Acta Informatica (2015) 52:559–571
DOI 10.1007/s00236-015-0224-3

ORIGINAL ARTICLE

Two-way pebble transducers for partial functions and their composition

Joost Engelfriet

Received: 8 August 2014 / Accepted: 13 February 2015 / Published online: 4 March 2015

Abstract Two-way finite state transducers are considered that use a finite number of pebbles, of which the life times must be nested. For every nondeterministic transducer that realizes a partial function, an equivalent deterministic transducer can be constructed. The composition of two deterministic transducers can be realized by one such transducer with a minimal number of pebbles.

1 Introduction

A pebble automaton is a two-way finite state automaton that uses a fixed, finite number of pebbles that it can drop on, and lift from, the squares of its input tape. Pebble automata recognize regular languages only, provided the life times of the pebbles are nested (otherwise they recognize the logarithmic space languages). Automata with nested pebbles were introduced in [21], and in [11] for tree-walking automata. A tree transducer with nested pebbles was introduced in [25] as a model for XML-based query languages. In [15] some results were proved for the two-way finite state transducer with nested pebbles (pebble transducer, for short), which is the restriction of the tree transducer of [25] to monadic trees, i.e., to strings.

One result of [15] is that every partial function that can be computed by a nondeterministic pebble transducer, can in fact be computed by a deterministic one, with the same number of pebbles. Unfortunately, the proof of this result in [15, Theorem 3] is wrong.[1] In this paper the result is stated in Corollary 7, with a (hopefully) correct proof.

Another result of [15] is that for every two deterministic pebble transducers M_1 and M_2, there is a deterministic pebble transducer M that computes the composition of the functions computed by M_1 and M_2. In the proof, the constructed transducer M has $(k_1 + 1)(k_2 + 2)$ pebbles, where k_i is the number of pebbles of M_i. It is conjectured in [15] that a transducer with $(k_1 + 1)(k_2 + 1) - 1$ pebbles can do the same job (and it is easy to see that, in general,

[1] The mistake in the proof is explained in New Observation 5.10 of [10].

J. Engelfriet (✉)
Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
e-mail: j.engelfriet@liacs.leidenuniv.nl

it cannot be done with less pebbles). In this paper the conjecture is proved, and stated in Theorem 11.

Our proofs are based on a reduction of pebble transducers to two-way finite state transducers (without pebbles), which is a straightforward generalization of the reduction of pebble automata to two-way finite state automata in [20]. This allows us to use the well-known facts that a partial function computed by a nondeterministic two-way finite state transducer can also be computed by a deterministic one (see, e.g., Theorem 22 of [12]), and that the deterministic two-way finite state transductions are closed under composition ([6]).

Other work on nested pebbles has appeared in, e.g., [2–5,13,14,16–19,22,26–28,30,32].

All results stated in this paper are *effective*. Results that are known from the literature, or can easily be concluded from results in the literature, are stated as Propositions.

2 Definitions

For binary relations $R_1 \subseteq X \times Y$ and $R_2 \subseteq Y \times Z$, where X, Y, Z are sets, we denote by $R_1 \circ R_2$ their composition $\{(x,z) \mid \exists y : (x,y) \in R_1,\ (y,z) \in R_2\}$. For classes of binary relations $\mathcal{R}_1$ and $\mathcal{R}_2$, we define $\mathcal{R}_1 \circ \mathcal{R}_2 = \{R_1 \circ R_2 \mid R_1 \in \mathcal{R}_1,\ R_2 \in \mathcal{R}_2\}$. For a binary relation $R \subseteq X \times Y$, the domain of R is $\mathsf{dom}(R) = \{x \in X \mid \exists y \in Y : (x,y) \in R\}$, and its range is $\mathsf{ran}(R) = \{y \in Y \mid \exists x \in X : (x,y) \in R\}$.

For a set Δ, the set of all strings over Δ (i.e., sequences of elements of Δ) is denoted Δ^*. For a string $w \in \Delta^*$, its length is denoted $|w|$, and $w(i)$ denotes its ith element (for $1 \leq i \leq |w|$). The empty string is denoted λ. For $k \in \mathbb{N} = \{0, 1, 2, \dots\}$, we define $\Delta^k = \{w \in \Delta^* \mid |w| = k\}$ and $\Delta^{\leq k} = \{w \in \Delta^* \mid |w| \leq k\}$.

2.1 Pebble transducers

A two-way finite state transducer (also called two-way generalized sequential machine, or 2gsm) is a finite state automaton with a two-way input tape, delimited by the endmarkers $\triangleleft$ and $\triangleright$, and a one-way output tape. A *k-pebble transducer* is a two-way finite state transducer that additionally carries k pebbles, each with a unique name, say, $1, \dots, k$. Initially there are no pebbles on the input tape, but during its computation the transducer can drop pebbles on the squares of the input tape, lift them, drop them again, etc. In one step of its computation, the transducer can determine which pebbles are lying on the current square of the input tape, lift one of these pebbles, or drop a new pebble. However, the life times of the pebbles should be nested. This means that at each moment of time, pebbles 1 to ℓ (for some $0 \leq \ell \leq k$) are on the input tape, and at such a moment the only pebble that can be dropped on the current square is pebble $\ell + 1$, whereas the only pebble that can be lifted from the current square (if it is on that square) is pebble ℓ.

Formally, a *k-pebble transducer* (with $k \geq 0$) is a system $M = (\Sigma, \Delta, Q, q_0, F, \delta)$ where Σ and Δ are the input and output alphabet, respectively, Q is the finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and δ is the finite set of transitions. Each transition is of the form $(q, \sigma, b) \mapsto (q', \varphi, w)$ with $q \in Q - F$, $\sigma \in \Sigma \cup \{\triangleleft, \triangleright\}$, $b \in \{0,1\}^k$, $q' \in Q$, $\varphi \in \{\mathsf{right}, \mathsf{left}, \mathsf{drop}, \mathsf{lift}\}$, and $w \in \Delta^*$. Intuitively, such a transition means that if M is in state q, σ is the symbol on the current square, and, for every $1 \leq m \leq k$, $b(m) = 1$ if and only if pebble m is on the current square, then M can go into state q', output the string w, and execute the instruction φ, i.e., move the reading head one square to the right or left, drop pebble $\ell + 1$ (the “next” pebble), or lift pebble ℓ (the “last” pebble); note that these instructions are undefined if, respectively, $\sigma = \rhd$, $\sigma = \lhd$, all k pebbles are on the input tape (i.e., $\ell = k$), or pebble ℓ is not on the current square (i.e., either $\ell = 0$ or $b(\ell) = 0$).

