Reversible Pebble Transducers

Luc Dartois
Université Paris Est Creteil, LACL, F-94010 Créteil, France

Paul Gastin
Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, 91190, Gif-sur-Yvette, France
CNRS, ReLaX, IRL 2000, Siruseri, India

Loïc Germerie Guizouarn
Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France

Shankaranarayanan Krishna
Indian Institute of Technology Bombay, Mumbai, India

Abstract

Deterministic two-way transducers with pebbles (aka pebble transducers) capture the class of polyregular functions, which extend the string-to-string regular functions allowing polynomial growth instead of linear growth. One of the most fundamental operations on functions is composition, and (poly)regular functions can be realized as a composition of several simpler functions. In general, composition of deterministic two-way transducers incur a doubly exponential blow-up in the size of the inputs. A major improvement in this direction comes from the fundamental result of Dartois et al. [10] showing a polynomial construction for the composition of *reversible* two-way transducers.

A precise complexity analysis for existing composition techniques of pebble transducers is missing. But they rely on the classic composition of two-way transducers and inherit the double exponential complexity. To overcome this problem, we introduce *reversible* pebble transducers. Our main results are efficient uniformization techniques for non-deterministic pebble transducers to reversible ones and efficient composition for reversible pebble transducers.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases Transducers, Polyregular functions, Reversibility, Composition, Uniformization

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.14

Related Version *Full Version*: `https://arxiv.org/abs/2506.11334`

1 Introduction

The theory of string transformations has garnered a lot of attention in recent years. The simplest kind of such transformations are *sequential functions* which are captured by deterministic finite automata whose transitions are labelled by output words. An example of a sequential function is $f(w) = w'$ where w, w' are words over an alphabet $\{a, b\}$, and w' is obtained from w by replacing in w, each a with bb, and each b with a. *Rational functions* strictly extend sequential functions by allowing the underlying automaton to be *unambiguous* instead of being deterministic. For instance, $f(ua) = au$ where $a \in \Sigma, u \in \Sigma^+$ which moves the last symbol of the input to the first position is a rational function, and not realizable by a sequential one.

While both sequential and rational functions are captured by "one-way" transducers, *regular functions* are realized by two-way transducers where the input is scanned in both directions. Regular functions strictly extend rational functions; for instance the regular function $f(w) = w^R$ where w^R is the reverse of w is not realizable as a rational function. Regular functions are also equivalent to Courcelle's word-to-word MSO transductions [9, 14], streaming string transducers [1] as well as combinator expressions [2, 3, 6, 12].

Sequential, rational as well as regular functions are all transformations of *linear growth*, where the sizes of the outputs are linear in the sizes of the inputs. More recently, [5] revisits functions of *polynomial growth*, that is, those whose output sizes are polynomial in the sizes of the input. These functions, aptly called *polyregular functions*, date back to [13, 15]. A logical characterization for polyregular functions was given in [7], namely, word-to-word MSO interpretations. MSO interpretations are equivalent to polyregular functions which are word-to-word functions recognised by deterministic, two-way pebble transducers [4].

Prior work [13, 4] on pebble transducers show that they are closed under composition, and are equivalent to polyregular functions. [13] of pebble transducers, as well as a *uniformization* of (non-determinsitic) pebble to deterministic ones with the same number of pebbles. The pebble transducers of [4] allow *comparison tests*, namely, one can compare the positions on which pebbles are dropped, or the position of the head and of a pebble. [4] gives high level ideas for the closure under composition for pebble transducers with comparison tests.

