\mathit{path\_tree}(\langle p_1, h\rangle), \dots, \mathit{path\_tree}(\langle p_k, h\rangle)) & \text{(by (c))} \\
\Rightarrow^*_{M',s} & \delta(t_1, \dots, t_k) & \text{(by IH)} \\
= & t. &
\end{array}$$

Subcase 2.2: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, h\rangle \Rightarrow^{l-1}_{M,s} t$.

(a) There is a rule $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{stay}\rangle$ in R. By $(r_6/1)$, the rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$ is in R', hence $\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \langle (p, \mathit{here}), u\rangle$.

(b) It follows from Definition 9.4(ii) that $\mathit{path\_tree}(\langle p, h\rangle) = \langle (p, \mathit{here}), u\rangle$.

Hence, we conclude that

$$\begin{array}{rcll}
\mathit{path\_tree}(\langle q, h\rangle) & = & \langle (q, \mathit{here}), u\rangle & \\
& \Rightarrow_{M',s} & \langle (p, \mathit{here}), u\rangle & \text{(by (a))} \\
& = & \mathit{path\_tree}(\langle p, h\rangle) & \text{(by (b))} \\
& \Rightarrow^*_{M',s} & t. & \text{(by IH)}
\end{array}$$

Subcase 2.3: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \mathit{up}(h)\rangle \Rightarrow^{l-1}_{M,s} t$.

(a) There is a rule $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{up}\rangle$ in R. By $(r_7/1)$, the rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \langle (p, \mathit{toChild}_j), \mathit{up}\rangle(\langle (q_0, \mathit{here}), \mathit{stay}\rangle, \dots, \langle (q_m, \mathit{here}), \mathit{stay}\rangle)$ is in R', which implies

$$\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(\langle (q_0, \mathit{here}), u\rangle, \dots, \langle (q_m, \mathit{here}), u\rangle).$$

(b) It follows from Definition 9.4(iii) that

$$\mathit{path\_tree}(\langle p, h\rangle) = \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(\langle (q_0, \mathit{here}), u\rangle, \dots, \langle (q_m, \mathit{here}), u\rangle).$$

Hence, we conclude that

$$\begin{array}{rcll}
\mathit{path\_tree}(\langle q, h\rangle) & = & \langle (q, \mathit{here}), u\rangle & \\
& \Rightarrow_{M',s} & \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(& \\
& & \quad \langle (q_0, \mathit{here}), u\rangle, \dots, \langle (q_m, \mathit{here}), u\rangle) & \text{(by (a))} \\
& = & \mathit{path\_tree}(\langle p, \mathit{up}(h)\rangle) & \text{(by (b))} \\
& \Rightarrow^*_{M',s} & t. & \text{(by IH)}
\end{array}$$

Subcase 2.4: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, down_i(h)\rangle \Rightarrow^{l-1}_{M,s} t$ for some $1 \leq i \leq rank(\sigma)$.

(a) There is a rule $\langle q, \sigma, 1, j\rangle \to \langle p, down_i\rangle$ in R. By $(r_8/1)$, the rule $\langle (q, here), \sigma, j\rangle \to \langle (p, toParent), down_i\rangle(\langle (q_0, here), stay\rangle, \ldots, \langle (q_m, here), stay\rangle)$ is in R', which implies

$$\langle (q, here), u\rangle \Rightarrow_{M',s} \langle (p, toParent), down_i(u)\rangle(\langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle).$$

(b) It follows from Definition 9.4(iii) that

$$path\_tree(\langle p, h\rangle) = \langle (p, toParent), down_i(u)\rangle(\langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle).$$

Hence we conclude that

$$\begin{array}{rcll} path\_tree(\langle q, h\rangle) & = & \langle (q, here), u\rangle & \\ & \Rightarrow_{M',s} & \langle (p, toParent), down_i(u)\rangle(& \\ & & \quad \langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle) & \text{(by (a))} \\ & = & path\_tree(\langle p, down_i(h)\rangle) & \text{(by (b))} \\ & \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Subcase 2.5: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, lift(h)\rangle \Rightarrow^{l-1}_{M,s} t$. Note that $lift(h) = (u, -)$.

(a) The rule $\langle q, \sigma, 1, j\rangle \to \langle p, lift\rangle$ is in R. By $(r_4/1)$, the rule $\langle (q, here), \sigma, j\rangle \to \langle p, stay\rangle$ is in R', which implies $\langle (q, here), u\rangle \Rightarrow_{M',s} \langle p, u\rangle$.

(b) By Definition 9.4(i), we have $path\_tree(\langle p, lift(h)\rangle) = \langle p, u\rangle$.

Thus we get

$$\begin{array}{rcll} path\_tree(\langle q, h\rangle) & = & \langle (q, here), u\rangle & \\ & \Rightarrow_{M',s} & \langle p, u\rangle & \text{(by (a))} \\ & = & path\_tree(\langle p, lift(h)\rangle) & \text{(by (b))} \\ & \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Case 3: $h = (u, u')$ such that $u' = u\omega u''$ for some $1 \leq \omega \leq maxr(\Sigma)$ and $u'' \in \mathbb{N}^*$. Then necessarily $b = 0$ and

$$\begin{aligned} path\_tree(\langle q, h\rangle) = \langle (q, toChild_\omega), u\rangle(& \\ path\_tree(\langle q_0, down_\omega(h)\rangle), \ldots, path\_tree(\langle q_m, down_\omega(h)\rangle)). & \quad (9.1) \end{aligned}$$

Now we distinguish six subcases, according to the type of the rule applied in the first step of the derivation.

Subcase 3.1: $\langle q, h\rangle \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle) \Rightarrow^{l-1}_{M,s} t$.

(a) The rule $\langle q, \sigma, 0, j\rangle \to \delta(\langle p_1, stay\rangle, \ldots, \langle p_k, stay\rangle)$ is in R. Hence, by $(r_5/0)$, the rule

$$\begin{aligned} &\langle (q, toChild_\omega), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to \\ &\qquad \delta(\langle (p_1, toChild_\omega), stay\rangle(y_1, \ldots, y_{m+1}), \ldots, \\ &\qquad\qquad \langle (p_k, toChild_\omega), stay\rangle(y_1, \ldots, y_{m+1})) \end{aligned}$$

is in R'. Moreover, it follows from (9.1) that

$$\begin{aligned} \mathit{path\_tree}(\langle q,h\rangle) \quad \Rightarrow_{M',s} \quad & \delta(\langle(p_1,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots, \\ & \qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)) \\ & \vdots \\ & \langle(p_k,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots, \\ & \qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle))). \end{aligned}$$

(b) There are integers $l_1,\ldots,l_k \le l-1$ and trees $t_1,\ldots,t_k \in T_\Sigma$ such that, $t = \delta(t_1,\ldots,t_k)$ and $\langle p_1,h\rangle \Rightarrow^{l_1}_{M,s} t_1, \ldots, \langle p_k,h\rangle \Rightarrow^{l_k}_{M,s} t_k$.

(c) By Definition 9.4(iii), we obtain that for each $1 \le i \le k$ we have

$$\begin{aligned} \mathit{path\_tree}(\langle p_i,h\rangle) = \langle(p_i,\mathit{toChild}_\omega),u\rangle(\\ \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)). \end{aligned}$$

Hence, we conclude that

$$\begin{aligned} & \mathit{path\_tree}(\langle q,h\rangle) \\ = \quad & \langle(q,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots, \\ & \qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)) && \text{(by (9.1))} \\ \Rightarrow_{M',s} \quad & \delta(\langle(p_1,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots, \\ & \qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)) \\ & \vdots \\ & \langle(p_k,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots, \\ & \qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle))) && \text{(by (a))} \\ = \quad & \delta(\mathit{path\_tree}(\langle p_1,h\rangle),\ldots,\mathit{path\_tree}(\langle p_k,h\rangle)) && \text{(by (c))} \\ \Rightarrow^*_{M',s} \quad & \delta(t_1,\ldots,t_k) && \text{(by IH)} \\ = \quad & t. \end{aligned}$$

Subcase 3.2: $\langle q,h\rangle \Rightarrow_{M,s} \langle p,h\rangle \Rightarrow^{l-1}_{M,s} t$.

(a) There is a rule $\langle q,\sigma,0,j\rangle \to \langle p,\mathit{stay}\rangle$ in R. By $(r_6/0)$, the rule $\langle(q,\mathit{toChild}_\omega),\sigma,j\rangle(y_1,\ldots,y_{m+1}) \to \langle(p,\mathit{toChild}_\omega),\mathit{stay}\rangle(y_1,\ldots,y_{m+1})$ is in R'. Moreover, it follows from (9.1) that

$$\begin{aligned} \mathit{path\_tree}(\langle q,h\rangle) \Rightarrow_{M',s} \langle(p,\mathit{toChild}_\omega),u\rangle(\\ \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)). \end{aligned}$$

(b) By Definition 9.4(iii), we have

$$\begin{aligned} \mathit{path\_tree}(\langle p,h\rangle) = \langle(p,\mathit{toChild}_\omega),u\rangle(\\ \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)). \end{aligned}$$

Hence, we conclude that

$$\begin{array}{lll} & \mathit{path\_tree}(\langle q, h\rangle) & \\ = & \mathit{path\_tree}(\langle q, h\rangle) = \langle (q, \mathit{toChild}_\omega), u\rangle(& \\ & \quad \mathit{path\_tree}(\langle q_0, \mathit{down}_\omega(h)\rangle), \ldots, \mathit{path\_tree}(\langle q_m, \mathit{down}_\omega(h)\rangle)) & \text{(by (9.1))} \\ \Rightarrow_{M',s} & \langle (p, \mathit{toChild}_\omega), u\rangle(& \\ & \quad \mathit{path\_tree}(\langle q_0, \mathit{down}_\omega(h)\rangle), \ldots, \mathit{path\_tree}(\langle q_m, \mathit{down}_\omega(h)\rangle)) & \text{(by (a))} \\ = & \mathit{path\_tree}(\langle p, h\rangle) & \text{(by (b))} \\ \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Subcase 3.3: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \mathit{up}(h)\rangle \Rightarrow^{l-1}_{M,s} t$.

(a) There is a rule $\langle q, \sigma, 0, j\rangle \to \langle p, \mathit{up}\rangle$ in R. By $(r_7/0)$, the rule

$$\begin{aligned} \langle (q, \mathit{toChild}_\omega), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to \langle (p, \mathit{toChild}_j), \mathit{up}\rangle(& \\ \langle (q_0, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \ldots, y_{m+1}), \ldots, & \\ \langle (q_m, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \ldots, y_{m+1})) & \end{aligned}$$

is in R'. Moreover, it follows from (9.1) that

$$\begin{aligned} \mathit{path\_tree}(\langle q, h\rangle) \Rightarrow_{M',s} \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(& \\ \mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle)). & \end{aligned}$$

(b) It follows from Definition 9.4(iii) that

$$\begin{aligned} \mathit{path\_tree}(\langle p, \mathit{up}(h)\rangle) = \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(& \\ \mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle)). & \end{aligned}$$

Hence, we conclude that

$$\begin{array}{lll} & \mathit{path\_tree}(\langle q, h\rangle) & \\ = & \langle (q, \mathit{toChild}_\omega), u\rangle(& \\ & \quad \mathit{path\_tree}(\langle q_0, \mathit{down}_\omega(h)\rangle), \ldots, \mathit{path\_tree}(\langle q_m, \mathit{down}_\omega(h)\rangle)) & \text{(by (9.1))} \\ \Rightarrow_{M',s} & \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(\mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle)) & \text{(by (a))} \\ = & \mathit{path\_tree}(\langle p, \mathit{up}(h)\rangle) & \text{(by (b))} \\ \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Subcase 3.4: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \mathit{down}_i(h)\rangle \Rightarrow^{l-1}_{M,s} t$ for some $1 \leq i \leq \mathit{rank}(\sigma)$ with $i \neq \omega$.