For a given input string $u \in \Sigma^*$, the input tape of M contains the string $\lhd u \rhd$. The squares of this tape are numbered $0, 1, \dots, |u|, |u|+1$. Accordingly, a *pebble configuration* of M on $\lhd u \rhd$ is a string $\pi \in \{0, \dots, |u|+1\}^{\leq k}$, where $\pi(m) = j$ means that pebble m is currently on square j (for $1 \leq m \leq |\pi|$ and $j \in \{0, \dots, |u|+1\}$), and $|\pi| < m \leq k$ means that pebble m is currently not on the input tape. A *configuration* of M on $\lhd u \rhd$ is a triple (q, i, π) with $q \in Q$, the current state of M, $i \in \{0, \dots, |u|+1\}$, the current position of the reading head, and $\pi \in \{0, \dots, |u|+1\}^{\leq k}$, the current pebble configuration. The one step computation relation $\vdash_u$ is defined in the obvious way on 4-tuples (q, i, π, v) where (q, i, π) is a configuration and $v \in \Delta^*$ is the current content of the output, as follows.

Let σ be the content of the current input square, i.e., $\sigma = \lhd$ if $i = 0$, $\sigma = u(i)$ if $1 \leq i \leq |u|$, and $\sigma = \rhd$ if $i = |u|+1$. Let $b \in \{0,1\}^k$ indicate which pebbles are placed on this square, i.e., for every $1 \leq m \leq k$, $b(m) = 1$ if and only if $\pi(m) = i$. If δ contains a transition $(q, \sigma, b) \mapsto (q', \varphi, w)$, then $(q, i, \pi, v) \vdash_u (q', i', \pi', vw)$ if the following holds:

if $\varphi = \mathsf{right}$, then $i \neq |u|+1$, $i' = i+1$ and $\pi' = \pi$,
if $\varphi = \mathsf{left}$, then $i \neq 0$, $i' = i-1$ and $\pi' = \pi$,
if $\varphi = \mathsf{drop}$, then $i' = i$, $|\pi| < k$ and $\pi' = \pi i$,
if $\varphi = \mathsf{lift}$, then $i' = i$, $\pi(|\pi|) = i$ and $\pi = \pi' i$.

The *transduction computed by M*, denoted τ_M, is the binary relation

$$\tau_M = \{(u, v) \in \Sigma^* \times \Delta^* \mid \exists (q, i, \pi) : (q_0, 0, \lambda, \lambda) \vdash_u^* (q, i, \pi, v),\ q \in F\}.$$

The k-pebble transducer M is *deterministic* if δ does not contain two transitions with the same left-hand side. In that case, τ_M is a partial function from Σ^* to Δ^*.

The class of transductions computed by k-pebble transducers is denoted by PT_k, and by DPT_k for the deterministic transducers.

Example 1 A deterministic 4-pebble transducer M can translate an input string u into an output string that is a concatenation of all strings $v\#w\#$ where v and w are (occurrences of) nonempty substrings of u and vw is in a given regular language R. The transducer M moves pebble 1 from square 1 to square n of the input tape, where $n = |u|$. When pebble 1 is at square i, M moves pebble 2 from square i to square n. In this way M systematically considers all nonempty substrings v of u. For a fixed position of pebbles 1 and 2, M systematically considers all nonempty substrings w of u, using pebbles 3 and 4 for that purpose. For each position of all 4 pebbles, M walks from pebble 1 to pebble 2 and then from pebble 3 to pebble 4, simulating a finite automaton that recognizes the regular language R. If $vw \in R$, then M repeats that walk and outputs the string $v\#w\#$. □

2.2 Counting transducers

Our first result (in the next section) will be the equivalence of the k-pebble transducer with a related type of two-way pebble transducer that uses its k pebbles in a very restricted, nonstandard way. Rather than manipulating pebbles by dropping and lifting them, a transducer M of this new type has all its k pebbles on the input tape at all times. To explain how M can move the pebbles, we first note that a pebble configuration $\pi \in \{0, \dots, |u|+1\}^k$ on $\lhd u \rhd$ (with all pebbles on the input tape) can be viewed as usual as a number $\mathsf{num}(\pi)$ in the $(|u|+2)$-ary number system, with $0 \leq \mathsf{num}(\pi) \leq (|u|+2)^k - 1$. As an example, for $k = 3$ and $|u| = 8$, the pebble configuration 097 (which means that pebble 1 is on $\lhd$, pebble 2 is on $\rhd$, and pebble

3 is on the 7th symbol of u) corresponds to the (decimal) number 97; in this case the pebble configurations on u of length k represent all numbers between 0 (all pebbles on $\triangleleft$) and 999 (all pebbles on $\triangleright$). For pebble configurations π_1 and π_2 such that $\mathsf{num}(\pi_1)+1 = \mathsf{num}(\pi_2)$, we say that π_2 is the *next* pebble configuration of π_1, and that π_1 is the *previous* pebble configuration of π_2. It is well known that a k-pebble transducer (as defined in Sect. 2.1) can count from 1 to $(|u|+2)^k$ by systematically constructing all pebble configurations of length k, going from one to the next, starting with the *first* pebble configuration of length k (representing 0, with all pebbles on $\triangleleft$) and ending with the *last* one (representing $(|u|+2)^k - 1$, with all pebbles on $\triangleright$). The transducer M of the new type only has the instructions right and left, which are now also defined when the current symbol under the reading head is $\triangleright$ or $\triangleleft$, respectively (provided the current pebble configuration is not the last or the first, respectively). The effect of moving right from $\triangleright$, is that the reading head jumps to $\triangleleft$ and the pebble configuration is changed to the next one. Similarly, the effect of moving left from $\triangleleft$, is that the reading head jumps to $\triangleright$ and the pebble configuration is changed to the previous one. We note here that (as in [25]) the reading head can be viewed as an additional pebble, viz. pebble $k+1$, and so a pair (i,π) consisting of the current position of the reading head and the current pebble configuration, can be viewed as the pebble configuration $\pi i \in \{0,\dots,|u|+1\}^{k+1}$ on $\triangleleft\, u \triangleright$. Viewed in this way, the instruction right changes the pebble configuration πi to the next one, and the instruction left changes it to the previous one. This shows the naturalness of the new interpretation of right and left. We will call the new type of transducer a k-counting transducer.