Our focus in this paper is on the complexity aspects of the composition and uniformization of pebble transducers. A practical setting where composition is useful is while synthesizing pebble transducers from polyregular functions [7]. Polyregular functions [5] are defined as the smallest class of functions closed under composition that contains sequential functions, the squaring function as well as the iterated reverse. In general, the composition of (pebble-less) two-way transducers has at least a doubly exponential blowup in the size of the input transducers [8]. An important class of two-way transducers for which composition is polynomial are *reversible* transducers [10]. Reversible transducers are those which are both deterministic and reverse-deterministic; moreover they are expressively equivalent to two-way transducers [10]. This makes reversibility a very attractive property for a transducer. However, the notion of reversible pebble transducers has not been considered yet. Therefore it is unknown if they have the same expressiveness as deterministic pebble transducers (polyregular functions), and whether they are also amenable to an efficient composition like their pebble-less counterparts. Our paper fills this gap.

Our Contributions

1. **Reversible Pebble Transducers**. We define reversible pebble transducers with equality tests allowing to check whether two pebbles are dropped on the same position, in addition to the basic tests allowing to check if a pebble is dropped on the current head position.
2. **Equivalent notions for pebble transducers**. We show that pebble transducers with basic tests only, as well as those extended with equality tests are equivalent. More precisely, given a k-pebble transducer with equality tests having n states, we show that we can construct an equivalent k-pebble transducer with basic tests having $\mathcal{O}(nk^2)$ states. The construction preserves both determinism and reverse-determinism.
3. **Composition of Reversible Pebble Transducers**. We show that reversible pebble transducers are closed under composition. Given points 1,2 above, we allow equality tests while studying composition. To be precise, given reversible pebble transducers $\mathcal{T}$ and $\mathcal{T}'$ with states Q, Q', having $n, m \geq 0$ pebbles, we construct a reversible pebble transducer $\mathcal{T}''$ with $nm + n + m$ pebbles and (i) $2 \cdot |Q| \cdot |Q'|$ states when $m = 0$, and (ii) $\mathcal{O}(|Q|^{m+2} \cdot |Q'| \cdot (n+1)^{m+3})$ states when $m > 0$.
   To compare with the existing composition approaches [13] and [4], the number of pebbles we require for composition matches with [13] who showed that this is the optimal number of pebbles needed. Regarding the number of states, [13] relies on the composition of deterministic two-way transducers, which, as mentioned before, incurs at least a doubly

exponential blowup. As can be seen from (ii) above, reversibility gives a much better complexity. The high level description for composition in [4] does not give details on the number of states, and incurs n extra pebbles. As such, we do not compare with them.

4. **Uniformization of non-deterministic Reversible Pebble Transducers**. Given a non-deterministic pebble transducer $\mathcal{T}$ with k pebbles and n states, we construct a reversible pebble transducer $\mathcal{T}'$ with $2^{\mathcal{O}((kn)^2)}$ states which uniformizes the relation $[\![\mathcal{T}]\!]$ computed by $\mathcal{T}$, i.e., computes a function $[\![\mathcal{T}']\!] \subseteq [\![\mathcal{T}]\!]$ with same domain.
   The uniformization [13] of non-deterministic to deterministic pebble transducers is done using results from [14], where uniformization is done via a sequence of tranformations, namely: (i) transductions computed by a 0-pebble transducer are definable using non-determinsitic MSO transductions, which can be uniformized using deterministic MSO transductions, (ii) deterministic MSO transductions are computed by deterministic two-way transducers. The uniformization for a given k-pebble transducer [13] is then computed as a composition of a k-counting transduction and the deterministic two-way transducer obtained in (ii). Due to (ii), this may incur a non-elementary complexity, while our uniformization technique only has an exponential complexity.
5. **Polyregular functions as Reversible Pebble Transducers**. Polyregular functions can be realized as (a) deterministic two-way pebble transducers [4] or, (b) as a composition of sequential functions, squaring and iterated reverse functions [5]. This results in two possibilities to compute a reversible pebble transducer for a polyregular function.
   a. If the polyregular function is presented as a deterministic two-way pebble transducers then we may use the uniformization construction of Item 4 to obtain an equivalent reversible pebble transducer.
   b. If a polyregular function f is given as a modular expression involving sequential, squaring and iterated reverse functions, then one can directly synthesize a reversible pebble transducer computing f using (i) the reversible transducers for squaring and iterated reverse given in Figures 1 and 5 (ii) reversible transducers for the sequential functions obtained using [10, Theorem 2], and (iii) our composition of the resultant reversible pebble transducers. Note that the reversible machines in Figures 1 and 5 are small with 6/5 states and 1/0 pebbles respectively. Once again, this is better than constructing first from the modular expression a deterministic pebble transducer for function f and then uniformizing it to get a reversible one.