(a) There is a rule $\langle q, \sigma, 0, j\rangle \to \langle p, \mathit{down}_i\rangle$ in R. By $(r_8/0)$, the rule

$$\begin{aligned} & \langle (q, \mathit{toChild}_\omega), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \to \langle (p, \mathit{toParent}), \mathit{down}_i\rangle(\\ & \quad \langle (q_0, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \ldots, y_{m+1}), \ldots, \langle (q_0, \mathit{toChild}_\omega), \mathit{stay}\rangle(y_1, \ldots, y_{m+1})) \end{aligned}$$

is in R'. Moreover, it follows from (9.1) that

$$\begin{aligned} & \mathit{path\_tree}(\langle q, h\rangle) \Rightarrow_{M',s} \\ & \quad \langle (p, \mathit{toParent}), \mathit{down}_i(u)\rangle(\mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle)). \end{aligned}$$

(b) By Definition 9.4(iii), we have

$$\begin{aligned}&\mathit{path\_tree}(\langle p, \mathit{down}_i(h)\rangle) = \\ &\qquad \langle (p, \mathit{toParent}), \mathit{down}_i(u)\rangle(\mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle))\end{aligned}$$

Hence, we conclude that

$$\begin{array}{rll} & \mathit{path\_tree}(\langle q, h\rangle) & \\ = & \langle (q, \mathit{toChild}_\omega), u\rangle(& \\ & \quad \mathit{path\_tree}(\langle q_0, \mathit{down}_\omega(h)\rangle), \ldots, \mathit{path\_tree}(\langle q_m, \mathit{down}_\omega(h)\rangle)) & \text{(by (9.1))} \\ \Rightarrow_{M',s} & \langle (p, \mathit{toChild}_j), \mathit{down}_i(u)\rangle(\mathit{path\_tree}(\langle q_0, h\rangle), \ldots, \mathit{path\_tree}(\langle q_m, h\rangle)) & \text{(by (a))} \\ = & \mathit{path\_tree}(\langle p, \mathit{down}_i(h)\rangle) & \text{(by (b))} \\ \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Subcase 3.5: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \mathit{down}_\omega(h)\rangle \Rightarrow^{l-1}_{M,s} t$.

There is a rule $\langle q, \sigma, 0, j\rangle \rightarrow \langle p, \mathit{down}_\omega\rangle$ in R. By $(r_8/0)$, the rule $\langle (q, \mathit{toChild}_\omega), \sigma, j\rangle(y_1, \ldots, y_{m+1}) \rightarrow y_{\nu+1}$ is in R', such that $p = q_\nu$. Moreover, it follows from (9.1) that $\mathit{path\_tree}(\langle q, h\rangle) \Rightarrow_{M',s} \mathit{path\_tree}(\langle p, \mathit{down}_\omega(h)\rangle)$.

Hence, we conclude that

$$\begin{array}{rll} \mathit{path\_tree}(\langle q, h\rangle) & \Rightarrow_{M',s} & \mathit{path\_tree}(\langle p, \mathit{down}_\omega(h)\rangle) \\ & \Rightarrow^*_{M',s} & t. \qquad\qquad \text{(by IH)} \end{array}$$

Subcase 3.6: $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \mathit{lift}(h)\rangle \Rightarrow^{l-1}_{M,s} t$. Note that $\mathit{lift}(h) = (u, -)$.

(a) The rule $\langle q, \sigma, 0, j\rangle \rightarrow \langle p, \mathit{lift}\rangle$ is in R. Then, by $(r_4/0)$, the rule $\langle (q, \mathit{toChild}_\omega), \sigma, j\rangle(y_1, \ldots, y_m) \rightarrow \langle p, \mathit{stay}\rangle$ is in R'. Moreover, it follows from (9.1) that $\mathit{path\_tree}(\langle q, h\rangle) \Rightarrow_{M',s} \langle p, u\rangle$.

(b) By Definition 9.4(i) we have $\mathit{path\_tree}(\langle p, \mathit{lift}(h)\rangle) = \langle p, u\rangle$.

Hence, we conclude that

$$\begin{array}{rlll} \mathit{path\_tree}(\langle q, h\rangle) & \Rightarrow_{M',s} & \langle p, u\rangle & \text{(by (a))} \\ & = & \mathit{path\_tree}(\langle p, \mathit{lift}(h)\rangle) & \text{(by (b))} \\ & \Rightarrow^*_{M',s} & t. & \text{(by IH)} \end{array}$$

Case 4: $h = (u, u')$ such that the longest common prefix of u and u' is a proper prefix of u. The proof of this case is analogous to Case 3, hence we leave it.

With this we have finished the proof of Statement 1.

Proof of Statement 2. Assume that $\mathit{path\_tree}(\langle q, h\rangle) \Rightarrow^l_{M',s} t$ and that $\mathit{test}(h) = (\sigma, b, j)$. We prove by induction on l.

(i) Let $l = 1$. We observe that, since M is in normal form, (r_1), $(r_5/1)$, and $(r_5/0)$ yield that M' can write at most one output symbol in one computation step. Consequently, $t = \delta \in \Delta^{(0)}$. Now we distinguish the following cases, according to the form of h.

Case 1: $h = (u, -)$. Then $b = \varepsilon$ and, by Definition 9.4(i), $\mathit{path\_tree}(\langle q, h\rangle) = \langle q, u\rangle$ holds. Hence, the rule $\langle q, \sigma, j\rangle \to \delta$ is in R'. Since this rule is obtained in (r_1), the output rule $\langle q, \sigma, \varepsilon, j\rangle \to \delta$ is in R. This concludes that $\langle q, h\rangle \Rightarrow_{M,s} \delta$.

Case 2: $h = (u, u)$. Then $b = 1$ and, by Definition 9.4(ii), $\mathit{path\_tree}(\langle q, h\rangle) = \langle (q, \mathit{here}), u\rangle$ holds. Hence, the rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \delta$ is in R'. This rule is obtained in $(r_5/1)$, hence the output rule $\langle q, \sigma, 1, j\rangle \to \delta$ is in R. Thus we have $\langle q, h\rangle \Rightarrow_{M,s} \delta$.

Case 3: $h = (u, u')$ with $u \neq u'$. Then $b = 0$ and, by Definition 9.4(iii)-(iv), the root of $\mathit{path\_tree}(\langle q, h\rangle)$ is $\langle (q, d), u\rangle$ for some direction $d \in \{\mathit{toParent}, \mathit{toChild}_\omega \mid 1 \leq \omega \leq \mathit{maxr}(\Sigma)\}$. Hence, the rule $\langle (q, d), \sigma, j\rangle \to \delta$ is in R'. This rule is obtained in $(r_5/0)$, thus the output rule $\langle q, \sigma, 0, j\rangle \to \delta$ is in R. Hence we have $\langle q, h\rangle \Rightarrow_{M,s} \delta$.

(ii) Assume that $l \geq 1$. Again we abbreviate induction hypothesis by IH and distinguish four cases, according to the shape of h.

Case 1: $h = (u, -)$. Then $b = \varepsilon$ and, by Definition 9.4(i), $\mathit{path\_tree}(\langle q, h\rangle) = \langle q, u\rangle$ holds. We distinguish three subcases, according to the type of the rule applied in the first step of the computation of t.

Subcase 1.1: $\langle q, u\rangle \Rightarrow_{M',s} \delta(\langle p_1, u\rangle, \ldots, \langle p_k, u\rangle) \Rightarrow^{l-1}_{M',s} t$. We observe the following.

(a) The rule $\langle q, \sigma, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \ldots, \langle p_k, \mathit{stay}\rangle)$ is in R'. This rule is obtained in (r_1). Hence, necessarily, the rule $\langle q, \sigma, \varepsilon, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \ldots, \langle p_k, \mathit{stay}\rangle)$ is in R, which concludes $\langle q, h\rangle \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle)$.

(b) There are integers $l_1, \ldots, l_k \leq l-1$ and trees $t_1, \ldots, t_k \in T_\Sigma$ such that, $t = \delta(t_1, \ldots, t_k)$ and $\langle p_1, u\rangle \Rightarrow^{l_1}_{M',s} t_1, \ldots, \langle p_k, u\rangle \Rightarrow^{l_k}_{M',s} t_k$.

(c) By Definition 9.4(i) $\mathit{path\_tree}(\langle p_1, h\rangle) = \langle p_1, u\rangle, \ldots, \mathit{path\_tree}(\langle p_k, h\rangle) = \langle p_k, u\rangle$.

(d) By the induction hypothesis

$$\langle p_1, h\rangle \Rightarrow^*_{M,s} t_1, \ldots, \langle p_k, h\rangle \Rightarrow^*_{M,s} t_k.$$

Hence, we conclude that

$$\begin{array}{rcll} \langle q, h\rangle & \Rightarrow_{M,s} & \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle) & \text{(by (a))} \\ & \Rightarrow^*_{M,s} & \delta(t_1, \ldots, t_k) & \text{(by (d))} \\ & = & t. & \end{array}$$

Subcase 1.2: $\langle q, u\rangle \Rightarrow_{M',s} \langle p, \varphi(u)\rangle \Rightarrow^{l-1}_{M,s} t$ for some $\varphi \in \{\mathit{stay}, \mathit{up}, \mathit{down}_i \mid 1 \leq i \leq \mathit{rank}(\sigma)\}$. We observe the following.

(a) There is a rule $\langle q, \sigma, j\rangle \to \langle p, \varphi\rangle$ in R'. This rule is obtained in (r_2), thus the rule $\langle q, \sigma, \varepsilon, j\rangle \to \langle p, \varphi\rangle$ is in R. This concludes $\langle q, h\rangle \Rightarrow_{M,s} \langle p, \varphi(h))\rangle$.

(b) It follows from Definition 9.4(i) that $\mathit{path\_tree}(\langle p, \varphi(h)\rangle) = \langle p, \varphi(u)\rangle$ (recall that $h = (u, -)$).

Hence, we obtain

$$\begin{array}{rcll} \langle q, h\rangle & \Rightarrow_{M,s} & \langle p, \varphi(h))\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} & t. & \text{(by IH)} \end{array}$$

Subcase 1.3: $\langle q, u\rangle \Rightarrow_{M',s} \langle (p, \mathit{here}), u\rangle \Rightarrow^{l-1}_{M',s} t$.

(a) The rule $\langle q, \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$ is in R'. This rule is obtained in (r_3). Hence, necessarily, the rule $\langle q, \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$ is in R', which concludes $\langle q, h\rangle \Rightarrow_{M',s} \langle p, \mathit{drop}(h)\rangle$.

(b) By Definition 9.4(ii), $\mathit{path\_tree}(\langle p, \mathit{drop}(h)\rangle) = \langle (p, \mathit{here}), u\rangle$. Note that $\mathit{drop}(h) = (u, u)$.

Hence, we conclude that

$$\begin{array}{lll} \langle q, h\rangle & \Rightarrow_{M,s} \langle p, \mathit{drop}(h)\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Case 2: $h = (u, u)$. Then necessarily $b = 1$ and $\mathit{path\_tree}(\langle q, h\rangle) = \langle (q, \mathit{here}), u\rangle$. We distinguish five subcases, according to the type of the rule applied in the first step of the computation of t.

Subcase 2.1: $\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \delta(\langle (p_1, \mathit{here}), u\rangle, \ldots, \langle (p_k, \mathit{here}), u\rangle) \Rightarrow^{l-1}_{M',s} t$.

(a) The rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \delta(\langle (p_1, \mathit{here}), \mathit{stay}\rangle, \ldots, \langle (p_k, \mathit{here}), \mathit{stay}\rangle)$ is in R' due to $(r_5/1)$. Hence, the rule $\langle q, \sigma, 1, j\rangle \to \delta(\langle p_1, \mathit{stay}\rangle, \ldots, \langle p_k, \mathit{stay}\rangle)$ is in R necessarily, which concludes $\langle q, h\rangle \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle)$.

(b) There are integers $l_1, \ldots, l_k \leq l - 1$ and trees $t_1, \ldots, t_k \in T_\Sigma$ such that, $t = \delta(t_1, \ldots, t_k)$ and $\langle (p_1, \mathit{here}), u\rangle \Rightarrow^{l_1}_{M',s} t_1, \ldots, \langle (p_k, \mathit{here}), u\rangle \Rightarrow^{l_k}_{M',s} t_k$.

(c) By Definition 9.4(ii), $\mathit{path\_tree}(\langle p_1, h\rangle) = \langle (p_1, \mathit{here}), u\rangle, \ldots, \mathit{path\_tree}(\langle p_k, h\rangle) = \langle (p_k, \mathit{here}), u\rangle$.

