Recent Results on Pebble Macro Tree Transducers (Extended abstract)

Zoltán Fülöp [*]
Department of Computer Science
University of Szeged
Árpád tér 2., H-6720 Szeged, Hungary
e-mail: `fulop@inf.u-szeged.hu`

Loránd Muzamel[†]
Department of Computer Science
University of Szeged
Árpád tér 2., H-6720 Szeged, Hungary
e-mail: `muzamel@inf.u-szeged.hu`

1 Preliminaries

The concept of a *pebble tree transducer* was introduced in [MSV03] as a model for XML query languages. Another, but equivalent definition was given in [EM03]. In this paper we use this latter definition.

An n-pebble tree transducer M is a finite state machine that translates input trees to output trees. In this way it computes a binary relation over trees, which is called a tree transformation. Let us take an input tree s to M. M has a reading head, which is a pointer to a node of s and can move to another node along the edges of s. The n pebbles, denoted by $1, \ldots, n$ can be dropped to and lifted from the current node u in a stack-like fashion: if the number of pebbles on s is $l \geq 1$, then pebble l can be lifted (provided it is at the node u), if the number of pebbles is $l < n$, then pebble $(l+1)$ can be dropped at u. 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, this is 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 a configuration node $\langle q, h\rangle$, which is always a leaf of the current sentential form ξ is extended. 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 where pebbles were dropped. 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 a node of s), and the presence of pebbles at node u.

[*]Research of this author was supported by the Hungarian Scientific Foundation (OTKA) under Grant T 030084
[†]Research of this author was partially supported by German Research Foundation (DFG) under GrantGK 334/3

Depending on the test, M may apply one of its finitely many rules to generate a tree which may contain further configurations. This tree is then replaced for $\langle q, h \rangle$ in a first order fashion. In this way M computes the next sentential form ξ'. If a sentential form t is computed which contains no configuration, then it is called an output tree to s and said that M translates s into t.

Moreover, the pebble tree transducer was extended in [EM03] with the capability of making macro calls in the right-hand sides of its rules like the macro tree transducer of [EV85]. The extended machine is called a *pebble macro tree transducer*. The main difference between a pebble tree transducer and a pebble macro tree transducer is the following. Due to the macro calls built in the rules of a pebble macro tree transducer, a configuration node of the sentential form may not only be the leaf but an arbitrary node. Hence, the next sentential form of the pebble macro tree transducer is computed by making a second order tree substitution instead of a first order one.

There are pebble (macro) tree transducers of which not every computation terminates: the computation may lead to an infinite cycle. In this case no output tree is computed. Broadly speaking, we call such a pebble (macro) tree transducer circular.

Note that a 0-pebble (macro) tree transducer can be thought of as a tree-walking transducer, i.e, as a tree transducer which moves along the edges of the input tree and computes an output tree in the way described above, however it has no pebbles to increase its computation power.

In the rest of this paper by a pebble (macro) tree transducer we mean an n-pebble (macro) tree transducer, for some n. Moreover, by an x tree transformation we mean a tree transformation computed by a tree transducer of type x. For instance, an n-pebble tree transformation is a tree transformation that can be computed by an n-pebble tree transducer.

In [EM03] several results were obtained for pebble transformations. As a main result, the following was shown.

Proposition 1.1. ([EM03], Theorem 10) *Every n-pebble (deterministic) tree transformation can be decomposed into $n+1$ 0-pebble (deterministic) tree transformations.*

This result expresses that the computation power of an n-pebble tree transducer can also be achieved by applying consecutively $n + 1$ tree walking transducers which do not use pebbles. Thus the composition closure of n-pebble (deterministic) tree transformations coincide with the composition closure of 0-pebble (deterministic) tree transformations.

2 Results

In the paper [FM05a] we examined how pebble macro tree transformations can be decomposed into simpler tree transformations. Unfortunately we were able to form our decomposition results only for strongly noncircular pebble macro tree transducers. Strong noncircularity is an easily testable restriction for pebble macro tree transducers, which implies noncircularity. Our main decomposition results are the following.