2 Reversible Pebble Transducers

Pebble Transducers with Equality Tests

In this section, we consider pebble transducers enhanced with equality tests between pebble positions, in addition to the classical query checking only if a pebble is present/absent at the head position. The equality tests are inspired from [5], where $\leq$-comparisons between any two pebble positions, as well as between the head and any pebble are allowed. Enhancing the basic tests, on whether a pebble is present/absent at the head, by allowing to check for any pair of pebbles whether or not they are at the same position is useful, especially when composing two pebble transducers.

A k-pebble automaton with equality tests (k-PA$_=$) is given by a tuple $\mathcal{A} = (Q, \Sigma, \delta, k, q_i, q_f)$ where Q is a finite set of states, Σ is the finite input alphabet, δ is the transition relation (see below), and $q_i, q_f \in Q$ respectively are initial and final states.

Given an input word $u \in \Sigma^*$, we define *positions* $\mathsf{pos}(u) = \{1, \ldots, |u|\}$. We visualise the input of a PA as a circular word $\#u$ where $\# \notin \Sigma$ is an endmarker reached when moving right (resp. left) from the last (resp. first) position of u. Keeping this in mind, we work with an *extended* set of positions $\mathsf{epos}(u) = \{0, 1, \ldots, |u|\}$. For instance, $\begin{smallmatrix} \# & a & b & c & d \\ 0 & 1 & 2 & 3 & 4 \end{smallmatrix}$.

The head of the automaton during a run over u, occupies a position from $\mathsf{epos}(u)$, denoted h. The states Q of the pebble automaton are partitioned into 3 disjoint sets, $Q_{+1} \uplus Q_0 \uplus Q_{-1}$ signifying movement of the head to the right, no movement of the head and movement of the head to the left, respectively. Given $\mathsf{h} \in \mathsf{epos}(u)$ and a state $q \in Q$, we define $\mathsf{h} + q = \mathsf{h}$ if $q \in Q_0$; $\mathsf{h} + q = (\mathsf{h} + 1 \mod (|u| + 1))$ if $q \in Q_{+1}$; and $\mathsf{h} + q = (\mathsf{h} - 1 \mod (|u| + 1))$ if $q \in Q_{-1}$. The intuition is that if the head moves to the right of the last position of the word, then it reads $\#$, and if the head moves to the left from $\#$, it reads the last position of u.

Likewise, the pebbles are dropped on $\mathsf{epos}(u)$ and follow a stack policy. We denote by $\mathsf{peb} \in \mathsf{epos}(u)^{\leq k}$ the stack of positions where pebbles are dropped.

A configuration of a k-pebble automaton on the input word u is a tuple $C = (q, \mathsf{peb}, \mathsf{h}) \in Q \times \mathsf{epos}(u)^{\leq k} \times \mathsf{epos}(u)$. In this configuration, the head of the automaton reads position h, there are $|\mathsf{peb}|$ pebbles dropped and the i-th pebble ($1 \leq i \leq |\mathsf{peb}|$) is dropped on position $\mathsf{peb}_i \in \mathsf{epos}(u)$. The initial configuration is $(q_i, \varepsilon, 0)$ and the accepting (final) configuration is $(q_f, \varepsilon, 0)$. Hence, in order to accept, the automaton should lift all pebbles and move back to the endmarker. Wlog, we assume that $q_i, q_f \in Q_0$ with $q_i \neq q_f$, and there are no transitions starting from q_f or going to q_i.