Hence, we conclude that

$$\begin{array}{lll} \langle q, h\rangle & \Rightarrow_{M,s} \delta(\langle p_1, h\rangle, \ldots, \langle p_k, h\rangle) & \text{(by (a))} \\ & \Rightarrow^*_{M,s} \delta(t_1, \ldots, t_k) & \text{(by IH)} \\ & = t. & \end{array}$$

Subcase 2.2: $\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \langle (p, \mathit{here}), u\rangle \Rightarrow^{l-1}_{M',s} t$. We observe the following.

(a) There is a rule $\langle (q, \mathit{here}), \sigma, j\rangle \to \langle (p, \mathit{here}), \mathit{stay}\rangle$ in R'. This rule is obtained in $(r_6/1)$, hence the rule $\langle q, \sigma, 1, j\rangle \to \langle p, \mathit{stay}\rangle$ is in R. Thus we have $\langle q, h\rangle \Rightarrow_{M,s} \langle p, h\rangle$.

(b) It follows from Definition 9.4(ii) that $\mathit{path\_tree}(\langle p, h\rangle) = \langle (p, \mathit{here}), u\rangle$.

Hence, we conclude that

$$\begin{array}{lll} \langle q, h\rangle & \Rightarrow_{M,s} \langle p, h\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Subcase 2.3:

$$\langle (q, \mathit{here}), u\rangle \Rightarrow_{M',s} \langle (p, \mathit{toChild}_j), \mathit{up}(u)\rangle(\langle (q_0, \mathit{here}), u\rangle, \ldots, \langle (q_m, \mathit{here}), u\rangle) \Rightarrow^{l-1}_{M',s} t.$$

(a) There is a rule

$$\langle (q, \mathit{here}), \sigma, j\rangle \to \langle (p, \mathit{toChild}_j), \mathit{up}\rangle(\langle (q_0, \mathit{here}), \mathit{stay}\rangle, \ldots, \langle (q_m, \mathit{here}), \mathit{stay}\rangle)$$

obtained in $(r_7/1)$ in R'. Hence the rule $\langle q,\sigma,1,j\rangle \to \langle p, up\rangle$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, up(h)\rangle$.

(b) It follows from Definition 9.4(iii) that

$$path\_tree(\langle p,h\rangle) = \langle (p, toChild_j), up(u)\rangle(\langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle).$$

Hence, we conclude that

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} \langle p, up(h)\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Subcase 2.4:

$$\langle (q, here), u\rangle \Rightarrow_{M',s} \\ \langle (p, toParent), down_i(u)\rangle(\langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle) \Rightarrow^*_{M',s} t.$$

(a) There is a rule

$$\langle (q, here), \sigma, j\rangle \to \langle (p, toParent), down_i\rangle(\langle (q_0, here), stay\rangle, \ldots, \langle (q_m, here), stay\rangle)$$

in R' due to $(r_8/1)$. Thus the rule $\langle q,\sigma,1,j\rangle \to \langle p, down_i\rangle$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, down_i(h)\rangle$.

(b) It follows from Definition 9.4(iii) that

$$path\_tree(\langle p,h\rangle) = \langle (p, toParent), down_i(u)\rangle(\langle (q_0, here), u\rangle, \ldots, \langle (q_m, here), u\rangle).$$

Hence, we obtain

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} \langle p, down_i(h)\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Subcase 2.5: $\langle (q, here), u\rangle \Rightarrow_{M',s} \langle p, u\rangle \Rightarrow^{l-1}_{M',s} t$. We observe the following.

(a) The rule $\langle (q, here), \sigma, j\rangle \to \langle p, stay\rangle$ is in R'. This rule is obtained in $(r_4/1)$, Hence, the rule $\langle q,\sigma,j\rangle \to \langle p, lift\rangle$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, lift(h)\rangle$.

(b) By Definition 9.4(i), $path\_tree(\langle p, lift(h)\rangle) = \langle p,u\rangle$.

Hence, we conclude that

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} \langle p, lift(h)\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Case 3: $h = (u,u')$ such that $u' = u\omega u''$ for some $1 \le \omega \le maxr(\Sigma)$ and $u'' \in \mathbb{N}^*$. Then necessarily $b = 0$ and

$$path\_tree(\langle q,h\rangle) = \langle (q, toChild_\omega), u\rangle(\\ path\_tree(\langle q_0, down_\omega(h)\rangle), \ldots, path\_tree(\langle q_m, down_\omega(h)\rangle)). \quad (9.2)$$

We distinguish six subcases, according to the type of the rule applied in the first step of the computation of t.

Subcase 3.1:

$$\begin{aligned} \mathit{path\_tree}(\langle q,h\rangle) &\Rightarrow_{M',s} \delta(\langle(p_1,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\\ &\qquad\qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle))\\ &\qquad\vdots\\ &\qquad \langle(p_k,\mathit{toChild}_\omega),u\rangle(\mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\\ &\qquad\qquad \mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)))\\ &\Rightarrow^{l-1}_{M',s} t. \end{aligned}$$

(a) The rule

$$\begin{aligned} &\langle(q,\mathit{toChild}_\omega),\sigma,j\rangle(y_1,\ldots,y_{m+1})\to\\ &\qquad \delta(\langle(p_1,\mathit{toChild}_\omega),\mathit{stay}\rangle(y_1,\ldots,y_{m+1}),\ldots,\\ &\qquad\qquad \langle(p_k,\mathit{toChild}_\omega),\mathit{stay}\rangle(y_1,\ldots,y_{m+1})) \end{aligned}$$

is in R'. This rule is obtained in $(r_5/0)$. Hence necessarily, the rule $\langle q,\sigma,0,j\rangle \to \delta(\langle p_1,\mathit{stay}\rangle,\ldots,\langle p_k,\mathit{stay}\rangle)$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \delta(\langle p_1,h\rangle,\ldots,\langle p_k,h\rangle)$.

(b) By Definition 9.4(iii), we obtain that for each $1\le i\le k$ we have

$$\begin{aligned} &\mathit{path\_tree}(\langle p_i,h\rangle) = \langle(p_i,\mathit{toChild}_\omega),u\rangle(\\ &\qquad \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)). \end{aligned}$$

(c) There are integers $l_1,\ldots,l_k \le l-1$ and trees $t_1,\ldots,t_k \in T_\Sigma$ such that, $t = \delta(t_1,\ldots,t_k)$ and $\mathit{path\_tree}(\langle p_1,h\rangle) \Rightarrow^{l_1}_{M',s} t_1,\ldots,\mathit{path\_tree}(\langle p_k,h\rangle) \Rightarrow^{l_k}_{M,s} t_k$.

Hence, we conclude that

$$\begin{aligned} \langle q,h\rangle &\Rightarrow_{M,s} \delta(\langle p_1,h\rangle,\ldots,\langle p_k,h\rangle) && \text{(by (a))}\\ &\Rightarrow^*_{M,s} \delta(t_1,\ldots,t_k) && \text{(by IH)}\\ &= t. \end{aligned}$$

Subcase 3.2:

$$\begin{aligned} &\mathit{path\_tree}(\langle q,h\rangle) \Rightarrow_{M',s} \langle(p,\mathit{toChild}_\omega),u\rangle(\\ &\qquad \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)) \Rightarrow^{l-1}_{M',s} t. \end{aligned}$$

(a) The rule $\langle(q,\mathit{toChild}_\omega),\sigma,j\rangle(y_1,\ldots,y_{m+1}) \to \langle(p,\mathit{toChild}_\omega),\mathit{stay}\rangle(y_1,\ldots,y_{m+1})$ is in R'. This rule is obtained in $(r_6/0)$. Hence the rule $\langle q,\sigma,0,j\rangle \to \langle p,\mathit{stay}\rangle$ in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p,h\rangle$.

(b) By Definition 9.4(iii)

$$\begin{aligned} &\mathit{path\_tree}(\langle p,h\rangle) = \langle(p,\mathit{toChild}_\omega),u\rangle(\\ &\qquad \mathit{path\_tree}(\langle q_0,\mathit{down}_\omega(h)\rangle),\ldots,\mathit{path\_tree}(\langle q_m,\mathit{down}_\omega(h)\rangle)). \end{aligned}$$

Hence, we conclude that

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} \langle p,h\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Subcase 3.3:

$$path\_tree(\langle q,h\rangle) \Rightarrow_{M',s} \langle (p, toChild_j), up(u)\rangle(path\_tree(\langle q_0,h\rangle),\dots,path\_tree(\langle q_m,h\rangle)) \Rightarrow^{l-1}_{M',s} t.$$

(a) The rule

$$\begin{aligned} \langle (q, toChild_\omega), \sigma, j\rangle(y_1,\dots,y_{m+1}) \to \langle (p, toChild_j), up\rangle(& \\ \langle (q_0, toChild_\omega), stay\rangle(y_1,\dots,y_{m+1}),\dots, & \\ \langle (q_m, toChild_\omega), stay\rangle(y_1,\dots,y_{m+1})) & \end{aligned}$$

is in R'. This rule is obtained in $(r_7/0)$. Hence necessarily, the rule $\langle q,\sigma,0,j\rangle \to \langle p, up\rangle$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, up(h)\rangle$.

(b) It follows from Definition 9.4(iii) that

$$path\_tree(\langle p, up(h)\rangle) = \langle (p, toChild_j), up(u)\rangle(path\_tree(\langle q_0,h\rangle),\dots,path\_tree(\langle q_m,h\rangle)).$$

Hence, we obtain

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} \langle p,up(h)\rangle & \text{(by (a))} \\ & \Rightarrow^*_{M,s} t. & \text{(by IH)} \end{array}$$

Subcase 3.4:

$$path\_tree(\langle q,h\rangle) \Rightarrow_{M',s} \langle (p, toParent), down_i(u)\rangle(path\_tree(\langle q_0,h\rangle),\dots,path\_tree(\langle q_m,h\rangle)) \Rightarrow^{l-1}_{M',s} t.$$

for some $1 \le i \le rank(\sigma)$, $i \ne \omega$. We observe the following.

(a) The rule

$$\begin{aligned} \langle (q, toChild_\omega), \sigma, j\rangle(y_1,\dots,y_{m+1}) \to \langle (p, toParent), down_i\rangle(& \\ \langle (q_0, toChild_\omega), stay\rangle(y_1,\dots,y_{m+1}),\dots,\langle (q_0, toChild_\omega), stay\rangle(y_1,\dots,y_{m+1})) & \end{aligned}$$

is in R'. This rule is obtained in $(r_8/0)$. Hence the rule $\langle q,\sigma,0,j\rangle \to \langle p, down_i\rangle$ is in R, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, down_i(h)\rangle$.

(b) By Definition 9.4(iii)

$$path\_tree(\langle p, down_i(h)\rangle) = \langle (p, toParent), down_i(u)\rangle(path\_tree(\langle q_0,h\rangle),\dots,path\_tree(\langle q_m,h\rangle))$$

Thus we get

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} & \langle p, down_i(h)\rangle \quad \text{(by (a))} \\ & \Rightarrow^*_{M,s} & t. \quad \text{(by IH)} \end{array}$$

Subcase 3.5: $path\_tree(\langle q,h\rangle) \Rightarrow_{M',s} path\_tree(\langle p, down_\omega(h)\rangle) \Rightarrow^{l-1}_{M',s} t$.

The rule $\langle (q, toChild_\omega), \sigma, j\rangle(y_1,\ldots,y_{m+1}) \to y_{\nu+1}$ is in R'. This rule is obtained in $(r_8/0)$. Hence necessarily, the rule $\langle q,\sigma,0,j\rangle \to \langle p, down_\omega\rangle$ is in R such that $p = q_\nu$, which concludes $\langle q,h\rangle \Rightarrow_{M,s} \langle p, down_\omega(h)\rangle$.

Hence, we conclude that

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} & \langle p, down_\omega(h)\rangle \\ & \Rightarrow^*_{M,s} & t. \quad \text{(by IH)} \end{array}$$

Subcase 3.6: $path\_tree(\langle q,h\rangle) \Rightarrow_{M',s} \langle p,u\rangle \Rightarrow^{l-1}_{M',s} t$. We observe the following.

