Pebble Macro Tree Transducers with Strong Pebble Handling

Abstract of the PhD Thesis

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**

PhD School in Computer Science

May 24, 2010

1 Preliminaries and the model we consider

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, MBPS05], 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 the corresponding thesis we consider pebble macro tree transducers of [EM03]. An *n-pebble macro tree transducer* (*n-pmtt*) 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. In the remainder of this thesis abstract M stands for an n-pmtt.

M has finitely many rules of the form $\langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$. 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 node v in the current sentential form ξ labelled by a configuration $\langle q, h\rangle$ with $h = (u, [u_1; \ldots; u_l])$, such that there is not any

other configuration nodes in the $\varepsilon - v$ path of ξ, i.e., v is an *outside-active node*. Then M takes a rule $r : \langle q, \sigma, b, j\rangle(y_1, \ldots, y_m) \to \zeta$, 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 $\varphi = \mathit{down}_i$, 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 $\varphi = \mathit{drop}$, 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 η. We denote this transition by $\xi \Rightarrow_{M,s} \eta$.

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;

By a *pebble macro tree transducer (pmtt)* we mean an n-pmtt for some $n \geq 0$. In the remainder of this thesis abstract M stands for an n-pmtt and s an input tree to M. The n-pmtt M 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;
- a *top-down tree transducer* if it is a 0-ptt and an smtt.

The classes of tree transformations computed by n-pmtts, n-ptts, smtts, mtts, and top-down tree transducers are denoted by $n\text{-}PMTT$, $n\text{-}PTT$, $sMTT$, MTT, and T, respectively. The deterministic, total, and context linear subclasses of each above classes are denoted by prefixing the classes with d, t, and cl, respectively. For instance, $n\text{-}dPMTT$ stands for the class of tree transformations computed by deterministic n-pebble macro tree transducers.

In the thesis we prove composition and decompostions results concerning pebble macro tree transducers as well as we prove that circularity problems concerning pebble macro tree tranducers are decidable.

2 Results of the thesis

2.1 Circularity

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. We call this phenomenon circularity.

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 (Lemma 14 of [FM08]) The following statements hold for the n-pmtt M.

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

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

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 or a pebble macro tree transducer is not weakly circular.

2.2 Composition and decomposition results

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 the arbitrary 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 following composition result. We denote by $YIELD$ the class of (nondeterministic) yield tree transformations.

Lemma 6.1 (Lemma 41 of [FM08]) For all $n \geq 0$ we have $n\text{-}PTT \circ YIELD \subseteq n\text{-}PMTT$. ◇

This construction of the above lemma 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, the arbitrary n-pebble macro tree transducer M, we effectively give an n-pebble tree transducer M' and a yield tree transformation $yield_g$ such that $\tau_M = \tau_{M'} \circ yield_g$. Hence we get the following decomposition result.

Corollary 7.9 (Corollary 4.11 of [FM09]) For all $n \geq 0$ we have $n\text{-}PMTT \subseteq n\text{-}PTT \circ YIELD$. ◇

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 characterization result.

Theorem 7.10 (Corollary 4.12 of [FM09]) For all $n \geq 0$ we have $n\text{-}PMTT = n\text{-}PTT \circ YIELD$. ◇

We note that in the construction of the proof of Corollary 7.9 we more or less fol-

low 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 of the thesis.

Therefore, we examine if there is another (semantically correct) decomposition construction which produces M' and $yield_g$ and preserves some good properties of M (like determinism and noncircularity) at least for some reasonably restricted class of pebble macro tree transducers. It turns out that the answer is positive. The first restriction we make concerning M is that it is deterministic (or context linear). Besides this restriction, for providing the semantical correctness, we also need to assume that M' is noncircular. (In the following M' and $yield_g$ denote the ptt and yield transformation, respectively, obtained from M by the alternative construction.)

Lemma 8.4 (Lemma 30 of [FM08]) If M is deterministic and M' is noncircular, then $\tau_M = \tau_{M'} \circ yield_g$ holds. ◇

Lemma 8.6 (Lemma 32 of [FM08]) If M is context-linear and M' is noncircular, then $\tau_M = \tau_{M'} \circ yield_g$ 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 the conditions of Lemmas 8.4 and 8.6 hold and we obtain slightly different versions of the decomposition $dtMTT \subseteq dtT \circ dtYIELD$ of [EV85], namely, the inclusions $dMTT \subseteq dtT \circ dYIELD$ and $clMTT \subseteq tT \circ dYIELD$.