Transitions are of the form $t = (q, a, \varphi, \mathsf{op}, q')$ with states $q, q' \in Q$, input letter $a \in \Sigma_\# = \Sigma \cup \{\#\}$, test φ which is a conjunction of atoms of the form $(\mathsf{h} = \mathsf{p}_i)$, $\neg(\mathsf{h} = \mathsf{p}_i)$, $(\mathsf{p}_i = \mathsf{p}_j)$ or $\neg(\mathsf{p}_i = \mathsf{p}_j)$ (with $1 \leq i, j \leq k$), and operation op of the form[1] drop_i, lift_i or nop (with $1 \leq i \leq k$). Transition t is *enabled* at configuration $C = (q, \mathsf{peb}, \mathsf{h})$ (denoted $C \models t$) if:

1. the letter read is a, i.e., $a = (\#u)_{\mathsf{h}}$
2. $\mathsf{peb}, \mathsf{h} \models \varphi$: the test succeeds, where
   - $\mathsf{peb}, \mathsf{h} \models (\mathsf{h} = \mathsf{p}_i)$ if $1 \leq i \leq |\mathsf{peb}|$ and $\mathsf{peb}_i = \mathsf{h}$,
   - $\mathsf{peb}, \mathsf{h} \models (\mathsf{p}_i = \mathsf{p}_j)$ if $1 \leq i, j \leq |\mathsf{peb}|$ and $\mathsf{peb}_i = \mathsf{peb}_j$,
3. $\mathsf{peb}, \mathsf{h} \models \mathsf{op}$: the operation can be executed where
   - $\mathsf{peb}, \mathsf{h} \models \mathsf{nop}$ is always true, and we let $\mathsf{nop}(\mathsf{peb}, \mathsf{h}) = \mathsf{peb}$,
   - $\mathsf{peb}, \mathsf{h} \models \mathsf{drop}_i$ if $|\mathsf{peb}| = i - 1$, and we let $\mathsf{drop}_i(\mathsf{peb}, \mathsf{h}) = \mathsf{peb} \cdot \mathsf{h}$,
   - $\mathsf{peb}, \mathsf{h} \models \mathsf{lift}_i$ if $|\mathsf{peb}| = i$ and $\mathsf{h} = \mathsf{peb}_i$, and we let $\mathsf{lift}_i(\mathsf{peb}, \mathsf{h}) = \mathsf{peb}_1 \cdots \mathsf{peb}_{i-1}$.

When $C \models t$ we have $C \xrightarrow{t} C' = (q', \mathsf{peb}', \mathsf{h}')$ where $\mathsf{peb}' = \mathsf{op}(\mathsf{peb}, \mathsf{h})$ and $\mathsf{h}' = \mathsf{h} + q'$.

Notice that the top pebble may be lifted from the pebble stack only when it is on the position being read. We often simply write p_i (resp. $\neg\mathsf{p}_i$) for the atomic test $(\mathsf{h} = \mathsf{p}_i)$ (resp. $\neg(\mathsf{h} = \mathsf{p}_i)$). Note that, if a transition $t = (q, a, \mathsf{p}_i, \mathsf{drop}_j, q')$ is enabled at some configuration C, then $i < j$ and $C' \models (\mathsf{p}_i = \mathsf{p}_j)$ where $C \xrightarrow{t} C'$.

An equality test $\mathsf{p}_i = \mathsf{p}_i$ allows to check if the size of the pebble stack is at least i. This is not possible if we only use atoms of the form p_i or $\neg\mathsf{p}_i$ (which is not an expressivity problem since one may always store the size of the pebble stack in the state of the automaton).

A run of $\mathcal{A}$ is a sequence $C_0 \xrightarrow{t_1} C_1 \cdots \xrightarrow{t_n} C_n$ where C_ℓ are configurations on the input word u and $t_\ell \in \delta$ are transitions of $\mathcal{A}$. The run is initial if $C_0 = (q_i, \varepsilon, 0)$ and final if $C_n = (q_f, \varepsilon, 0)$. It is accepting if it is both initial and final.