(a) The rule $\langle q,\sigma,0,j\rangle \to \langle p, lift\rangle$ is in R. Then it follows from $(r_4/0)$, that the rule $\langle (q, toChild_\omega), \sigma, j\rangle(y_1,\ldots,y_m) \to \langle p, stay\rangle$ is in R'. It follows from (9.1) that $\langle q,h\rangle \Rightarrow_{M,s} \langle p, lift(h)\rangle$. Note that $lift(h) = (u,-)$.

(b) By Definition 9.4(i), $path\_tree(\langle p, lift(h)\rangle) = \langle p,u\rangle$.

Hence, we conclude that

$$\begin{array}{lll} \langle q,h\rangle & \Rightarrow_{M,s} & \langle p, lift(h)\rangle \quad \text{(by (a))} \\ & \Rightarrow^*_{M,s} & t. \quad \text{(by IH)} \end{array}$$

Case 4: $h = (u,u')$ such that the longest common prefix of u and u' is a proper prefix of u. The proof of this case is analogous to Case 3, hence we leave it.

With this, we have finished the proof of Statement 2 and also of Lemma 9.5. ◇

We can straightforwardly extend Lemma 9.5 for n-ptts and $(n-1)$-pmtts, where $n \geq 1$, such that each n-pmtt M can be simulated by an $n-1$-pmtt M'.

Theorem 9.6 For each $n \geq 1$, *n-PTT* $\subseteq$ *$(n-1)$-PMTT* and *n-dPTT* $\subseteq$ *$(n-1)$-dPMTT*.

Proof. Let M be an n-ptt, construct the $(n-1)$-pmtt M' in the following way.

- The use of pebbles $1,\ldots,n-1$ of M is simulated in the trivial way, i.e., step-by-step, with the $n-1$ pebbles of M'.
- The use of the last pebble of M is simulated by path-tree-like macro calls of M' as in case $n = 1$.

The construction preserves determinism obviously. ◇

Note that the proof of Theorem 9.6 is based on Lemma 9.5 and this latter, by Observation 9.3(b), works for weak pebble handling. Moreover, it is straightforward that the above extension of Lemma 9.5 for n-ptts and $(n-1)$-pmtts preserves weak pebble handling. Hence we have $n\text{-}PTT_w \subseteq (n-1)\text{-}PMTT_w$ and $n\text{-}dPTT_w \subseteq (n-1)\text{-}dPMTT_w$, where the subscript w denotes the weak pebble versions of the corresponding tree transformation classes. This answers the question raised in the Conclusion of [EM03].

10 Further results

This section consists of three subsections. In Subsection 10.1 we show some consequences and give some applications of the results obtained in Sections 5, 7 and 9. In particular, we show that the inverses of compositions of pebble macro tree transformations effectively preserve regularity and that the domains of (compositions of) pebble macro tree transformations are effectively regular. Then we apply these results to show that the type checking and the (more general) almost always type checking problems for pebble macro tree transformations are decidable.

In Subsection 10.2 we consider the decision problems for the concepts of circularity that we introduced in Section 4. We show that each of them is decidable.

In Subsection 10.3 we introduce the concept of a pebble alternating tree-walking automaton and show that the tree languages recognized by deterministic non-looping pebble alternating tree-walking automata form a strict hierarchy with respect to the number of pebbles. An immediate consequence of this result is that the domains of deterministic and not strongly circular n-pebble tree transformations form a proper hierarchy with respect to n.

The results of Subsections 10.1, 10.2, and 10.3 can be found in Section 6 of [FM09], Section 4.2 of [FM08], and [Muz08], respectively.

10.1 Type checking of pmtts

By Theorems 7.9 and 5.3 we obtain the following decomposition.

Corollary 10.1 For each $n \geq 0$, we have $n\text{-}PMTT \subseteq n\text{-}PTT \circ 0\text{-}PTT$. ◇

Next we recall an inclusion result, which is proved in Lemma 34 of [EM03].

Proposition 10.2 $0\text{-}PTT \subseteq sMTT$. ◇

Then we can show the following inclusions.

Theorem 10.3 For each $n \geq 0$,

$$\begin{array}{rlcl}
(1) & n\text{-}PMTT & \subseteq & 0\text{-}PTT \circ YIELD^{n+1}, \\
(2) & n\text{-}PMTT & \subseteq & 0\text{-}PTT^{n+2}, \\
(3) & n\text{-}PMTT & \subseteq & sMTT^{n+2}.
\end{array}$$

and for each $n \geq 1$,

$$\begin{array}{rlcl}
(4) & n\text{-}PTT & \subseteq & 0\text{-}PTT \circ YIELD^{n}, \\
(5) & n\text{-}PTT & \subseteq & 0\text{-}PTT^{n+1}, \\
(6) & n\text{-}PTT & \subseteq & sMTT^{n+1}.
\end{array}$$

Proof. We prove (1) by induction on n. The case $n = 0$ is verified by Theorem 7.10. If $n > 0$, then we have

$$\begin{array}{lll} n\text{-}PMTT & \subseteq & n\text{-}PTT \circ YIELD \qquad \text{(by Corollary 7.9)} \\ & \subseteq & (n-1)\text{-}PMTT \circ YIELD \qquad \text{(by Theorem 9.6)} \\ & \subseteq & 0\text{-}PTT \circ YIELD^n \circ YIELD \qquad \text{(by the induction hypothesis)} \\ & = & 0\text{-}PTT \circ YIELD^{n+1}. \end{array}$$

Then, (2) follows from (1) and Theorem 5.3, while (3) follows from (2) and Proposition 10.2.

We prove (4) as follows

$$\begin{array}{lll} n\text{-}PTT & \subseteq & (n-1)\text{-}PMTT \qquad \text{(by Theorem 9.6)} \\ & \subseteq & 0\text{-}PTT \circ YIELD^n. \qquad \text{(by (1))} \end{array}$$

and leave the proof of (5) and (6). ◇

Note that (5) was shown in Theorem 10 of [EM03] for weak pebble handling. Above, we gave not only an alternative proof but generalized their result. The mapping EncPeb appearing in the proof of Theorems 10 of [EM03] is strongly based on the weak pebble handling, hence we think that proof cannot be generalized for the strong pebble case. We also note that (6) and (3) were also concluded in Theorem 35 and in Section 8 of [EM03], respectively, for weak pebble handling.

In the next result we will need the particular tree transformation called monadic insertion, cf. Example 6 of [EM03]. Assume that $\Delta = \Sigma \cup \{\overline{\sigma}^{(1)} \mid \sigma \in \Sigma\}$. A *monadic insertion (or regular insertion)* is a tree transformation $mon_\Sigma \subseteq T_\Sigma \times T_\Delta$ that consists of all pairs (s, t) such that $s \in T_\Sigma$ and t is obtained from s by inserting an arbitrary number of unary symbols $\overline{\sigma}$ above each σ-labeled node in s. We denote the class of all monadic insertions mon_Σ by MON.

Theorem 10.4 The inverses of compositions of pebble macro tree transformations effectively preserve regularity.

Proof. It is enough to prove the statement for one pebble macro tree transformation, hence, by (3) of Theorem 10.3, it is enough to prove that the inverse of an arbitrary stay-macro tree transformation effectively preserves regularity. This can be seen as follows (see [EM03, MBPS05a]). By Lemma 27 of [EM03], $sMTT \subseteq MON \circ MTT$, where MON is the class of monadic insertions introduced in Section 2.2. The inverse of a monadic insertion effectively preserves regularity because this inverse just removes barred symbols from trees, therefore it can easily be realized by a linear top-down tree transducer. Moreover, linear top-down tree transformations preserve regularity of tree languages [Tha69, Eng75]. By Theorem 7.4 of [EV85], the inverse of each macro tree transformation also effectively preserves regularity. We conclude that the inverse of each stay-macro tree transformation effectively preserves regularity, which finishes the proof. ◇

Corollary 10.5 The domains of (compositions of) pebble macro tree transformations are effectively regular.

Proof. Let $\tau \subseteq T_\Sigma \times T_\Delta$ be a composition of pebble macro tree transformations. Since $dom(\tau) = \tau^{-1}(T_\Delta)$ and T_Δ is a regular tree language, by Theorem 10.4 we obtain that $dom(\tau)$ is effectively regular. $\diamond$

As another application, we will show that the type checking and the (more general) almost always type checking problems for pebble macro tree transformations are decidable. By the *type checking problem of a tree transformation class $\mathcal{C}$* we mean the following decision problem.

Input: A tree transformation $\tau \subseteq T_\Sigma \times T_\Delta$ of $\mathcal{C}$ and regular tree languages $L_{in} \subseteq T_\Sigma$, $L_{out} \subseteq T_\Delta$.

Output: $\begin{cases} \text{“yes”} & \text{if } \tau(L_{in}) \subseteq L_{out} \text{ (i.e., } \tau(L_{in}) - L_{out} = \emptyset) \\ \text{“no”} & \text{otherwise.} \end{cases}$

Moreover, the *almost always type checking problem* (introduced in [EM03]) is specified as follows.

Input: A tree transformation $\tau \subseteq T_\Sigma \times T_\Delta$ of $\mathcal{C}$ and regular tree languages $L_{in} \subseteq T_\Sigma$, $L_{out} \subseteq T_\Delta$.

Output: $\begin{cases} \text{“yes”} & \text{if } \tau(L_{in}) - L_{out} \text{ is finite} \\ \text{“no”} & \text{otherwise.} \end{cases}$

Theorem 10.6 The type checking and the almost always type checking problems of pebble macro tree transformations are decidable.

Proof. The proof the type checking case simply follows from (3) of Theorem 10.3 and Corollary 44 of [EM03], which states that the type checking problem of compositions of stay-macro tree transformations are decidable.

The proof of the almost always type checking case is similarly obtainable from (2) of Theorem 10.3 and Theorem 45 of [EM03]. $\diamond$

10.2 Deciding circularity problems

The circularity problem of attribute grammars and attributed tree transducers is decidable, see the decision algorithm in [Knu68, Knu71] and its adaptation for attributed tree transducers in [FV98], respectively. Here we consider the following decision problems.

Weak circularity problem (wc-problem): Given a pebble (macro) tree transducer M, is it weakly circular, or not? The *circularity problem (c-problem)* and the *strong circularity problem (sc-problem)* are defined analogously.

Lemma 10.7 The sc-problem for deterministic pmtts is decidable.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a deterministic n-pmtt.

Assume first that M is total. Now, since M is deterministic, M is not strongly circular if and only if $dom(\tau_M) = T_\Sigma$. By Corollary 10.5, the tree language $dom(\tau_M)$ is effectively regular while T_Σ is obviously regular. Moreover, as a corollary of Theorem 10.3 of

Section 10 of Chapter II of [GS84], the equivalence problem for regular tree languages is decidable. Hence the sc-problem for total and deterministic pmtts is decidable.

If M is not total, then we construct a total and deterministic n-pmtt $M' = (Q, \Sigma, \Delta', q_0, R')$, such that $\Delta' = \Delta \cup \{\gamma_0^{(0)}, \gamma_1^{(1)}, \ldots, \gamma_d^{(d)}\}$, where $d = \mathit{maxr}(Q)$, and $\gamma_0, \gamma_1, \ldots, \gamma_d$ are new symbols of ranks $0, \ldots, d$, respectively, R' is the smallest set of rules such that each rule of R is in R', and for every $m \geq 0$, $q \in Q^{(m)}$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, \ldots, \mathit{maxr}(\Sigma)\}$, if $\mathit{rhs}_M(q, \sigma, b, j) = \emptyset$, then the rule $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \gamma_m(y_1, \ldots, y_m)$ is in R'. It is easy to see that M' is strongly circular iff M is strongly circular. This proves our lemma. $\diamond$

In the following theorem we reduce the sc-problem for nondeterministic pmtts to the sc-problem of deterministic pmtts.

Theorem 10.8 The sc-problem for pmtts is decidable.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a nondeterministic n-pmtt. We construct a deterministic n-pmtt M', such that M is strongly circular iff M' is. For this, let $d = \max\{|\mathit{rhs}_M(q, \sigma, b, j)| \mid q \in Q,\ \sigma \in \Sigma, b \in \{0,1\}^{\leq n},\ j \in \{0, \ldots, \mathit{maxr}(\Sigma)\}\}$ be the maximum of the numbers of nondeterministic choices of M, and $M' = (Q, \Sigma, \Delta', q_0, R')$, where

- $\Delta' = \Delta \cup \{\delta_1, \ldots, \delta_d\}$ for new symbols $\delta_1, \ldots, \delta_d$ of rank $1, \ldots, d$, respectively, and
- R' is the smallest set containing the following rules.