Another natural restriction could 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.

A natural and appropriate restriction is that M is not weakly circular. We show that if M is not weakly circular (nwc) then M' is noncircular (nc). Hence, we obtain that 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 (Corollary 37 of [FM08])

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

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

Since every partial yield tree transformation can be computed by a noncircular and deterministic 0-pebble tree transducer, 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 (Theorem 38 of [FM08])

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

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

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

The next topic we consider in the thesis is the solution of an open problem raised in Section 8 of [EM03]. Namely, we prove that each (deterministic) n-pebble tree transducer M, provided $n \geq 1$, can be simulated by a (deterministic) $(n-1)$-pebble macro tree transducer M'.

Theorem 9.6 (Theorem 5.7 of [FM09]) For each $n \geq 1$, $n\text{-}PTT \subseteq (n-1)\text{-}PMTT$ and $n\text{-}dPTT \subseteq (n-1)\text{-}dPMTT$. ◇

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.

2.4 Further results

There are some important consequences of Theorems 7.10 and 9.6.

Theorem 10.3 (Theorem 6.5 of [FM09]) For each $n \geq 0$,

$$\begin{array}{lrcl} (1) & n\text{-}PMTT & \subseteq & 0\text{-}PTT \circ YIELD^{n+1}, \\ (2) & n\text{-}PMTT & \subseteq & 0\text{-}PTT^{n+2}, \\ (3) & n\text{-}PMTT & \subseteq & sMTT^{n+2}. \end{array}$$

and for each $n \geq 1$,

$$\begin{array}{lrcl} (4) & n\text{-}PTT & \subseteq & 0\text{-}PTT \circ YIELD^{n}, \\ (5) & n\text{-}PTT & \subseteq & 0\text{-}PTT^{n+1}, \\ (6) & n\text{-}PTT & \subseteq & sMTT^{n+1}. \end{array}$$

◇

In the above theorem $sMTT$ denotes 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 $n\text{-}PMTT \subseteq sMTT^{n+2}$ of Theorem 10.3, we obtain the following results.

Theorem 10.4 (Theorem 6.7 of [FM09]) The inverses of compositions of pebble macro tree transformations effectively preserve regularity. ◇

Corollary 10.5 (Corollary 6.8 of [FM09]) The domains of (compositions of) pebble macro tree transformations are effectively regular. ◇

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, MBPS05] 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 the type checking result for pebble tree transducers from the decomposition $n\text{-}PMTT \subseteq sMTT^{n+2}$ of Theorem 10.3 and the fact that the type checking problem for compositions of stay-macro tree transformations is decidable of Corollary 44 of [EM03].

Theorem 10.6 (Theorem 6.9 of [FM09]) The type checking problem of pebble macro tree transformations are decidable. ◇

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 $n\text{-}PMTT \subseteq sMTT^{n+2}$ of Theorem 10.3 that the domains of pebble macro tree transformations are effectively regular (Corollary 10.5). This can be used directly to prove that the various types of circularity problems are decidable.

Theorem 10.8 (Theorem 20 of [FM08]) The strong circularity problem (sc-problem) for pmtts is decidable. ◇

Theorem 10.9 (Theorem 21 of [FM08]) The circularity problem (c-problem) for pmtts is decidable. ◇

Corollary 10.11 (Corollary 36 of [FM08]) The weak circularity problem (wc-problem) for pmtts is decidable. ◇

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. In particular we use deterministic and nonlooping n-pebble alternating tree-walking automata, and the corresponding tree language class is denoted by $n\text{-}PATWA_{nl}$. The deterministic subclass is denoted by $n\text{-}dPATWA_{nl}$.

It turns out that nonlooping n-pebble alternating tree-walking automata recognize the domains of not strongly circular pebble tree transformations. Formally, for each $n \geq 0$ we have $n\text{-}PATWA_{nl} = dom(n\text{-}PTT_{nsc})$. 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. In fact, it immediately follows from the next theorem and from the fact that $n\text{-}PATWA_{nl} = dom(n\text{-}PTT_{nsc})$.