[1] We chose to specify with index i of drop and lift which pebble is dropped or lifted. This allows to get determinism or reverse-determinism without adding extra tests to transitions. We could have simply used drop and lift at the expense of adding tests to transitions to ensure (reverse-) determinism.

The automaton is *deterministic* if for all pairs of transitions $t = (q, a, \varphi, \mathsf{op}, r)$ and $t' = (q, a, \varphi', \mathsf{op}', r')$, if t and t' may be simultaneously enabled (at some configuration C) then $t = t'$. The fact that an operation op is enabled may be written as a test $\overline{\mathsf{op}}$: $\overline{\mathsf{nop}} = \mathsf{true}$, $\overline{\mathsf{drop}_i} = (\mathsf{p}_{i-1} = \mathsf{p}_{i-1}) \wedge \neg(\mathsf{p}_i = \mathsf{p}_i)$ (with $\mathsf{p}_0 = \mathsf{p}_0$ identified with true), and $\overline{\mathsf{lift}_i} = (\mathsf{h} = \mathsf{p}_i) \wedge \neg(\mathsf{p}_{i+1} = \mathsf{p}_{i+1})$. Then, the transitions t and t' may be simultaneously enabled if and only if the test $\varphi \wedge \overline{\mathsf{op}} \wedge \varphi' \wedge \overline{\mathsf{op}'}$ is satisfiable. This gives a more syntactic definition of determinism. In particular, a transtion t with operation lift_i and a transition t' with atomic test $\neg\mathsf{p}_i$ cannot be simultaneously enabled.

We define the reverse op^r of an operation op by $\mathsf{nop}^r = \mathsf{nop}$, $\mathsf{drop}_i^r = \mathsf{lift}_i$ and $\mathsf{lift}_i^r = \mathsf{drop}_i$. Notice that if $\mathsf{peb}, \mathsf{h} \models \mathsf{op}$ then $\mathsf{op}(\mathsf{peb}, \mathsf{h}), \mathsf{h} \models \mathsf{op}^r$ and $\mathsf{peb} = \mathsf{op}^r(\mathsf{op}(\mathsf{peb}, \mathsf{h}), \mathsf{h})$.

A transition $t = (q, a, \varphi, \mathsf{op}, q')$ is *reverse-enabled* at configuration $C' = (q', \mathsf{peb}', \mathsf{h}')$ (denoted $C' \models_{\mathsf{rev}} t$) if there is a configuration $C = (q, \mathsf{peb}, \mathsf{h})$ enabling t such that $C \xrightarrow{t} C'$. More explicitly, the only possibility for configuration C is given by $\mathsf{h} = \mathsf{h}' - q'$ (corresponding to $\mathsf{h}' = \mathsf{h} + q'$), and $\mathsf{peb} = \mathsf{op}^r(\mathsf{peb}', \mathsf{h})$ (corresponding to $\mathsf{peb}' = \mathsf{op}(\mathsf{peb}, \mathsf{h})$). Then, $C' \models_{\mathsf{rev}} t$ iff $\mathsf{peb}', \mathsf{h}' - q' \models \mathsf{op}^r$ and $\mathsf{op}^r(\mathsf{peb}', \mathsf{h}' - q'), \mathsf{h}' - q' \models \varphi$. In this case, we write $C = t^{-1}(C') = (q, \mathsf{op}^r(\mathsf{peb}', \mathsf{h}' - q'), \mathsf{h}' - q')$. In particular, a transtion t with operation drop_i and a transition t' with atomic test $\neg\mathsf{p}_i$ and operation nop cannot be simultaneously reverse-enabled.

The automaton is *reverse-deterministic* if for all pairs of transitions $t = (q, a, \varphi, \mathsf{op}, r)$ and $t' = (q', a, \varphi', \mathsf{op}', r)$, if t and t' may be simultaneously reverse-enabled (at some configuration C') then $t = t'$.

The pebble automaton is *reversible* if it is both *deterministic* and *reverse-deterministic*.