Formally, a *k-counting transducer* (with $k \geq 0$) is a system $M = (\Sigma, \Delta, Q, q_0, F, \delta)$, defined in the same way as a k-pebble transducer except that it does not have instructions drop and lift, i.e., $\varphi \in \{\mathsf{right}, \mathsf{left}\}$ in every transition $(q,\sigma,b) \mapsto (q',\varphi,w)$. A pebble configuration of M on $\triangleleft\, u \triangleright$ (with $u \in \Sigma^*$) is a string $\pi \in \{0,\dots,|u|+1\}^k$, and a configuration of M on $\triangleleft\, u \triangleright$ is of the form (q,i,π), as for the k-pebble transducer (but with the pebble configuration of length k). The one step relation $\vdash_u$ is defined as for the k-pebble transducer on 4-tuples (q,i,π,v), except that now:

$$\begin{aligned}
&\text{if } \varphi = \mathsf{right} \text{ then either } i \neq |u|+1,\ i' = i+1 \text{ and } \pi' = \pi,\\
&\qquad \text{or } i = |u|+1,\ i' = 0, \text{ and } \pi' \text{ is the next pebble configuration of } \pi,\\
&\text{if } \varphi = \mathsf{left} \text{ then either } i \neq 0,\ i' = i-1 \text{ and } \pi' = \pi,\\
&\qquad \text{or } i = 0,\ i' = |u|+1, \text{ and } \pi' \text{ is the previous pebble configuration of } \pi.
\end{aligned}$$

Finally, the *transduction computed by M*, denoted τ_M, is

$$\tau_M = \{(u,v) \in \Sigma^* \times \Delta^* \mid \exists (q,i,\pi) : (q_0, 0, 0^k, \lambda) \vdash_u^* (q,i,\pi,v),\ q \in F\}.$$

Note that 0^k denotes a string of k 0's; it is the first pebble configuration. Determinism of M is defined as for the k-pebble transducer.

The class of transductions computed by k-counting transducers is denoted by CT_k, and by DCT_k for the deterministic transducers.

Example 2 A 4-counting transducer M' that computes the same transduction as the 4-pebble transducer M of Example 1, systematically considers all possible pebble configurations by repeatedly moving to the right. For each such pebble configuration it checks that the pebbles are not on an endmarker, and it checks that pebble 2 is not to the left of pebble 1, and pebble 4 is not to the left of pebble 3. If so, pebbles 1 and 2 determine a nonempty substring v of the input string u, and pebbles 3 and 4 determine a nonempty substring w. Then M' operates in the same way as M. □

Whenever we construct a k-pebble or k-counting transducer, we can also use transitions containing an identity instruction $\varphi = \mathsf{stay}$, with the semantics $i' = i$ and $\pi' = \pi$. Such an instruction can easily be simulated in two steps, first moving right and then moving left (or vice versa if the reading head is on $\triangleright$).

3 Equivalence of pebble and counting transducers

This section contains a basic result, underlying the proofs of our main results. We first show that the two types of two-way pebble transducer, defined in the previous section, are equivalent, and then we use this to characterize the k-pebble transductions in terms of 0-pebble transductions. One direction of the characterization was shown for $k = 1$ in the proof of Lemma 1 of [15]. For arbitrary k the characterization was presented for transducers with empty output alphabet, i.e., for automata, in [20] (and rediscovered by this author).

Lemma 3 *For every $k \in \mathbb{N}$, $PT_k = CT_k$ and $DPT_k = DCT_k$.*

Proof We first show the obvious fact that every k-counting transducer M can be simulated by a k-pebble transducer M'. The transducer M' initializes the simulation by dropping all its pebbles on $\triangleleft$, and then stepwise simulates M. The instruction right of M is simulated by the same instruction of M' when the reading head is not on $\triangleright$. Now assume that the reading head *is* on $\triangleright$. If all pebbles are on $\triangleright$, then M' aborts. Otherwise, M' must construct the next pebble configuration of M. To do this, M' first lifts all pebbles $m + 1$ to k, where m is the largest number such that pebble m is not on $\triangleright$. Then M' walks to the left, finds pebble m and moves it one square to the right. Finally, M' walks to the left until it is on $\triangleleft$, and drops pebbles $m + 1$ to k. The instruction left of M is simulated in a symmetrical way. Note that the initialization and the simulation of each step are deterministic subroutines. Thus, $\mathrm{CT}_k \subseteq \mathrm{PT}_k$ and $\mathrm{DCT}_k \subseteq \mathrm{DPT}_k$.

We now show that every k-pebble transducer M can be simulated by a k-counting transducer M'. Again, M' stepwise simulates M. A pebble configuration of M with pebbles 1 to ℓ on the input tape is simulated by the pebble configuration of M' where pebbles 1 to ℓ are on the same squares as those of M, and pebbles $\ell + 1$ to k are all on the same square as the reading head. The number ℓ is kept in the finite state of M'. Note that M' already starts in the correct configuration (with $\ell = 0$ in its finite state). The simulation of the instructions drop and lift is easy: $\ell := \ell + 1$ and $\ell := \ell - 1$, respectively, in the finite state. To simulate an instruction right, M' first checks that the reading head is not on $\triangleright$. Then M' has to move its reading head and all pebbles $\ell + 1$ to k one square to the right. To do this, M' repeatedly moves to the right (i.e., executes the instruction right with the semantics of the counting transducer) until the reading head and all pebbles $\ell + 1$ to k are again on the same square. Note that each time M' moves to the right from $\triangleright$, the next pebble configuration is realized. For instance, for $|u| = 8$, $k = 4$ and $\ell = 2$, if $\mathsf{num}(\pi) = 7444$ for the current pebble configuration π and the reading head is on square 4, then, after the simulation of right, $\mathsf{num}(\pi') = 7455$ for the new pebble configuration π', and the reading head is on square 5; note that 74555 is the first number after 74444 for which the three least significant digits are equal. The simulation of the instruction left is symmetrical. Again, the simulation of each step is a deterministic subroutine, and so $\mathrm{PT}_k \subseteq \mathrm{CT}_k$ and $\mathrm{DPT}_k \subseteq \mathrm{DCT}_k$. □

In the remainder of this section we show that the k-counting transductions, and hence the k-pebble transductions, can be characterized as the compositions of a very specific kind of deterministic k-counting transductions with the 0-counting transductions (i.e., 2gsm's). Let us now define these specific k-counting transductions. There is one for each input alphabet Σ, called $\mathsf{peb}_{k,\Sigma}$; it is the mapping P_k in [20].

For $u \in \Sigma^*$, a string $\pi \in \{0, \ldots, |u|+1\}^k$ will be called a *k-pebble configuration* on $\triangleleft\, u \triangleright$; it is a pebble configuration of a k-counting transducer with input alphabet Σ. We will need an obvious encoding of the pair $(\triangleleft\, u \triangleright, \pi)$ as a string over the alphabet $(\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0, 1\}^k$. If $u = \sigma_1 \cdots \sigma_n$ with $n \geq 0$ and $\sigma_i \in \Sigma$, then

$$\mathsf{code}(\triangleleft\, u \triangleright, \pi) = (\triangleleft, b_0)(\sigma_1, b_1) \cdots (\sigma_n, b_n)(\triangleright, b_{n+1})$$

where $b_i(m) = 1$ if and only if $\pi(m) = i$, for every $0 \leq i \leq n+1$ and $1 \leq m \leq k$.