  For every $m \geq 0$, $q \in Q^{(m)}$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$ and $j \in \{0, \ldots, \mathit{maxr}(\Sigma)\}$, if $\mathit{rhs}_M(q, \sigma, b, j) = \{\zeta_1, \ldots, \zeta_l\}$, and $l \geq 1$, then the rule $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \delta_l(\zeta_1, \ldots, \zeta_l)$ is in R'.

It is easy to prove that M is strongly circular iff so is M'. $\diamond$

In the next theorem we prove that not only the sc-problem, but also the c-problem is decidable for pmtts. We will show that each configuration $\langle q, h\rangle$ of a pmtt M can be encoded by a particular input tree, and we can simulate the computation of M starting in $\langle q, h\rangle$ by a pmtt M'. Hence, if $\langle q, h\rangle$ is a circular configuration, then (starting in the initial configuration) M' will loop when processing the tree which encodes $\langle q, h\rangle$, i.e., M' is strongly circular. Thus, for deciding the c-problem of M, it is sufficient to decide the sc-problem of M'.

We note that pmtts (even 0-ptts) are able to make a *preorder traversal* of an input tree (see the preorder traversals made by pebble tree-walking automata in [EH99, EHB99]). This essential feature of pmtts is due to the fact that a pmtt can test the child number and the label of the current node.

Theorem 10.9 The c-problem for pmtts is decidable.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an n-pmtt and $\Sigma' = \{\langle \sigma, b\rangle \mid \sigma \in \Sigma,\ b \in \{0,1\}^{\leq n}\} \cup \{(q, \langle \sigma, b\rangle) \mid q \in Q,\ \sigma \in \Sigma,\ b \in \{0,1\}^{\leq n}\}$, where the rank of each symbol $\langle \sigma, b\rangle$ (or

$(q, \langle\sigma, b\rangle))$ of Σ' is the rank of σ. For every tree $s \in T_{\Sigma'}$, let $trunc(s) \in T_\Sigma$ be the tree retrieved from s by deleting the states and bitvectors. We can construct an n-pmtt $M' = (Q', \Sigma', \Delta, q_0', R')$ which works as follows. We describe M' intuitively on an arbitrary input tree $s \in \Sigma'$.

- In the first phase of the computation, M' checks, whether s fulfills the following requirements or not (by making at most $n+2$ preorder traversals). If one of the items a) b) or c) is not true for s, then M' stops without producing an output.
  - a) There must be exactly one node u in s labelled by a symbol of the form $(q, \langle\sigma, b\rangle)$ in s. All the other nodes in s are of the form $\langle\sigma, b\rangle$ (1 preorder traversal).
  - b) There is a number $0 \le l \le n$, such that every node of s is labelled by a symbol $\langle\sigma, b\rangle$ (or $(q, \langle\sigma, b\rangle)$) $\in \Sigma'$ with $|b| = l$ (1 preorder traversal).
  - c) For every $1 \le k \le l$, there must be exactly one node in s, labelled by a symbol $\langle\sigma, b\rangle$ (or $(q, \langle\sigma, b\rangle)$) $\in \Sigma'$, where $b(k) = 1$ (l preorder traversals).
- For every $1 \le k \le l$, let u_k be the (only) node, where the k-th bit of the bit vector component is 1. By making l preorder traversals, M' places pebble 1 at node u_1,..., and pebble l at node u_l, respectively.
- Finally, by a preorder traversal, M' searches the (only) node u labelled by the symbol of the form $(q, \langle\sigma, b\rangle)$, and simulates M starting out from u, respecting only the Σ-components of the nodes (i.e. M' simulates the calculation of M on $trunc(s)$, where pebbles $1, \ldots, l$ are placed at nodes $u_1, \ldots, u_l$, respectively, starting out at node u).

It is straightforward that M' is strongly circular iff M is circular, since if s fulfills properties a), b), and c), then s determines a configuration $\langle q, (u, [u_1; \ldots; u_l])\rangle$ of M and $trunc(s)$. Thus, if M can fall into an infinite cycle from the configuration $\langle q, (u, [u_1; \ldots; u_l])\rangle$, then so can M' starting out from its initial configuration, and vice versa. With this, we have reduced the c-problem of pmtts to the sc-problem of pmtts, which is decidable by Theorem 10.8. $\diamond$

Since ptts are special pmtts, by Theorems 10.8 and 10.9, we obtain the following result.

Corollary 10.10 Both the sc-problem and the c-problem for ptts are decidable.

We have also obtained the following.

Corollary 10.11 The wc-problem for pmtts is decidable.

Proof. Given a pmtt M, construct the ptt M' associated with M (Definition 8.1). By Lemma 8.9, M is weakly circular if and only if M' is circular. Since this latter is decidable by Corollary 10.10, it is also decidable if M is weakly circular. $\diamond$

10.3 The domain hierarchy of not strongly circular dptts

A *hierarchy* is a family $(K_n \mid n \geq 1)$, where K_n is a class such that $K_n \subseteq K_{n+1}$ for every $n \geq 1$. This hierarchy is *strict* if $K_n \subset K_{n+1}$ for every $n \geq 1$

The tree recognizer *n-pebble alternating tree-walking automaton*, defined as follows, models the behaviour of an n-ptt focused on its domain.

Definition 10.12 For $n \geq 0$, an *n-pebble alternating tree-walking automaton* (shortly *n-patwa*) is a system $A = (Q, \Sigma, q_0, q_{yes}, R)$, where

- Q is a finite nonempty set, the *set of states*,
- Σ is a ranked alphabet, the *input alphabet*,
- $q_0 \in Q$ is a distinguished state, the *initial state*,
- $q_{yes} \notin Q$ is a new state, the *accepting state*,
- R is a finite *set of rules*, of the form $\langle q, \sigma, b, j\rangle \to \{\langle p_1, \varphi_1\rangle, \ldots, \langle p_m, \varphi_m\rangle\}$ where $q \in Q$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, $0 \leq j \leq maxr(\Sigma)$, and $p_1, \ldots, p_m, p \in Q$, $\varphi \in I_{\sigma,b,j}$. $\diamond$

By a *pebble alternating tree-walking automaton (patwa)* we mean an n-patwa for some n. A tree $s \in T_\Sigma$ is called an *input tree to A* or just an input tree. In the remainder of this subsection A stands for the n-patwa $A = (Q, \Sigma, q_0, q_{yes}, R)$.

We say that A is *deterministic*, if there is $\underline{\text{at most one}}$ rule of R with left-hand side $\langle q, \sigma, b, j\rangle$ for every $q \in Q$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, 1, \ldots, maxr(\Sigma)\}$. Next we introduce further syntactic restrictions for patwa.

Definition 10.13 A is

- an *alternating tree-walking automaton* (shortly *atwa*), if A is a 0-patwa.
- an *n-pebble tree-walking automaton* (shortly *n-ptwa*)[EH07], if the right-hand side of its each rule has exactly one element. (By a *pebble tree-walking automaton (ptwa)* we mean an n-ptwa for some n.)
- a *tree-walking automaton* (shortly *twa*)[AU71], if A is a 0-ptwa. $\diamond$

In order to define the semantics of patwa, we will use the concept of a pebble configuration, of the execution of an instruction on a pebble configuration, and of a configuration, cf. Subsection 3.3.

Due to alternation, A is capable to make arbitrary number of parallel computations (threads) while processing an input tree s. Hence, the computation relation is defined over the subsets of the set $C_{A,s}$ of configurations over s. Now we turn to introduce this computation relation.

Definition 10.14 Let $s \in T_\Sigma$ be an input tree. The *computation relation of A on s* is the binary relation $\vdash_{A,s} \subseteq \mathcal{P}(C_{A,s}) \times \mathcal{P}(C_{A,s})$ such that, for all configuration sets $H_1, H_2 \in \mathcal{P}(C_{A,s})$ we have $H_1 \vdash_{A,s} H_2$ if and only if there is a rule $\langle q, \sigma, b, j\rangle \to \{\langle p_1, \varphi_1\rangle, \ldots, \langle p_m, \varphi_m\rangle\}$ in R and there is a configuration $\langle q, h\rangle \in H_1$, such that $test(h) = (\sigma, b, j)$ and $H_2 = (H_1 - \{\langle q, h\rangle\}) \cup \{\langle p_1, \varphi_1(h)\rangle, \ldots, \langle p_m, \varphi_m(h)\rangle\}$.
$\diamond$

The n-patwa A works as follows on an input tree s. It starts in the *initial configuration set* $\{\langle q_0, (\varepsilon, [\,])\rangle\}$. Then, applying $\vdash_{A,s}$ step by step, it computes further configuration sets. The goal is that each parallel computation spawned from the initial configuration is accepting, in other words, to terminate in an *accepting configuration set* $H \in \mathcal{P}(C_{A,s})$, which means that the state-component of each configuration in H is q_{yes}. Note that, by definition, there is no computation step from an accepting configuration set.

Let $ACC_{A,s} = \{q_{yes}\} \times PC_{A,s}$ be the largest accepting configuration set. Thus the tree language recognized by A is defined as follows.

Definition 10.15 The *tree language recognized by A* is

$$L(A) = \{s \in T_\Sigma \mid \langle q_0, (\varepsilon, [\,])\rangle \vdash_{A,s}^{*} H, \text{ for some } H \subseteq ACC_{A,s}\}. \qquad \diamond$$

The classes of tree languages computed by n-patwa, atwa, n-ptwa, and twa are denoted by *n-PATWA*, *ATWA*, *n-PTWA*, and *TWA*, respectively. Moreover, $PATWA = \bigcup_{n\geq 0} n\text{-}PATWA$ and $PTWA = \bigcup_{n\geq 0} n\text{-}PTWA$. The deterministic subclasses of the above tree language classes are denoted by prefixing their names with the letter *d*, e.g., *n-dPATWA*, *dATWA*, etc.

It should be clear that with the growing number of pebbles, the recognizing power of patwa and ptwa do not decrease, i.e., $n\text{-}PATWA \subseteq (n+1)\text{-}PATWA$, and $n\text{-}PTWA \subseteq (n+1)\text{-}PTWA$ for every $n \geq 0$. The proof of the following lemma is obvious.

Lemma 10.16 For each $n \geq 0$ we have $n\text{-}PATWA = dom(n\text{-}PTT)$ and $n\text{-}dPATWA = dom(n\text{-}dPTT)$. $\diamond$

Corollary 10.17 *$dom(PTT) = REG$.*

Proof. By Corollary 10.5, we have $dom(PTT) \subseteq REG$. The reverse inclusion follows from Lemma 10.16, and from the fact that each (conventional) top-down tree automaton is a special 0-patwa. $\diamond$

We note that Corollary 10.17 can also be obtained from Theorem 4.7 of [Muz08], stating that $n\text{-}PATWA = REG$, and from Lemma 10.16. However, in this thesis we did not elaborate the details of Theorem 4.7 of [Muz08], even if it is a main result of that paper, because our Corollary 10.5 is a significantly stronger result.

Now we introduce the concept of a looping patwa. Roughly speaking, A is *looping*, if it has an infinite computation on an input tree, which starts in the initial configuration. Note that the looping property for patwa is analogous with the strong circularity of pebble macro tree transducers (defined in Section 4). However, we use the name looping because of historical reasons.

More exactly, a configuration $\langle q, h\rangle \in C_{A,s}$ is a *looping configuration*, if there is a configuration set $H \subseteq C_{A,s}$, such that $\langle q, h\rangle \in H$ and $\langle q, h\rangle \vdash^{+}_{A,s} H$. Moreover, A is *looping*, if there is an input tree $s \in T_\Sigma$, a configuration set $H \subseteq C_{A,s}$ such that

- H contains a looping configuration and
- $\langle q_0, (\varepsilon, [\,])\rangle \vdash^{*}_{A,s} H$.

Otherwise, A is *nonlooping*.