A k-pebble transducer with equality tests (k-$\mathsf{PT}_=$) $\mathcal{T} = (Q, \Sigma, \delta, k, q_i, q_f, \Gamma, \mu)$ is a k-pebble automaton (k-$\mathsf{PA}_=$) $\mathcal{A} = (Q, \Sigma, \delta, k, q_i, q_f)$ extended with an output alphabet Γ and an output function μ mapping each transition of $\mathcal{A}$ to a word in Γ^*. The semantics of $\mathcal{T}$ is a relation $[\![\mathcal{T}]\!] \subseteq \Sigma^* \times \Gamma^*$ consisting of all pairs (u, v) such that there is an accepting run $C_0 \xrightarrow{t_1} C_1 \cdots \xrightarrow{t_n} C_n$ on $\#u$ with $v = \mu(t_1) \cdots \mu(t_n)$.

A pebble transducer is deterministic (reverse-deterministic, reversible) whenever the underlying pebble automaton is. The semantics of a deterministic (reverse-deterministic, reversible) pebble transducer is a partial function.

▶ Remark 2.1. Even though the semantics of 0-pebble transducers is different to that of two-way transducers as defined in [10], the two models are equivalent. Given a transducer with n states using one semantics, one can build a transducer realising the same relation using the other semantics, with $\mathcal{O}(n)$ states. Moreover, determinism and reverse-determinism are preserved by this translation (see the extended version [11]).

Basic Pebble Transducers

A (basic) pebble automaton (k-PA) or transducer (k-PT) is one in which equality tests of the form $(\mathsf{p}_i = \mathsf{p}_j)$ or $\neg(\mathsf{p}_i = \mathsf{p}_j)$ are not used. In some cases, we use a bit vector $\overline{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$ to denote a full test φ which is a conjunction over all $1 \leq i \leq k$ of atoms p_i if $b_i = 1$ and $\neg\mathsf{p}_i$ if $b_i = 0$.

▶ **Example 2.2.** Figure 1 depicts a reversible 1-pebble transducer computing the squaring function. This function outputs the concatenation of as many copies of the input word as there are letters in it. For the i^{th} copy, the i^{th} letter is marked. The blue part of the transducer copies its input and marks the positions on which the unique pebble is dropped. The black and red part initially drops the pebble on the first position, then lifts the pebble and drops it on the next position at each iteration. It moves to state q_2 and ends when the pebble is lifted from the last position. Without looking at the pebble operations, the transducer is

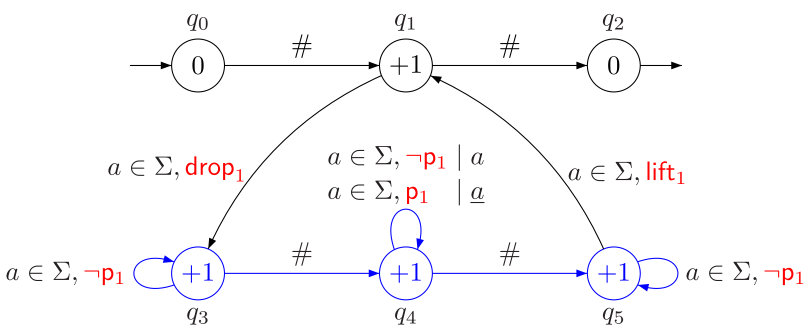

Figure 1 Reversible 1-pebble transducer for the function *squaring*. The labels $i \in \{0, +1, -1\}$ inside the states represent that they lie in Q_i. To keep the figure light, ε-outputs of transitions are omitted. The same transducer with state q_3 in Q_{-1} would also compute the squaring function.