We define the mapping $\mathsf{pebb}_{k,\Sigma}$ (with two b's!) such that for $u \in \Sigma^*$,

$$\mathsf{pebb}_{k,\Sigma}(u) = \mathsf{code}(\triangleleft\, u \triangleright, \pi_0) \cdots \mathsf{code}(\triangleleft\, u \triangleright, \pi_s)$$

where $s = (|u|+2)^k - 1$ and $\mathsf{num}(\pi_j) = j$ for every $0 \leq j \leq s$. In other words, $\mathsf{pebb}_{k,\Sigma}(u)$ is the concatenation of all consecutive (encodings of) k-pebble configurations on $\triangleleft\, u \triangleright$. To define our intended specific k-counting transductions, we note that the first symbol of $\mathsf{pebb}_{k,\Sigma}(u)$ is $(\triangleleft, 1^k)$ and its last symbol is $(\triangleright, 1^k)$, where 1^k denotes a string of k 1's; moreover, these symbols do not occur anywhere else in $\mathsf{pebb}_{k,\Sigma}(u)$.

We now define the mapping $\mathsf{peb}_{k,\Sigma}$ (with one b!) such that for $u \in \Sigma^*$,

$$\mathsf{pebb}_{k,\Sigma}(u) = (\triangleleft, 1^k) \cdot \mathsf{peb}_{k,\Sigma}(u) \cdot (\triangleright, 1^k).$$

Thus, $\mathsf{peb}_{k,\Sigma}(u)$ is the result of removing the first and last symbol from $\mathsf{pebb}_{k,\Sigma}(u)$. In the proof of the next theorem (Theorem 4), a transducer that receives $\mathsf{peb}_{k,\Sigma}(u)$ as input, will view $\triangleleft$ as $(\triangleleft, 1^k)$ and $\triangleright$ as $(\triangleright, 1^k)$; in this way, the transducer views the content of its input tape as $\mathsf{pebb}_{k,\Sigma}(u)$.

For $k \geq 0$, we denote by PEB_k the class of all mappings $\mathsf{peb}_{k,\Sigma}$, where Σ is a ranked alphabet.

It is easy to see that $\mathsf{pebb}_{k,\Sigma}$ is in DCT_k: by repeatedly moving right, a deterministic k-counting transducer M can consecutively go through the k-pebble configurations π_0 to π_s, and for each $0 \leq j \leq s$ output the string $\mathsf{code}(\triangleleft\, u \triangleright, \pi_j)$. To be precise, M has initial state q and final state q', and it has the transitions $(q, \sigma, b) \mapsto (q, \mathsf{right}, (\sigma, b))$ for all $(\sigma, b) \neq (\triangleright, 1^k)$, plus the transition $(q, \triangleright, 1^k) \mapsto (q', \mathsf{left}, (\triangleright, 1^k))$. It should be clear that therefore also $\mathsf{peb}_{k,\Sigma}$ is in DCT_k, just suppressing the first and last symbol of the output of M. Consequently $\mathrm{PEB}_k \subseteq \mathrm{DPT}_k$, by Lemma 3.

We now state the characterization of the k-pebble transductions, of which the proof is intuitively obvious from Lemma 3.

Theorem 4 *For every $k \geq 0$, $PT_k = PEB_k \circ PT_0$ and $DPT_k = PEB_k \circ DPT_0$.*

Proof Using Lemma 3, we prove that $\mathrm{CT}_k = \mathrm{PEB}_k \circ \mathrm{CT}_0$ and $\mathrm{DCT}_k = \mathrm{PEB}_k \circ \mathrm{DCT}_0$. For a k-counting transducer $M = (\Sigma, \Delta, Q, q_0, F, \delta)$ we define the *associated* 0-counting transducer $M' = (\Sigma', \Delta, Q, q_0, F, \delta')$ where $\Sigma' = (\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0, 1\}^k$ and δ' is defined as follows. If $(q, \sigma, b) \mapsto (q', \varphi, w)$ is a transition in δ, then δ' contains the transition $(q, (\sigma, b), \lambda) \mapsto (q', \varphi, w)$, where $(\triangleleft, 1^k)$ is identified with $\triangleleft$ and $(\triangleright, 1^k)$ with $\triangleright$. Note that M' is deterministic if and only if M is deterministic. It should be clear that $\mathsf{peb}_{k,\Sigma} \circ \tau_{M'} = \tau_M$. Indeed, if M moves right from $\triangleright$, then M' moves right from some $(\triangleright, b)$ that is not identified with $\triangleright$, and so M' moves from the last symbol of $\mathsf{code}(\triangleleft\, u \triangleright, \pi_j)$, where u is the input string and π_j is the current pebble configuration of M, to the first symbol of $\mathsf{code}(\triangleleft\, u \triangleright, \pi_{j+1})$. And similarly when M moves left from $\triangleleft$. As long as M does not move right from $\triangleright$ or move left from $\triangleleft$, M' stays on the substring $\mathsf{code}(\triangleleft u \triangleright, \pi_j)$ of $\mathsf{pebb}_{k,\Sigma}(u) = \triangleleft\, \mathsf{peb}_{k,\Sigma}(u) \triangleright$, simulating M.

This shows that $\mathrm{CT}_k \subseteq \mathrm{PEB}_k \circ \mathrm{CT}_0$ and $\mathrm{DCT}_k \subseteq \mathrm{PEB}_k \circ \mathrm{DCT}_0$. The reverse inclusions hold because, obviously, every 0-counting transducer with input alphabet $(\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0,1\}^k$ is the associated transducer of a k-counting transducer with input alphabet Σ. $\square$

The constructions involved in this theorem, for the case that the output alphabet is empty, are used in [20] to reduce the descriptional complexity of two-way k-pebble automata to that of two-way (0-pebble) automata: if nondeterministic two-way automata can be simulated by deterministic two-way automata with a polynomial number of states (which is not known, cf. [29]), then the same is true for k-pebble automata. We will use Theorem 4 in the following sections in a similar way, to transfer results for 2gsm's to k-pebble transducers.

4 Uniformizers

In this section we use Theorem 4 to prove our first main result: that every partial function in PT_k is in DPT_k. In fact, we will prove the stronger result that every transduction in PT_k has a uniformizer in DPT_k. A *uniformizer* of a binary relation R is a binary relation R' such that $R' \subseteq R$ and $\mathsf{dom}(R') = \mathsf{dom}(R)$, see, e.g., [1,8]. Note that the only uniformizer of a partial function f is f itself. We first prove the result for $k = 0$, i.e., for 2gsm's. A proof was already sketched in the proof of Theorem 4.9 of [9]. Here we give a proof using the results of [12].