Let us denote the "nonlooping subclasses" of the above tree language classes by $n\text{-}PATWA_{nl}$, $n\text{-}dPATWA_{nl}$, etc. The proof of the following lemma is also obvious.

Lemma 10.18 For each $n \geq 0$ we have $n\text{-}PATWA_{nl} = dom(n\text{-}PTT_{nsc})$ and $n\text{-}dPATWA_{nl} = dom(n\text{-}dPTT_{nsc})$. $\diamond$

Theorem 1 of [BC06] states that $dTWA \subset TWA$, i.e., that deterministic tree-walking automata are less powerful than their nondeterministic counterparts. We notice that the separating tree language treated by [BC06] (which cannot be recognized by a deterministic twa) can also be recognized by a nonlooping twa. Thus, $dTWA_{nl} \subset TWA_{nl}$. Moreover, Proposition 1 of [MSS06] states that $dTWA = dTWA_{nl}$. Hence we obtain the following "nonlooping version" of the above proper inclusion result.

Proposition 10.19 $dTWA \subset TWA_{nl}$. $\diamond$

We will need the following result.

Proposition 10.20 ([MSS06], Theorem 1) $dTWA = co\text{-}dTWA$. $\diamond$

One of the main results of [BSSS06] is Theorem 1.1 which states that ptwa do not recognize all regular tree languages, i.e, that $PTWA \subset \text{REG}$. Using the obvious fact that $PTWA_{nl} \subseteq PTWA$, we obtain the following statement.

Proposition 10.21 $PTWA_{nl} \subset \text{REG}$. $\diamond$

Moreover Theorem 1.2 of [BSSS06] states that the recognizing power of n-ptwa is strictly less than that of $(n{+}1)$-ptwa for each $n \geq 1$, i.e, that $n\text{-}PTWA \subset (n{+}1)\text{-}PTWA$.

We note that this result implies the proper inclusion $TWA \subset \text{REG}$, which was proved in Theorem 2 of [BC05]. We are going to obtain the "nonlooping version" of the hierarchy $n\text{-}PTWA \subset (n+1)\text{-}PTWA$. For this we make the following observations.

- In [MSS06] it was shown that, for each n-ptwa A with weak pebble handling, a nonlooping n-ptwa A' with weak pebble handling can be constructed, such that $L(A) = L(A')$. (Weak pebble handling was discussed in Section 1.)
- It was shown in Lemma 5.1 of [BSSS06] that for each n-ptwa A an n-ptwa A' with weak pebble handling can be constructed, such that $L(A) = L(A')$.

By the above two observations and by the strict hierarchy n-$PTWA \subset (n+1)$-$PTWA$, $n \geq 1$ we conclude the following result.

Proposition 10.22 For each $n \geq 0$, n-$PTWA_{nl} \subset (n+1)$-$PTWA_{nl}$. $\diamond$

Next we prove that the class of complements of tree languages in n-$dPATWA_{nl}$ is the same as n-$PTWA_{nl}$.

Lemma 10.23 For each $n \geq 0$, co-n-$dPATWA_{nl} = n$-$PTWA_{nl}$.

Proof. $\underline{co\text{-}n\text{-}dPATWA_{nl} \subseteq n\text{-}PTWA_{nl}}$: Let $A = (Q, \Sigma, q_0, q_{yes}, R)$ be a deterministic and nonlooping n-patwa. We construct the n-ptwa $A' = (Q, \Sigma, q_0, q_{yes}, R')$ such that R' is the smallest set of rules satisfying the following conditions.

- For each $q \in Q$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, \ldots, maxr(\Sigma)\}$, if there is no rule in R with left-hand side $\langle q, \sigma, b, j\rangle$, then the accepting rule $\langle q, \sigma, b, j\rangle \to \langle q_{yes}, stay\rangle$ is in R'.
- Each pebble tree-walking rule $\langle q, \sigma, b, j\rangle \to \langle p, \varphi\rangle$ of R is also in R'.
- For each alternating rule $\langle q, \sigma, b, j\rangle \to \{\langle p_1, stay\rangle, \langle p_2, stay\rangle\}$, the pebble tree-walking rules $\langle q, \sigma, b, j\rangle \to \langle p_1, stay\rangle$ and $\langle q, \sigma, b, j\rangle \to \langle p_2, stay\rangle$ are in R'.

Since M is nonlooping, it is obvious that also M' is nonlooping. The proof of $L(A') = \overline{L(A)}$ is straightforward, hence we omit it.

$\underline{n\text{-}PTWA_{nl} \subseteq co\text{-}n\text{-}dPATWA_{nl}}$: Let $A = (Q, \Sigma, q_0, q_{yes}, R)$ be a nonlooping ptwa. We construct the deterministic patwa $A' = (Q', \Sigma, q_0, q'_{yes}, R')$ as follows.

- $Q' = Q \cup \{q_{yes}\}$
- R' is the smallest set of rules satisfying the following conditions.
  - For each $q \in Q$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, \ldots, maxr(\Sigma)\}$, if there is no rule in R with left-hand side $\langle q, \sigma, b, j\rangle$, then the accepting rule $\langle q, \sigma, b, j\rangle \to \langle q'_{yes}, stay\rangle$ is in R'.
  - For each $q \in Q$, $\sigma \in \Sigma$, $b \in \{0,1\}^{\leq n}$, and $j \in \{0, \ldots, maxr(\Sigma)\}$, if $\{\langle q_1, \varphi_1\rangle, \ldots, \langle q_m, \varphi_m\rangle\}$ is the set of state-instruction pairs that are the right-hand sides of rules in R with left-hand side $\langle q, \sigma, b, j\rangle$, then the rule $\langle q, \sigma, b, j\rangle \to \{\langle q_1, \varphi_1\rangle, \ldots, \langle q_m, \varphi_m\rangle\}$ is in R'.

Again, it is obvious that A' is deterministic, nonlooping, and that $\overline{L(M)} = L(M')$. $\diamond$

Now we prove the following proper inclusion result.

Theorem 10.24 $dTWA \subset dATWA_{nl}$.

Proof. We prove by contradiction. Let us assume that $dTWA = dATWA_{nl}$ and argue as follows.

a) Obviously, $co\text{-}dTWA = co\text{-}dATWA_{nl}$.

b) By a) and Proposition 10.20, we get $dTWA = co\text{-}dATWA_{nl}$.

c) By b) and $co\text{-}dATWA_{nl} = TWA_{nl}$ (the case $n = 0$ of Lemma 10.23), we obtain $dTWA = TWA_{nl}$, which contradicts Proposition 10.19. $\diamond$

Next we prove that the class of tree languages recognized by deterministic and nonlooping n-patwa form a proper hierarchy with respect to n.

Theorem 10.25 For each $n \geq 0$, $n\text{-}dPATWA_{nl} \subset (n+1)\text{-}dPATWA_{nl}$.

Proof. The inclusion $n\text{-}dPATWA_{nl} \subseteq (n+1)\text{-}dPATWA_{nl}$ is obvious. We prove that the inclusion is proper by contradiction. Let us assume that $n\text{-}dPATWA_{nl} = (n+1)\text{-}dPATWA_{nl}$. Then also $co\text{-}n\text{-}dPATWA_{nl} = co\text{-}(n+1)\text{-}dPATWA_{nl}$ and, by Lemma 10.23, we obtain that $n\text{-}PTWA_{nl} = (n+1)\text{-}PTWA_{nl}$. However, this contradicts Proposition 10.22. $\diamond$

Finally, we prove, that deterministic and nonlooping patwa cannot recognize all regular tree languages.

Theorem 10.26 $dPATWA_{nl} \subset$ REG.

Proof. The inclusion $dPATWA_{nl} \subseteq$ REG comes from Lemma 10.18 and Corollary 10.5. We prove that the inclusion is proper by contradiction. For this, assume that $dPATWA_{nl} =$ REG.

a) Then $co\text{-}dPATWA_{nl} = co$-REG.

b) By a) and Proposition 2.2 we obtain $co\text{-}dPATWA_{nl} =$ REG.

c) By b) and Lemma 10.23 we get $PTWA_{nl} =$ REG, which contradicts Proposition 10.21. $\diamond$

Applying Lemma 10.18 and Theorems 10.24, 10.25, and 10.26 we obtain the following theorem.

Theorem 10.27 (1) $dTWA \subset dom(0\text{-}dPTT_{nsc})$.

(2) For each $n \geq 0$, $dom(n\text{-}dPTT_{nsc}) \subset dom((n+1)\text{-}dPTT_{nsc})$.

(3) $dom(dPTT_{nsc}) \subset$ REG. $\diamond$

In Fig. 6 we visualize the strict hierarchy $(dom(n\text{-}dPTT_{nsc}) \mid n \geq 0)$ and its relation to the classes REG and $dTWA$.

$$\begin{tikzcd}[row sep=small]
\text{Cor. 10.17} & \\
\mathrm{REG} = dom(PTT) \arrow[d, dash, "\text{Th. 10.27(3)}"] & \\
dom(dPTT_{nsc}) \arrow[d, dash] & \\
\vdots \arrow[d, dash] & \left.\vphantom{\begin{matrix}a\\a\\a\\a\\a\end{matrix}}\right\} \text{Th. 10.27(2)} \\
dom(1\text{-}dPTT_{nsc}) \arrow[d, dash] & \\
dom(0\text{-}dPTT_{nsc}) \arrow[d, dash, "\text{Th. 10.27(1)}"] & \\
dTWA &
\end{tikzcd}$$

Figure 6: The hierarchy $(dom(n\text{-}dPTT_{nsc}) \mid n \geq 0)$ and its relation to REG and $dTWA$.

11 Conclusions

We considered pebble macro tree transducers with strong pebble handling. As the first main result, we proved the characterization $n\text{-}PMTT = n\text{-}PTT \circ YIELD$ for each $n \geq 0$ (Theorem 7.10).

We have also investigated decompositions of special pmtts, such that determinism and noncircularity are preserved. In fact, we gave the following decompositions of the tree transformation classes computed by deterministic and by context-linear not weakly circular pmtts: $n\text{-}dPMTT_{nwc} \subseteq n\text{-}dtPTT_{nc} \circ dYIELD$ and $n\text{-}clPMTT_{nwc} \subseteq n\text{-}tPTT_{nc} \circ dYIELD$ (Theorem 8.10). As a corollary, we obtained the following results concerning composition closures: $n\text{-}dPMTT^*_{nwc} = n\text{-}dPTT^*_{nc}$ and $n\text{-}clPMTT^*_{nwc} = n\text{-}PTT^*_{nc}$ (Theorem 8.11). Further corollaries are $0\text{-}dPMTT_{nwc} \subseteq (0\text{-}dPTT_{nc})^2$ and $0\text{-}clPMTT_{nwc} \subseteq (0\text{-}PTT_{nc})^2$ (Theorem 8.12).

Then we proved that each n-ptt can be simulated by an $(n-1)$-pmtt, i.e., that the inclusion $n\text{-}(d)PTT \subseteq (n-1)\text{-}(d)PMTT$ (Theorem 9.6) holds for each $n \geq 1$.

As corollaries, we obtained the inclusions $n\text{-}PMTT \subseteq 0\text{-}PTT \circ YIELD^{n+1}$, $n\text{-}PMTT \subseteq 0\text{-}PTT^{n+2}$, and $n\text{-}PMTT \subseteq sMTT^{n+2}$ for every $n \geq 0$; and $n\text{-}PTT \subseteq 0\text{-}PTT \circ YIELD^n$, $n\text{-}PTT \subseteq 0\text{-}PTT^{n+1}$, and $n\text{-}PTT \subseteq sMTT^{n+1}$ for every $n \geq 0$ (Theorem 10.3).

We have also proved that the inverses of (compositions of) pebble macro tree transformations effectively preserve regularity (Theorem 10.4), which implies that the domains of (compositions of) pebble macro tree transformations are regular languages (Corollary 10.5).

As a contribution to XML theory, we concluded that the type checking problem of pebble macro tree transformations is decidable (Theorem 10.6).

Moreover, we have also obtained from Corollary 10.5 that the three circularity problems (the sc-problem, the c-problem, and the wc-problem) introduced in this thesis are decidable for pmtts. (Theorems 10.8, 10.9, and Corollary 10.11).