Theorem 10.25 (Theorem 5.7 of [Muz08]) For each $n \geq 0$, $n\text{-}dPATWA_{nl} \subset$

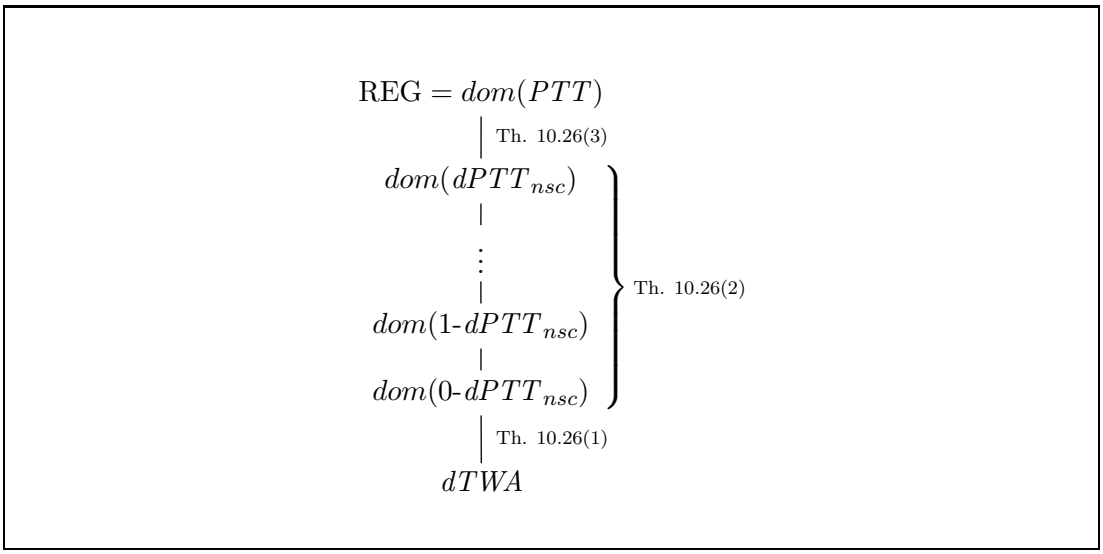

Figure 1: The hierarchy $(dom(n\text{-}dPTT_{nsc}) \mid n \geq 0)$ and its relation to REG and $dTWA$.

$(n+1)$-$dPATWA_{nl}$. ◇

Finally, using the straightforward fact that, for each $n \geq 0$, tree languages recognized by deterministic and non looping n-patwa are the same as domains of deterministic and not strongly circular pebble tree transformations, we obtain the following result.

Theorem 10.26

(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) $dom(dPTT_{nsc}) \subset \mathrm{REG}$. ◇

We note that Theorem 10.26 appears first time in the present thesis. However, all important background was published in [Muz08].

In Figure 1 we visualize the strict hierarchy $(dom(n\text{-}dPTT_{nsc}) \mid n \geq 0)$ and its relation to the class REG of regular tree languages and the class $dTWA$ of tree languages recognizable by deterministc tree-walking automata.

3 Methods applied in the thesis

We mainly apply some standard proof technics of the theory of tree transducers. One of them is the composition, which means that we find a machine (tree transducer) which computes the tree transformation obtained as the composition of tree transformations computed by other machines. With this method we can prove that a class of tree transformations is closed under composition. Another method is the decomposition, meaning that we consider if a tree transformation computed by a certain kind of tree

transducer appears as the composition of two (or more) tree transformations computed by simpler machines. Both methods result substantial information about the working of the tree transducers that we consider. In obtaining the positive decidability results concerning the circularity problems of pebble macro tree transducers we reduce these problems to decidable problems of regular tree languages. At the end of the thesis we compare the classes of domains of pebble tree transducers in the way that we set up the Hasse-diagram of those classes with respect to inclusion.

The main results of the thesis are proved by the following extension of the *simultaneous induction* which was used for macro tree transducers (among others) in [Eng75, FV98].

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

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

IB: $\mathrm{L}(0)$ holds.

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

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

This version of simultaneous induction turned out to be rather useful in proving the correctness of composition and decomposition constructions concerning pebble macro tree transducers because statements could be described as instances of K and L. Besides, we frequently applied the proof by induction on the structure of trees.

4 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.

References

[BMN02] G. J. Bex, S. Maneth, and F. Neven. A formal model for an expressive fragment of XSLT. *Information Systems*, 27:21–39, 2002.

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

[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.

[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.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. System Sci.*, 31:71–146, 1985.

[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.

[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.

[MBPS05] 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.

[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.