Proposition 5 *Every transduction in PT_0 has a uniformizer in DPT_0.*

Proof A *writing 0-pebble transducer* is a 0-pebble transducer that, in addition, can write on its input tape. For this purpose, it has an additional "work alphabet" Ω that contains the input alphabet and the endmarkers. Its transitions are of the form $(q, \sigma, b) \mapsto (q', \sigma', \varphi, w)$ with $\sigma, \sigma' \in \Omega$, meaning that σ is overwritten by σ' in the current square. A *Hennie machine* is a writing 0-pebble transducer M for which there is a number $k \in \mathbb{N}$ such that for each $(u, v) \in \tau_M$ there is a *k-visiting* computation of M on input u with output v, which means that each square of the input tape is visited at most k times. For more details see Section 7 of [12]. The Hennie machine was introduced as an accepting device in [23], where it was shown that, due to the k-visiting restriction, it accepts regular languages only. It was shown in [12] that the Hennie machine computes the MSO definable string transductions, i.e., the string transductions that are definable in monadic second-order logic.

It is easy to see that for every 0-pebble transducer M there is a Hennie machine M' such that $\tau_{M'}$ is a uniformizer of τ_M. In fact, if $(u, v) \in \tau_M$ then M has a computation on input u (with some output v') that never enters the same state twice at the same square (because if, during some computation, M enters state q at square j on time t_1 and on time $t_2 > t_1$, then the subcomputation between t_1 and t_2 can be skipped). Such a computation is k-visiting, where k is the number of states of M. A Hennie machine M' can easily simulate the k-visiting computations of M using the work alphabet $(\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0, 1, \ldots, k\}$ (where Σ is the input alphabet) and counting the visits to each square in the symbol at that square. To be precise, each transition $(q, \sigma, b) \mapsto (q', \varphi, w)$ of M is replaced by all transitions $(q, (\sigma, i), b) \mapsto (q', (\sigma, i+1), \varphi, w)$ of M' with $0 \le i < k$, where we identify $(\sigma, 0)$ with σ.

By Theorem 25 of [12], every transduction computed by a Hennie machine is a nondeterministic MSO definable string transduction (defined in Section 6.1 of [12]). By the proof of Theorem 21 of [12], for every nondeterministic MSO definable string transduction $\tau' = \tau_{M'}$

there is a *deterministic* MSO definable string transduction τ'' (defined in Section 4 of [12]) that is a uniformizer of τ' and hence a uniformizer of τ_M. By Theorem 13 of [12], τ'' can be computed by a deterministic 0-pebble transducer M'', and so $\tau_{M''} = \tau''$ is a uniformizer of τ_M. □

We now prove our first main result.

Theorem 6 *For every $k \geq 0$, every transduction in PT_k has a uniformizer in DPT_k.*

Proof By Theorem 4 every transduction in PT_k is of the form $\mathtt{peb}_{\Sigma,k} \circ \tau$ with $\tau \in \mathrm{PT}_0$. By Proposition 5, τ has a uniformizer $\tau' \in \mathrm{DPT}_0$. This clearly implies that $\mathtt{peb}_{\Sigma,k} \circ \tau'$ is a uniformizer of $\mathtt{peb}_{\Sigma,k} \circ \tau$, and $\mathtt{peb}_{\Sigma,k} \circ \tau'$ is in DPT_k by Theorem 4. □

Corollary 7 *For every $k \geq 0$, if $\tau \in PT_k$ is a partial function, then $\tau \in DPT_k$.*

5 Composition

In this section we prove that the composition of the transductions of two deterministic pebble transducers with k and m pebbles respectively, can be computed by a deterministic pebble transducer with $(k+1)(m+1)-1 = km+k+m$ pebbles. For $k = m = 0$, i.e., for deterministic 2gsm's, this was proved in [6] (see also Theorems 8.10 and 7.14 of [7]).

Proposition 8 *$DPT_0 \circ DPT_0 \subseteq DPT_0$.*

Let PEBB_k denote the class of all mappings $\mathtt{pebb}_{k,\Sigma}$, where Σ is a ranked alphabet. It should be clear that $\mathrm{PEBB}_k \subseteq \mathrm{PEB}_k \circ \mathrm{DPT}_0$ and $\mathrm{PEB}_k \subseteq \mathrm{PEBB}_k \circ \mathrm{DPT}_0$ (where the 0-pebble transducer adds or removes the first and last symbol, respectively). This, together with Proposition 8 and Theorem 4, gives the following corollary.

Corollary 9 *For every $k \geq 0$, $DPT_k = PEBB_k \circ DPT_0$.*

From Proposition 8 and Corollary 9 (or Theorem 4) we immediately obtain the inclusion $\mathrm{DPT}_k \circ \mathrm{DPT}_0 \subseteq \mathrm{DPT}_k$. In the next lemma we prove a special case of the symmetric inclusion $\mathrm{DPT}_0 \circ \mathrm{DPT}_k \subseteq \mathrm{DPT}_k$.

Lemma 10 *For every $k \geq 0$, $DPT_0 \circ PEBB_k \subseteq DPT_k$.*

Proof Let $\tau \in \mathrm{DPT}_0$ where τ is a partial function from Σ^* to Δ^*. We will construct a deterministic k-pebble transducer N such that $\tau_N = \tau \circ \mathtt{pebb}_{\Delta,k}$.

Let $M = (\Sigma, \Delta \cup \{\lhd, \rhd\}, Q, q_0, F, \delta)$ be a deterministic 0-pebble transducer such that $\tau_M(u) = \lhd\,\tau(u)\,\rhd$ for every $u \in \Sigma^*$. A configuration (q, i, λ) of M (where λ is the empty pebble configuration) will be denoted by (q, i). Without loss of generality, we assume that at each computation step M outputs at most one symbol.

Consider an input string $u \in \mathrm{dom}(\tau_M)$. Each occurrence of an output symbol in $\tau_M(u)$ is produced by M in a certain configuration (q, i) during the next computation step, where q is a state and i is a square of the input tape $\lhd\, u\, \rhd$. By the above assumption, this configuration is unique, because if two such occurrences would be produced by M in the same configuration, then the computation of M on input u would loop, contradicting the fact that $u \in \mathrm{dom}(\tau_M)$. We may assume that $\lhd$ is produced in configuration $(q_\lhd, 0)$ and $\rhd$ in configuration $(q_\rhd, 0)$ for certain states $q_\lhd$ and $q_\rhd$ (probably the initial state and a final state of M); thus, M produces output $\lhd$ and $\rhd$ when it reads $\lhd$.

In the computations of N, a k-pebble configuration $\pi \in \{0, 1, \dots, |v|, |v|+1\}^k$ on the output string $\tau_M(u) = \triangleleft\, v \,\triangleright$, where $v = \tau(u)$, is uniquely represented by a pair $(\alpha, \widetilde{\pi})$ where $\alpha \in Q^k$ associates a state with each pebble and $\widetilde{\pi} \in \{0, 1, \dots, |u|, |u|+1\}^k$ is a pebble configuration of N on $\triangleleft\, u \,\triangleright$, such that, for every $1 \leq m \leq k$, the pair $(\alpha(m), \widetilde{\pi}(m))$ is the configuration in which M produces (the symbol in) square $\pi(m)$ of the output tape. The association α is kept in the finite state of N.