We also showed that the domains of deterministic and not strongly circular n-pebble tree transformations form a strict hierarchy with respect to n, see Theorem 10.27 and Fig. 6.

Using alternative proof methods, we more or less have extended the decomposition and type checking results of [EM03] for pmtts with strong pebble handling. In the proof of Theorem 10.3, which is the hearth of the extension, we do not use the tree transformation *EncPeb* of [EM03]. In fact *EncPeb* is applicable only for n-pmtts with weak pebbles. However, it is an interesting open problem whether we can extend *EncPeb* for strong pebbles and use this extension in decomposing n-pmtts (n-ptts) into 0-pmtts (0-ptts), as it was done for the weak pebble case in [EM03].

As we mentioned, the n-ptt M' constructed in Definition 7.1 from an arbitrary n-pmtt M is nondeterministic. However, another construction may exist showing that $n\text{-}dPMTT \subseteq n\text{-}dPTT \circ dYIELD$ holds. If this is the case, then (with similar proofs) we could conclude the deterministic (and maybe also noncircular) versions of the inclusions obtained in Theorem 10.3.

It is still open whether n-pebble macro tree transducers are more powerful than n-pebble macro tree transducers with weak pebble handling (i.e., than n-pebble macro tree transducers of [EM03]). If n-pmtts and n-pmtts with weak pebble handling have the same transformation capacity and hence, for each n-pmtt, there is and equivalent n-pmtt with weak pebbles, then we can apply *EncPeb* to re-obtain our decomposition results for strong pebble cases, such that determinism and noncircularity is preserved.

12 Magyar nyelvű összefoglaló

A disszertációban kavics makró fatranszformátorokat vizsgálunk. Egy *n-kavics makró fatranszformátor* alatt egy olyan M, véges állapotú eszközt értünk, amely egy rangolt ábécé feletti véges fákat transzformál át egy másik rangolt ábécé feletti véges fákká. Az input és output ábécé elemein kívül maguk az állapotok is rangolt szimbólumok. M szabályai $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$ alakúak, ahol q egy állapot, σ egy input szimbólum, b egy legfeljebb n hosszúságú bit vektor, j egy nemnegatív egész, $y_1, \ldots, y_m$ paraméter változók, továbbá ζ egy olyan rangolt fa, melynek csúcsai output szimbólumok, $\langle p, \varphi\rangle$ alakú állapot-utasítás párok és (levélben) $y_1, \ldots, y_m$. Megjegyezzük, hogy $\langle p, \varphi\rangle$ rangja megegyezik a p állapot rangjával, valamint a φ utasítás *stay*, *up* vagy $down_i$ alakú lehet.

M rendelkezik egy mutatóval, amely egy s input fa tetszőleges csúcsára mutathat és annak élein mozoghat, továbbá M veremszerűen elhelyezheti-felszedheti az $1, \ldots, n$ számú kavicsokat s csúcsain. Egy s-en elhelyezett kavics bizonyos információkat nyújthat M-nek, amely módosíthatja a számítást.

M az s input fán vándorolva mondatformák egy sorozatát számítja ki. *Mondatforma* alatt egy olyan ξ fát értünk, amely output szimbólumokból és $\langle p, (u, [u_1; \ldots; u_l])\rangle$ alakú *konfigurációkból* áll valamely $0 \leq l \leq n$-re, ahol p a konfiguráció állapota, u egy s-beli csúcs, amelyre M mutatója mutat, végül $u_1, \ldots, u_l$ szintén s-beli csúcsok, amelyek azt jelölik, hogy az $1, \ldots, l$ számú kavicsok el vannak helyezve s-en, mégpedig rendre az $u_1, \ldots, u_l$ csúcsokon. Egy $\langle p, (u, [u_1; \ldots; u_l])\rangle$ konfiguráció $(u, [u_1; \ldots; u_l])$ részét *kavics konfigurációnak* mondjuk. Megjegyezzük, hogy $\langle p, (u, [u_1; \ldots; u_l])\rangle$ rangja megegyezik p rangjával.

A számítás egy speciális mondatformával, a $\langle q_0, (\varepsilon, [\,])\rangle$ *kezdőkonfigurációval* indul, ahol q_0 jelenti M kezdőállapotát, ε a gyökérre mutató pointert, $[\,]$ pedig kavics pozícióknak az üres listáját.

Ezek után M a következőképpen működik. Tegyük fel, hogy a számítás egy ξ mondatformánál tart. Tekintsük ξ-nek egy olyan v csúcsát, amelynek a címkéje egy $\langle p, (u, [u_1; \ldots; u_l])\rangle$ alakú konfiguráció, és a ξ gyökerétől v-ig tartó út minden v-től különböző csúcsának output szimbólum (tehát nem konfiguráció) a címkéje. Vegyünk továbbá egy olyan $r : \langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$, M-beli szabályt, feltéve ha létezik, amelyre teljesül, hogy

- az s input fa u csúcsa σ-val címkézett,
- a b bit vektor l hosszú és pontosan azokon az i indexeken 1, ahol $u_i = u$, továbbá
- az s fa u csúcsa j-ik fia az apjának ($j = 0$ esetben $u = \varepsilon$ maga a gyökér).

Ekkor M az r szabályt az alábbikaban leírt módon *alkalmazza* a ξ mondatforma v csúcsában a következő mondatforma kiszámítására.

1) Minden ζ-ban előforduló φ utasítás végrehajtódik az $(u, [u_1; \ldots; u_l])$, kavics konfiguráción, amelyből eredményül keletkezett kavics konfigurációt $\varphi((u, [u_1; \ldots; u_l]))$-lel jelöljük. Ha $\varphi = down_i$ ($\varphi = up$), akkor $\varphi((u, [u_1; \ldots; u_l])) = (u', [u_1; \ldots; u_l])$,

amely azt jelöli, hogy a mutató elmozdult az u csúcs i-edik fiára (apjára), vagyis u'-re. Továbbá, ha $\varphi = \textit{drop}$, akkor $\varphi((u,[u_1;\ldots;u_l])) = (u,[u_1;\ldots;u_l;u])$, amely azt jelenti, hogy M az $l+1$-ik kavicsot az u csúcsra helyezte el (feltéve ha $l < n$). Végül, ha $\varphi = \textit{lift}$, akkor $\varphi((u,[u_1;\ldots;u_l])) = (u,[u_1;\ldots;u_{l-1}])$, vagyis M az l-edik kavicsot felszedte az u_l csúcsról (feltéve ha $l > 0$). (Megjegyezzük, hogy kavicsot csak arra a pozícióra helyezhetünk, ahová a mutató mutat, míg kavicsot felemelni tetszőleges pozícióról lehet. Ez utóbbit *erős kavics kezelésnek* hívjuk, szemben a gyenge kavics kezeléssel (lásd lentebb).)

2) Tetszőleges φ, ζ-ban előforduló utasításra a $\varphi((u,[u_1;\ldots;u_l]))$ kavics konfiguráció behelyettesítődik a ζ-beli φ szimbólum minden előfordulásába. Jelöljük ζ'-vel a keletkezett fát.

3) Végül ζ' (amelynek a levelei lehetnek $y_1,\ldots,y_m$, paraméter változók) másodrendű módon behelyettesítődik a ξ mondatforma $\langle p,(u,[u_1;\ldots;u_l])\rangle$ konfigurációval címkézett v csúcsába. A helyettesítés eredményezi a számítás következő mondatformáját.

Ha a keletkezett mondatforma nem tartalmaz konfigurációt, akkor ez az M kavics fatranszformátor s inputjának egy *output fája.* A tézisben τ_M-mel jelöljük az M által kiszámított fatranszformációt, amely az összes ily módon keletkező input-output fákból álló párok halmaza.

M működésére vonatkozóan az alábbi megjegyzéseket tesszük.

- Az eredetileg [EM03]-ban bevezetett kavics makró fatranszformátor ú.n. *gyenge kavics kezeléssel* működik, ami még annyi megszorítást jelent a tézisben szereplő általánosabb fatranszformátorral szemben, hogy kavicsot csak úgy szedhetünk fel az input fáról, ha az adott kavics pozíciója egyben a mutató pozíciója. Tehát csak olyan $(u,[u_1;\ldots;u_l])$ kavics konfiguráción hajthatjuk végre a *lift* utasítást, amelyre $u = u_l$.
- Ha M minden állapota 0 rangú, akkor M egy *n-kavics fatranszformátor.*
- Lehetséges, hogy M számítási sorozata az s input fán soha nem fog output fában terminálni. Az ilyen esetekről a 4. fejezetben részletesen tárgyalunk és bevezetjük a vonatkozó *gyenge cirkularitás*, *cirkularitás* és *erős cirkularitás* fogalmakat.

Az első fő eredményünk egy kavics makró fatranszformátorokra vonatkozó "yield típusú" kompozíció. Nevezetesen, tetszőleges M, n-kavics fatranszformátor és $\textit{yield}_g$ fatranszformáció esetén (ahol g egy 0-rangú szimbólumokból fákba történő leképezés), megkonstruálunk egy olyan M', n-kavics makró fatranszformátort, amelyre teljesül a $\tau_M \circ \textit{yield}_g = \tau_{M'}$ egyenlőség. Ezzel bebizonyítjuk az $n\text{-}PTT \circ \textit{YIELD} \subseteq n\text{-}PMTT$ kompozíciós eredményt, lásd a 6.1 Lemmát, ahol $n\text{-}PTT$ ($n\text{-}PMTT$) jelenti az n-kavics (makró) fatranszformátorok által kiszámított fatranszformációk osztályát, míg $\textit{YIELD}$ jelenti a nemdeterminisztikus yield fatranszformációk osztályát. A kompozíciónkkal kapcsolatban megjegyezzük, hogy ha M és $\textit{yield}_g$ determinisztikusak (totálisak), akkor ugyanúgy M' is determinisztikus (totális) lesz.

Továbbá, a fenti eredmény fordítottjaként, kavics makró fatranszformátorokat dekomponálunk. Nevezetesen, tetszőleges M, n-kavics makró fatranszformátorhoz megkonstruálunk egy olyan M', n-kavics fatranszformátort és egy $yield_g$ fatranszformációt, amelyre $\tau_M = \tau_{M'} \circ yield_g$, lásd a 7.7 Lemmát. Így kapjuk az $n\text{-}PMTT \subseteq n\text{-}PTT \circ YIELD$ dekompozíciós eredményt, lásd a 7.9 Következményt. Továbbá, a 6.1 Lemmából és 7.9 Következményből nyilvánvalóan következik az $n\text{-}PMTT = n\text{-}PTT \circ YIELD$ karakterizáció (7.10 Tétel).

Sajnos a fenti dekompozíciós konstrukció gyengesége, hogy M' minden esetben erősen cirkuláris és nem determinisztikus (akármilyen speciálisnak választjuk meg a kiindulási M-et). Ezt a problémát a 7.3 fejezetben tárgyaljuk részletesebben, ahol olyan, kavics makró fatranszformátorokra vonatkozó dekompozíciós technikát keresünk, amely megőrzi a determinisztikusságot és a nemcirkularitást. Megadunk egy olyan dekompozíciós konstrukciót (8.1 Definíció), amely bizonyos speciális kavics makró fatranszformátorokra alkalmazható és amelyik megőrzi az előbb említett tulajdonságokat. A 8.4 és 8.6 Lemmákban megmutatjuk, hogy ha M determinisztikus (vagy kontext-lineáris), továbbá M' nemcirkuláris, akkor valóban teljesül a $\tau_M = \tau_{M'} \circ yield_g$ egyenlőség.

Ezek után megvizsgáljuk, hogy a 8.1 Definícióban megadott dekompozíciós konstrukcióban mikor teljesül, hogy M' nemcirkuláris. Egy triviális eset az, amikor M egy makró fatranszformátor. Ekkor nyilvánvalóan M' egy felszálló fatranszformátor, lásd [EV85]-öt, amely biztosan nem lehet cirkuláris. Egy kevésbé triviális eredményünk pedig az, hogy ha M nem gyengén cirkuláris, akkor M' nemcirkuláris. Ezzel megkapjuk tézisünk egy következő fő eredményét, a 8.10 Következményt, amely kimondja, hogy tetszőleges determinisztikus (vagy kontext-lineáris) nem gyengén cirkuláris M, n-kavics makró fatranszformátor esetén megkonstruálhatunk olyan M', n-kavics fatranszformátort és $yield_g$ fatransformációt, amelyre $\tau_M = \tau_{M'} \circ yield_g$ teljesül.