Theorem 2.1. *Every strongly noncircular n-pebble deterministic macro tree transformation is the composition of a noncircular n-pebble deterministic tree transformation and a noncircular deterministic* 0*-pebble tree transformation.*

Roughly speaking, Theorem 2.1. means that the computation power of an n-pebble deterministic macro tree transducer can also be achieved by applying an n-pebble deterministic tree transducer and a 0-pebble deterministic tree transducer. If the n-pebble deterministic macro tree transducer is required to be context-linear, then we get the following decomposition result.

Theorem 2.2. *Every strongly noncircular n-pebble context-linear macro tree transformation is the composition of a noncircular n-pebble tree transformation and a noncircular deterministic* 0*-pebble tree transformation.*

Since pebble tree transducers are also (special) pebble macro tree transducers, Theorem 2.1. and Theorem 2.2. imply the following results.

Theorem 2.3. *The composition closure of strongly noncircular n-pebble deterministic macro tree transformations coincide with the composition closure of noncircular n-pebble deterministic tree transformations.*

Theorem 2.4. *The composition closure of strongly noncircular n-pebble context-linear macro tree transformations coincide with the composition closure of noncircular n-pebble tree transformations.*

Moreover, applying Theorems 2.1. and 2.2. to case $n = 0$, we get the following decomposition results, which are partial answers to an open question raised in Section 8 of [EM03].

Corollary 2.5. *Every strongly noncircular* 0*-pebble deterministic macro tree transformation is the composition of two noncircular* 0*-pebble deterministic tree transformations.*

Corollary 2.6. *Every strongly noncircular* 0*-pebble context-linear macro tree transformation is the composition of two noncircular* 0*-pebble tree transformations.*

Combining Proposition 1.1. with Theorem 2.1. and with Theorem 2.2., respectively, we obtain the following results.

Theorem 2.7. *Every strongly noncircular n-pebble deterministic macro tree transformation is the composition of $n + 1$ noncircular* 0*-pebble deterministic tree transformation and a* 0*-pebble tree deterministic transformation.*

Theorem 2.8. *Every strongly noncircular n-pebble context-linear macro tree transformation is the composition of $n+1$ noncircular 0-pebble deterministic tree transformation and a 0-pebble tree transformation.*

Hence, the computation power of a strongly noncircular n-pebble macro tree transducer can also be achieved by applying consecutively $n+2$ tree-walking transducers which do not use pebbles and do not make macro-calls.

Recently, in [FM05b] we examined the domains of n-pebble tree transformations. In [EM03] it was shown that 0-pebble tree transducers and the attributed tree transducers of [Fül81] are semantically equivalent. Moreover, in [Bar82], it was shown that the domains of partial attributed tree transformations are recognizable tree languages. Hence, the domains of 0-pebble tree transformations are also recognizable tree languages. In [FM05b] we generalized this latter result in the following way.

Theorem 2.9. *The domain of the tree transformation computed by an n-pebble tree transducer is a recognizable tree language.*

Using this result we can also prove the following one.

Theorem 2.10. *The circularity problem of deterministic n-pebble tree transducers is decidable.*

On the other hand, the decidability of the circularity problem for pebble macro tree transducers is still open.

References

[Bar82] M. Bartha. An algebraic definition of attributed transformations. *Acta Cybernet.*, 5:409–421, 1982.

[EM03] J. Engelfriet and S. Maneth. A Comparison of Pebble Tree Transducers with Macro Tree Transducers. *Acta Informatica*, 39:613–698, 2003.

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

[FM05a] Z. Fülöp and L. Muzamel. Decomposition Results for Pebble Macro Tree Transducers. *submitted*, 2005.

[FM05b] Z. Fülöp and L. Muzamel. On the Domains of Pebble Tree Transformations. *manuscript*, 2005.

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

[MSV03] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *J. of Comp. Syst. Sci.*, 66:66–97, 2003.