For fixed $(\alpha, \widetilde{\pi})$, the string $\mathsf{code}(\triangleleft\, v \,\triangleright, \pi)$ can be produced by N as output, by simulating the computation of M on input u. When M, in configuration (q, i), outputs a symbol $a \in \Delta \cup \{\triangleleft, \triangleright\}$ during the next computation step, N instead outputs the symbol (a, b) with $b \in \{0, 1\}^k$ such that, for every $1 \leq m \leq k$, $b(m) = 1$ if and only if $(\alpha(m), \widetilde{\pi}(m)) = (q, i)$, i.e., pebble m is in the current square and is associated with the current state of M.

Next we explain how N can realize the representation of the next k-pebble configuration of π, for given $(\alpha, \widetilde{\pi})$. First N walks to $\triangleleft$ and determines the largest number m, $1 \leq m \leq k$, such that $(\alpha(m), \widetilde{\pi}(m)) \neq (q_\triangleright, 0)$ (which means that pebble m is not on the last symbol of $\triangleleft\, v \,\triangleright$). If there is no such m, then N halts in a final state. If m exists, then N lifts the pebbles $m+1$ to k. After that, N must move pebble m one square to the right on $\triangleleft\, v \,\triangleright$. To do that, it again simulates the computation of M on input u. On the moment that M is in configuration $(\alpha(m), \widetilde{\pi}(m))$, N lifts pebble m and continues the simulation of M until the next output symbol is produced. Then N drops pebble m on the current square, associates it with the current state of M, and stops the simulation. Finally, N walks to $\triangleleft$, drops the pebbles $m+1$ to k, and associates them with $q_\triangleleft$.

Initially, N checks that the input string u is in the domain of τ_M. Since the domain of τ_M is regular (see, e.g., [31]), N can simulate a finite automaton recognizing it. If $u \in \mathsf{dom}(\tau_M)$, then N drops all its pebbles on $\triangleleft$ and associates them with $q_\triangleleft$. □

We now prove our second main result.

Theorem 11 *For every $k, m \in \mathbb{N}$, $DPT_k \circ DPT_m \subseteq DPT_{km+k+m}$.*

Proof We first observe that it suffices to show that

$$\mathrm{PEB}_k \circ \mathrm{PEBB}_m \subseteq \mathrm{PEBB}_{km+k+m} \circ \mathrm{DPT}_0.$$

Indeed, $\mathrm{DPT}_k \circ \mathrm{DPT}_m = \mathrm{PEB}_k \circ \mathrm{DPT}_0 \circ \mathrm{DPT}_m = \mathrm{PEB}_k \circ \mathrm{DPT}_0 \circ \mathrm{PEBB}_m \circ \mathrm{DPT}_0 \subseteq \mathrm{PEB}_k \circ \mathrm{DPT}_m \circ \mathrm{DPT}_0 = \mathrm{PEB}_k \circ \mathrm{PEBB}_m \circ \mathrm{DPT}_0 \circ \mathrm{DPT}_0 \subseteq \mathrm{PEB}_k \circ \mathrm{PEBB}_m \circ \mathrm{DPT}_0$ by Theorem 4, Corollary 9, Lemma 10, Corollary 9, and Proposition 8, respectively. If the above inclusion holds, then this is included in $\mathrm{PEBB}_{km+k+m} \circ \mathrm{DPT}_0 \circ \mathrm{DPT}_0 \subseteq \mathrm{PEBB}_{km+k+m} \circ \mathrm{DPT}_0 = \mathrm{DPT}_{km+k+m}$ by Proposition 8 and Corollary 9.

Let $r = km + k + m$. Let Σ be an alphabet, let $\Delta = (\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0, 1\}^k$, let $\Gamma = (\Sigma \cup \{\triangleleft, \triangleright\}) \times \{0, 1\}^r$, and let $\Omega = (\Delta \cup \{\triangleleft, \triangleright\}) \times \{0, 1\}^m$. We will define a deterministic 0-pebble transducer M with input alphabet Γ and output alphabet Ω, such that

$$\tau_M(\mathsf{pebb}_{r,\Sigma}(u)) = \mathsf{pebb}_{m,\Delta}(\mathsf{peb}_{k,\Sigma}(u))$$

for every $u \in \Sigma^*$. In what follows we identify $(\triangleleft, 1^k)$ with $\triangleleft$ and $(\triangleright, 1^k)$ with $\triangleright$, in the alphabet Δ. Thus $\Delta \cup \{\triangleleft, \triangleright\} = \Delta$, and so the symbols in Ω are of the form $((\sigma, b_1), b_2)$ with $\sigma \in \Sigma \cup \{\triangleleft, \triangleright\}$, $b_1 \in \{0, 1\}^k$ and $b_2 \in \{0, 1\}^m$.

For $u \in \Sigma^*$, we recall that $\mathsf{pebb}_{k,\Sigma}(u) = \mathsf{code}(\triangleleft\, u \,\triangleright, \pi_0) \cdots \mathsf{code}(\triangleleft\, u \,\triangleright, \pi_s)$ where $\pi_0, \dots, \pi_s$ is the sequence of consecutive k-pebble configurations on $\triangleleft\, u \,\triangleright$. Due to the above identification,

$$\mathsf{pebb}_{m,\Delta}(\mathsf{peb}_{k,\Sigma}(u)) = \mathsf{code}(\mathsf{pebb}_{k,\Sigma}(u), \rho_0) \cdots \mathsf{code}(\mathsf{pebb}_{k,\Sigma}(u), \rho_t)$$

where $\rho_0, \ldots, \rho_t$ is the sequence of consecutive m-pebble configurations on $\mathsf{pebb}_{k,\Sigma}(u)$. We also have that $\mathsf{pebb}_{r,\Sigma}(u) = \mathsf{code}(\lhd u \rhd, \eta_0) \cdots \mathsf{code}(\lhd u \rhd, \eta_z)$ where $\eta_0, \ldots, \eta_z$ is the sequence of consecutive r-pebble configurations on $\lhd u \rhd$. Note that for $n = |\lhd u \rhd|$, we have $|\mathsf{pebb}_{k,\Sigma}(u)| = n^{k+1}$, and hence $|\mathsf{pebb}_{m,\Delta}(\mathsf{peb}_{k,\Sigma}(u))| = |\mathsf{pebb}_{r,\Sigma}(u)| = n^{(k+1)(m+1)}$.