A következő fő eredményünkben, a 9.6 Tételben, bebizonyítjuk, hogy tetszőleges M, n-kavics fatranszformátor szimulálható $(n-1)$-kavics makró fatranszformátorral (feltéve, ha $n \geq 1$). Képletben, tetszőleges $n \geq 1$ esetén $n\text{-}PTT \subseteq (n-1)\text{-}PMTT$. A konstrukcióban rejlő ötlet, hogy a legutolsó kavics működését makró hívásokkal tudjuk szimulálni.

A 7.10 és 9.6 Tételek fontos következményekhez vezetnek. Néhányat összegyűjtöttünk a 10.3 Tételben, melyek közül a legérdekesebbek a következő dekompozíciók: $n\text{-}PTT \subseteq 0\text{-}PTT^{n+1}$ és $n\text{-}PTT \subseteq sMTT^{n+1}$ tetszőleges $n \geq 1$-re, továbbá ezek makró verziói, vagyis $n\text{-}PMTT \subseteq 0\text{-}PTT^{n+2}$ és $n\text{-}PMTT \subseteq sMTT^{n+2}$ tetszőleges $n \geq 0$-ra, ahol $sMTT$ alatt az [EM03]-ban bevezetett *stay-makró fatranszformációk* osztályát értjük.

A fenti dekompozíciós eredményekből a következő eredmények adódnak.

1) Az $n\text{-}PMTT \subseteq sMTT^{n+2}$ tartalmazásból és a stay-makró fatranszformációkra vonatkozó [EM03]-beli eredményekből következik, hogy a kavics makró fatranszformációk (kompozíciói) megőrzik a regularitást (10.4 Tétel). Következésképpen a kavics makró fatranszformációk értelmezési tartományai regulárisak (10.5 Következmény). A 10.4 Tételnek további következménye a 10.6 Tétel, mely szerint az XML elméletből ismert típus ellenőrzési és "majdnem mindig" típus ellenőrzési problémák eldönthetőek a kavics makró fatranszformációkra.

2) A kavics makró fatranszformációk értelmezési tartományának reguláris voltát ki-

használva (10.5 Következmény) megadunk olyan algoritmusokat, amelyek a kavics makró fatranszformátorok gyenge cirkularitási, cirkularitási valamint erős cirkularitási tulajdonságait eldönti (rendre a 10.8, 10.9 Tételek és a 10.11 Következmény).

Végezetül megvizsgáljuk a kavics fatranszformációk értelmezési tartományait. Definiáljuk az *n-kavics alternáló fabejáró automata* fogalmát, amely egy n-kavics fatranszformátor input fán való működését modellezi. Az új automata fogalom segítségével a 10.27 Tételben bebizonyítjuk, hogy a determinisztikus nem erősen cirkuláris n-kavics fatranszformációk értelmezési tartományaiból képzett fanyelvosztályok valódi tartalmazási hierarchiát képeznek n értékére vonatkozóan.

A tézis felépítése a következő. A 2. fejezetben definiáljuk a szükséges alapfogalmakat, majd a 3. fejezetben bevezetjük a kavics makró fatranszformátor fogalmát. A 4. fejezetben definiáljuk a három cirkularitási fogalmat és összefüggéseket állapítunk meg közöttük. Az 5. fejezetben általánosítjuk az [EV85]-ben bevezetett yield fatranszformációkat, majd a főbb eredményeinkhez szükséges előkészítő lemmákat igazolunk rájuk vonatkozóan. A 6. fejezetben megmutatjuk, hogy minden n-kavics fatranszformáció és yield fatranszformáció kompozíciója kiszámítható egy n-kavics fatranszformátorral. A 7. fejezetben kimondjuk és bebizonyítjuk az n-kavics makró fatranszformátorokra vonatkozó (általános esetben is működő) dekompozíciós eredményünket, a 8. fejezetben pedig speciális kavics makró fatranszformátorokra vonatkozó dekompozíciós eredményeket bizonyítunk. A 9. fejezetben megmutatjuk, hogy tetszőleges n-kavics fatranszformátor szimulálható egy $(n-1)$-kavics fatranszformátorral (feltéve, ha $n \geq 1$). A 10. fejezetben felsoroljuk a kompozíciós, dekompozíciós és a 9. fejezetben szereplő eredmények fontos következményeit és alkalmazásait. Végezetül, a 11. fejezetben összefoglaljuk eredményeinket és további nyitott problémákat vetünk fel.

Az 5., 6., és 8. fejezetek eredményei [FM08]-ban, a 7. és 9. fejezetek eredményei [FM09]-ben, a 10. fejezet eredményei [Muz08]-ban és [FM08, FM09]-ben kerültek publikálásra.

13 Acknowledgments

I would like to express my gratitude to Professor Zoltán Fülöp who supervised my progress and freely shared his knowledge on the theory of tree automata and tree transducers. Professor Fülöp invested enormous effort in my training and made me enthusiastic about the beauty of making research in theoretical computer science.

I am also grateful to Professor Heiko Vogler for a three-month staying at the Faculty of Informatics of the Technical University of Dresden in 2004. With his support and without other obligations, I had a very productive research period during that time.

I could not have achieved the strict requirement of the PhD School without the knowledge I got from my mathematics teachers from primary school up to university. In particular, I thank my secondary school matematics teacher, Mrs Ferdinánd for her excellent teaching work.

I have an everlasting gratitude towards my family. Their love always provided me with hope and power to do my studies.

Finally, I thank my bride, Emese Orsolya Barcsa, for her love, support and patience while I was working on this thesis.

Loránd Muzamel

14 The author's publications cited in the thesis

[FM08] Z. Fülöp and L. Muzamel. Circularity and Decomposition Results for Pebble Macro Tree Transducers. *Journal of Automata, Languages and Combinatorics*, 13(1):3–44, 2008.

[FM09] Z. Fülöp and L. Muzamel. Pebble Macro Tree Transducers with Strong Pebble Handling. *Fundam. Inf.*, 89(2-3):207–257, 2009.

[Muz08] L. Muzamel. Pebble Alternating Tree-Walking Automata and Their Recognizing Power. *Acta Cybernetica*, 18(3):427–450, 2008.

Index

References

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

[Bak79] B. S. Baker. Composition of top-down and bottom-up tree transductions. *Inform. and Control*, 41:186–213, 1979.

[BC05] M. Bojańczyk and T. Colcombet. Tree-walking automata do not recognize all regular languages. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC '05)*, pages 234–243, New York, NY, USA, 2005. ACM Press.

[BC06] M. Bojańczyk and T. Colcombet. Tree-walking automata cannot be determinized. *Theoretical Computer Science*, 350:164–173, 2006.

[BMN02] G. J. Bex, S. Maneth, and F. Neven. A formal model for an expressive fragment of XSLT. *Information Systems*, 27:21–39, 2002.

[Boo83] R. V. Book. Thue-systems and the Church-Rosser property: replacement systems, specification of formal languages and presentations of monoids. In L. Cummings, editor, *Progress in combinatorics on words*, pages 1–38. Academic Press, New York, 1983.

[BSSS06] M Bojańczyk, M. Samuelides, T. Schwentick, and L. Segoufin. Expressive power of pebble automata. In *ICALP'06: Proceedings of 33rd International Colloquium on Automata, Languages and Programming*, pages 157–168. Springer Berlin / Heidelberg, 2006.

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

[EH99] J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83, London, UK, 1999. Springer-Verlag.

[EH05] J. Engelfriet and Hendrik Jan Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. Technical Report 05-02, Leiden University, The Netherlands, April 2005.

[EH07] Joost Engelfriet and Hendrik Jan Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. *Logical Methods in Computer Science*, 3(2), 2007.

[EHB99] J. Engelfriet, H. J. Hoogeboom, and J.-P. Van Best. Trips on Trees. *Acta Cybernet.*, 14:51–64, 1999.

[EM99] J. Engelfriet and S. Maneth. Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations. *Inform. and Comput.*, 154:34–91, 1999.

[EM03] J. Engelfriet and S. Maneth. A Comparison of Pebble Tree Transducers with Macro Tree Transducers. *Acta Informatica*, 39:613–698, 2003.

[Eng75] J. Engelfriet. Bottom–up and top–down tree transformations — A comparison. *Math. Systems 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*, pages 241–286. New York, Academic Press, 1980.

[Eng81] J. Engelfriet. Tree transducers and syntax-directed semantics. Technical Report Memorandum 363, Technische Hogeschool Twente, March 1981. also in: Proceedings of the Colloquium on Trees in Algebra and Programming (CAAP 1992), Lille, France 1992.

[ES77] J. Engelfriet and E. M. Schmidt. IO and OI, Part I. *J. Comput. System Sci.*, 15(3):328–58, 1977.

[ES78] J. Engelfriet and E.M. Schmidt. IO and OI, Part II. *J. Comput. System Sci.*, 16(1):67–99, 1978.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. System Sci.*, 31:71–146, 1985.

[EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoret. Comput. Sci.*, 42(3):251–368, 1986.

[Fis68] M.J. Fischer. *Grammars with macro–like productions.* PhD thesis, Harvard University, Massachusetts, 1968.

[FM08] Z. Fülöp and L. Muzamel. Circularity and Decomposition Results for Pebble Macro Tree Transducers. *Journal of Automata, Languages and Combinatorics*, 13(1):3–44, 2008.

[FM09] Z. Fülöp and L. Muzamel. Pebble Macro Tree Transducers with Strong Pebble Handling. *Fundam. Inf.*, 89(2-3):207–257, 2009.

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

[FV98] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics — Formal Models Based on Tree Transducers.* Monographs in Theoretical Computer Science, An EATCS Series. Springer-Verlag, 1998.

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

[GS97] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 1–68. Springer-Verlag, 1997.

[Hue80] G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *J. ACM*, 27:797–821, 1980.

[Iro61] E. T. Irons. A syntax directed compiler for ALGOL 60. *Comm. of the ACM*, 4:51–55, 1961.

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

[Knu71] D. E. Knuth. Semantics of context-free languages: Correction. *Math. Systems Theory*, 5(1):95–96, 1971. Errata of [Knu68].

[Küh98] A. Kühnemann. Benefits of Tree Transducers for Optimizing Functional Programs. In V. Arvind and R. Ramanunjam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1530 of *LNCS*, pages 146–157. Springer-Verlag, 1998.

[KV94] A. Kühnemann and H. Vogler. Synthesized and inherited functions — a new computational model for syntax–directed semantics. *Acta Inform.*, 31:431–477, 1994.

[MBPS05a] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML Type Checking with Macro Tree Transducers. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS' 05)*, pages 283–294. ACM Press, 2005.

[MBPS05b] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML Type Checking with Macro Tree Transducers. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS' 05)*, pages 283–294. ACM Press, 2005.

[MN01] S. Maneth and F. Neven. Recursive structured document transformation. In R. Connor and R. Mendelzon, editors, *Research issues in structured and semistructured database programming - Revised papers DBLP 99*, volume 1949 of *Lect. Notes Comput. Sci.*, pages 80–98. Springer-Verlag, 2001.

[MSS06] A. Muscholl, M. Samuelides, and L. Segoufin. Complementing deterministic tree-walking automata. *Information Processing Letters*, 99:33–39, 2006.

[MSV03] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *J. of Comput. Syst. Sci.*, 66:66–97, 2003.

[Muz08] L. Muzamel. Pebble Alternating Tree-Walking Automata and Their Recognizing Power. *Acta Cybernetica*, 18(3):427–450, 2008.

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

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

[Via01] V. Vianu. A Web Odyssey: From Codd to XML. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems (PODS' 01)*, pages 1–15. ACM Press, 2001.

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

[Voi02] J. Voigtländer. Conditions for Efficiency Improvement by Tree Transducer Composition. In Sophie Tison, editor, *13th International Conference on Rewriting Techniques and Applications, Copenhagen, Denmark, Proceedings*, volume 2378 of *LNCS*, pages 222–236. Springer-Verlag, July 2002.

[WM95] R. Wilhelm and D. Maurer. *Compiler Design*. Addison-Wesley, 1995.