We now give meaningful names to the $r = km + k + m = m(k+1) + k$ pebbles involved in $\mathsf{pebb}_{r,\Sigma}$. From 1 to r, they receive the names

$$\begin{aligned}
&\langle 1, 1\rangle, \ldots, \langle 1, k\rangle, \langle 1, k+1\rangle,\\
&\langle 2, 1\rangle, \ldots, \langle 2, k\rangle, \langle 2, k+1\rangle,\\
&\ldots\\
&\langle m, 1\rangle, \ldots, \langle m, k\rangle, \langle m, k+1\rangle,\\
&\langle m+1, 1\rangle, \ldots, \langle m+1, k\rangle.
\end{aligned}$$

The pebbles in the last row, $\langle m+1, 1\rangle, \ldots, \langle m+1, k\rangle$, correspond to the pebbles $1, \ldots, k$ that determine the k-pebble configurations $\pi_0, \ldots, \pi_s$ of $\mathsf{pebb}_{k,\Sigma}$ on $\lhd u \rhd$. The pebbles in the first m rows determine the m-pebble configurations $\rho_0, \ldots, \rho_t$ of $\mathsf{pebb}_{m,\Delta}$ on $\mathsf{pebb}_{k,\Sigma}(u)$. The pebbles in the jth row, $1 \leq j \leq m$, determine the position of pebble j on $\mathsf{pebb}_{k,\Sigma}(u)$. To be precise, the pebbles $\langle j, 1\rangle, \ldots, \langle j, k\rangle$ determine the k-pebble configuration π_ℓ such that pebble j occurs in the substring $\mathsf{code}(\lhd u \rhd, \pi_\ell)$ of $\mathsf{pebb}_{k,\Sigma}(u)$, and the pebble $\langle j, k+1\rangle$ determines the position of pebble j in that substring.

The transducer M is very simple: it computes a symbol-to-symbol string homomorphism from Γ to Ω. Walking from left to right through the input tape containing $\mathsf{pebb}_{r,\Sigma}(u)$, it changes each symbol (σ, b), with $\Sigma \cup \{\lhd, \rhd\}$ and $b \in \{0,1\}^r$, into the symbol $((\sigma, b_1), b_2)$ such that

- $b_1(i) = b(\langle m+1, i\rangle)$ for every $1 \leq i \leq k$, and
- for every $1 \leq j \leq m$, $b_2(j) = 1$ if and only if
  - $b(\langle j, i\rangle) = b(\langle m+1, i\rangle)$ for every $1 \leq i \leq k$, and
  - $b(\langle j, k+1\rangle) = 1$.

It should be clear, after some thinking, that M indeed translates each input string $\mathsf{pebb}_{r,\Sigma}(u)$ into $\mathsf{pebb}_{m,\Delta}(\mathsf{peb}_{k,\Sigma}(u))$. □

It is well known (and easy to see) that for every transduction $\tau \in \mathrm{DPT}_k$ and every string u in its domain, $|\tau(u)| = O(|u|^{k+1})$. In fact, if M is a deterministic k-pebble transducer computing τ, then the number of configurations of M on $\lhd u \rhd$ is $O(|u|^{k+1})$ and each configuration occurs at most once in the computation of M on input u. As observed in the proof of Theorem 11, $|\mathsf{pebb}_{m,\Delta}(\mathsf{peb}_{k,\Sigma}(u))| = (|u|+2)^{(k+1)(m+1)}$. Hence $\mathsf{peb}_{k,\Sigma} \circ \mathsf{pebb}_{m,\Delta}$ cannot be computed by a deterministic pebble transducer with less than $(k+1)(m+1) - 1$ pebbles. Since $\mathsf{peb}_{k,\Sigma} \in \mathrm{DPT}_k$ and $\mathsf{pebb}_{m,\Delta} \in \mathrm{DPT}_m$, this shows that Theorem 11 is optimal.

In the same way as Proposition 8 it can be shown that $\mathrm{DPT}_0 \circ \mathrm{PT}_0 \subseteq \mathrm{PT}_0$. From this it follows in the same way as above that $\mathrm{DPT}_k \circ \mathrm{PT}_m \subseteq \mathrm{PT}_{km+k+m}$ for every $k, m \in \mathbb{N}$.

By Theorem 11, $\mathrm{DPT} = \bigcup_{k \in \mathbb{N}} \mathrm{DPT}_k$ is closed under composition (as already proved in [15]). This is not true for $\mathrm{PT} = \bigcup_{k \in \mathbb{N}} \mathrm{PT}_k$.

Proposition 12 *PT is not closed under composition.*

Proof The proof is entirely similar to the proof of Lemma 15 of [12], as follows. Let τ be the transduction $\{(a^n, w\#w \mid n \geq 0,\ w \in \{a,b\}^*,\ |w| = n\}$. Obviously, $\tau \in \mathrm{PT}_0 \circ \mathrm{DPT}_0$ (first translate a^n into all strings $w \in \{a,b\}^*$ of length n, then translate w into $w\#w$).

Assume that τ is computed by a k-pebble transducer M with m states, that produces at most one output symbol at each computation step. The number of configurations of M on $\triangleleft a^n \triangleright$ is $m \cdot (n+2)^{k+1}$. Choose n such that $2^n > m \cdot (n+2)^{k+1}$. Consider the behavior of M on input a^n, and consider in particular the configuration of M when it has just produced the symbol # of the output string $w\#w$. Since there are 2^n output strings $w\#w$ for a^n, there exist two strings w_1 and w_2 for which this configuration is the same. Then the computation of output $w_1\#w_1$ can be switched halfway to the computation of $w_2\#w_2$, resulting in a computation of output $w_1\#w_2$ with $w_1 \neq w_2$. □

We finally show that the composition closure PT* of PT does not contain more partial functions than PT: all partial functions in PT* are already in DPT. We define $\mathrm{PT}^1 = \mathrm{PT}$, $\mathrm{PT}^{n+1} = \mathrm{PT}^n \circ \mathrm{PT}$, and $\mathrm{PT}^* = \bigcup_{n\geq 1} \mathrm{PT}^n$.

Theorem 13 *PT* has uniformizers in DPT.*

Proof Consider a transduction $\tau_1 \circ \cdots \circ \tau_n$ with $\tau_i \in \mathrm{PT}$ for $1 \leq i \leq n$. Since $\mathsf{dom}(\tau_i)$ is regular (see, for instance, [15]), we may assume that the range of τ_{i-1} is contained in the domain of τ_i for $2 \leq i \leq n$ (when computing τ_{i-1}, simulate a finite automaton for $\mathsf{dom}(\tau_i)$ on the output). By Theorem 6, τ_i has a uniformizer τ_i' in DPT. By the above assumption, $\tau_1' \circ \cdots \circ \tau_n'$ is a uniformizer of $\tau_1 \circ \cdots \circ \tau_n$, and $\tau_1' \circ \cdots \circ \tau_n'$ is in DPT by Theorem 11. □

6 Conclusion

By Proposition 12, $\mathrm{PT} \subsetneq \mathrm{PT}^2$. It is open whether $\mathrm{PT}^n \subsetneq \mathrm{PT}^{n+1}$ for every n.

A variant of pebble handling is to allow a pebble to be lifted also when the reading head is *not* at the square where the pebble was dropped. In the literature these are called "strong" pebbles, as opposed to the "weak" pebbles in this paper. It is shown in [2] that strong k-pebble tree automata have the same expressive power as weak ones. It is not difficult to show that deterministic strong 1-pebble transducers have the same power as weak ones, but apart from that it is open whether or not strong k-pebble transducers have more power than weak ones. It is also open whether the results of this paper are also valid for strong pebble transducers.

We finally mention that it is open whether the equivalence problem for deterministic k-pebble transducers is decidable. By an argument similar to the one in the proof of Theorem 4 of [15], it can be shown that every deterministic k-pebble transduction is a composition of k deterministic macro tree transductions. For an overview on the equivalence problem for deterministic macro tree transducers see [24].

References

1. Benedikt, M., Engelfriet, J., Maneth, S.: Determinacy and rewriting of top–down and MSO tree transformations. In: Chatterjee, K., Sgall, J. (Eds.) Proceedings of the MFCS 2013, Lecture Notes in Computer Science 8087, pp. 146–158. Springer, Berlin (2013)
2. Bojańczyk, M., Samuelides, M., Schwentick, T., Segoufin, L.: Expressive power of pebble automata. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (Eds.) Proceedings of the ICALP 2006, Lecture Notes in Computer Science 4051, pp. 157–168. Springer, Berlin (2006)
3. Bollig, B., Gastin, P., Monmege, B.: Weighted specifications over nested words. In: Pfenning, F. (Ed.) Proceedings of the FOSSACS 2013, Lecture Notes in Computer Science 7794, pp. 385–400. Springer, Berlin (2013)
4. Bollig, B., Gastin, P., Monmege, B., Zeitoun, M.: Pebble weighted automata and weighted logics. ACM Trans. Comput. Logic **15**(2) (2014), Article 15
5. Bollig, B., Gastin, P., Monmege, B., Zeitoun, M.: Logical characterization of weighted pebble walking automata. In: Proceedings of the CSL-LICS 2014. ACM Press (2014), Article 19
6. Chytil, M.P., Jákl, V.: Serial composition of 2-way finite-state transducers and simple programs on strings. In: Salomaa, A., Steinby, M. (Eds.) Proceedings of the ICALP 1977, Lecture Notes in Computer Science 52, pp. 135–147. Springer, Berlin (1977)
7. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic—A Language-Theoretic Approach. Cambridge University Press, Cambridge (2012)
8. Engelfriet, J.: On tree transducers for partial functions. Inform. Proc. Lett. **7**, 170–172 (1978)
9. Engelfriet, J.: Three hierarchies of transducers. Math. Syst. Theory **15**, 95–125 (1982)
10. Engelfriet, J.: Context-Free Grammars with Storage (1986/2014), CoRR abs/1408.0683
11. Engelfriet, J., Hoogeboom, H.J.: Tree-walking pebble automata. In: Karhumäki, J., Maurer, H., Paŭn, G., Rozenberg, G. (Eds.) Jewels are Forever, Contributions to Theoretical Computer Science in Honor of Arto Salomaa, pp. 72–83. Springer, Berlin (1999)
12. Engelfriet, J., Hoogeboom, H.J.: MSO definable string transductions and two-way finite-state transducers. ACM Trans. Comput. Logic **2**, 216–254 (2001)
13. Engelfriet, J., Hoogeboom, H.J.: Automata with nested pebbles capture first-order logic with transitive closure. Logical Methods Comput. Sci. **3**(2) (2007), Paper 3
14. Engelfriet, J., Hoogeboom, H.J., Samwel, B.: XML transformation by tree-walking transducers with invisible pebbles. In: Libkin, L. (Ed.) Proceedings of the 26th POD, pp. 63–72. ACM Press (2007)
15. Engelfriet, J., Maneth, S.: Two-way finite state transducers with nested pebbles. In: Diks, K., Rytter, W. (Eds.) Proceedings of the MFCS 2002, Lecture Notes in Computer Science 2420, pp. 234–244. Springer, Berlin (2002)
16. Engelfriet, J., Maneth, S.: A comparison of pebble tree transducers with macro tree transducers. Acta Informatica **39**, 613–698 (2003)
17. Fülöp, Z., Muzamel, L.: Pebble macro tree transducers with strong pebble handling. Fundamenta Informatica **89**, 207–257 (2008)
18. Fülöp, Z., Muzamel, L.: Circularity, composition, and decomposition results for pebble macro tree transducers. J. Automata Lang. Comb. **13**, 3–44 (2008)
19. Gastin, P., Monmege, B.: Adding pebbles to weighted automata: easy specification & efficient evaluation. Theor. Comput. Sci. **534**, 24–44 (2014)
20. Geffert, V., Ištoňová, Ľ.: Translation from classical two-way automata to pebble two-way automata. RAIRO-Theor. Inf. Appl. **44**, 507–523 (2010)
21. Globerman, N., Harel, D.: Complexity results for two-way and multi-pebble automata and their logics. Theor. Comput. Sci. **169**, 161–184 (1996)
22. Goris, E., Marx, M.: Looping caterpillars [semistructured data querying]. In: Proceedings of the LICS 2005, pp. 51–60, IEEE
23. Hennie, F.C.: One-tape, off-line Turing machine computations. Inf. Control **8**, 553–578 (1965)
24. Maneth, S.: Equivalence problems for tree transducers: a brief survey. In: Ésik, Z., Fülöp, Z. (Eds.) Proceedings of the AFL 2014, EPTCS, vol. 151, pp. 74–93 (2014)
25. Milo, T., Suciu, D., Vianu, V.: Typechecking for XML transformers. J. Comput. Syst. Sci. **66**, 66–97 (2003)
26. Muscholl, A., Samuelides, M., Segoufin, L.: Complementing deterministic tree-walking automata. Inform. Process. Lett. **99**, 33–39 (2006)
27. Muzamel, L.: Pebble alternating tree-walking automata and their recognizing power. Acta Cybernetica **18**, 427–450 (2008)
28. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Logic **5**, 403–435 (2004)

29. Pighizzini, G.: Two-way finite automata: old and recent results. In: Formenti, E. (Ed.) Proceedings of the AUTOMATA & JAC 2012, EPTCS, vol. 90, pp. 3–20 (2012)
30. Samuelides, M., Segoufin, L.: Complexity of pebble tree-walking automata. In: Csuhaj-Varjú, E., Ésik, Z. (Eds.) Proceedings of the FCT 2007, Lecture Notes in Computer Science 4639, pp. 458–469. Springer, Berlin (2007)
31. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM J. Res. Dev. **3**, 198–200 (1959)
32. Tan, T.: Graph reachability and pebble automata over infinite alphabets. ACM Trans. Comput. Logic **14**(3) (2013), Article 19