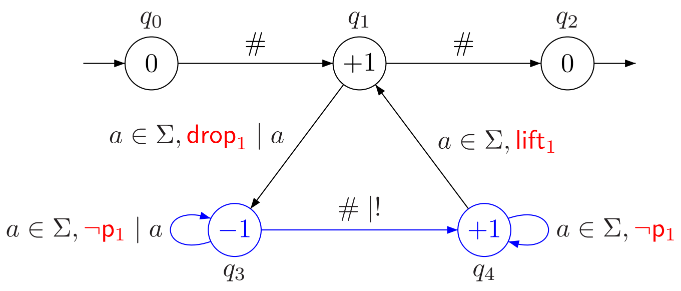

Figure 2 Reversible 1-pebble transducer for the function *all prefixes in reverse*.

deterministic (resp. reverse-deterministic) at every state except state q_5 (resp. q_3). However, since the lift_1 operation can only be triggered at the position on which the pebble 1 is dropped, the pebble guards lift_1 and $\neg \mathsf{p}_1$ cannot be simultaneously enabled, and thus the transducer is deterministic at q_5. Symmetrically, the reverse of a drop_1 operation can only be enabled on the position marked by p_1, and thus drop_1 and $\neg \mathsf{p}_1$ cannot be simultaneously reverse-enabled, and then the transducer is reverse-deterministic at q_3, proving that we have a reversible transducer.

▶ **Example 2.3.** Figure 2 represents a reversible 1-pebble transducer for the function which maps a word to the concatenation of the reverse of all its prefixes, with a ! symbol between each one. A run of this transducer on the word abb produces the word $a!ba!bba!$.
Using the same arguments as in the previous example, one can show that the transducer is reversible.

3 Simulating equality tests of pebbles

In this section, we show the equivalence between a basic k pebble transducer (k-PT) and a k pebble transducer allowing equality tests (k-PT$_=$). The precise statement is given by Theorem 3.1, followed by a proof sketch, the full proof can be found in [11].

▶ **Theorem 3.1.** *Given a k-pebble transducer with equality tests $\mathcal{A}$ having n states, one can construct a basic k-pebble transducer $\mathcal{B}$ with $n2^{k^2}$ states such that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.*

Moreover, if $\mathcal{A}$ is deterministic (reverse-deterministic, reversible) then so is $\mathcal{B}$.

Sketch of proof. The main idea while simulating $\mathcal{A}$ is to keep track of which pairs of pebbles are on the same position in a coherent manner: that is, if $p_i = p_j$ and $p_j = p_\ell$ then we also have $p_i = p_\ell$. While constructing $\mathcal{B}$, we store the equalities of the positions of the k pebbles as a $k \times k$ boolean matrix M where $M_{i,i} = 1$ represents that the i^{th} pebble has been dropped, and $M_{i,j} = 1$ if the i^{th} and j^{th} pebbles are on the same position. Then if we denote by Q the set of states of $\mathcal{A}$, the set of states of $\mathcal{B}$ is $Q' = Q \times \{0,1\}^{k^2}$.

We now explain how the information is used and updated. Initially, no pebble is dropped, and the initial state of $\mathcal{B}$ is $(q_i, \overline{0})$ where $\overline{0}$ denotes the $k \times k$ zero matrix and q_i is the initial state of $\mathcal{A}$. Similarly, a run is accepting only when all pebbles are lifted. Thus the final state of $\mathcal{B}$ is $(q_f, \overline{0})$ where q_f is the final state of $\mathcal{A}$. Recall that basic pebble transducers can see whether a given pebble is at the same position as the reading head, thus transitions of $\mathcal{A}$ involving a guard of the form $\mathsf{h} = \mathsf{p}_i$ or $\neg(\mathsf{h} = \mathsf{p}_i)$ can be kept in $\mathcal{B}$. Tests of the form $\mathsf{p}_i = \mathsf{p}_j$ are replaced by true if $M_{i,j} = 1$ and by false otherwise. On transitions where there is no pebble lifting or dropping, the matrix M does not need to be updated. The i^{th} pebble p_i can be dropped only if the $(i-1)^{th}$ pebble is already dropped (that is, $M_{i-1,i-1} = 1$) and the i^{th} one is not ($M_{i,i} = 0$). On dropping p_i, the matrix M is updated by setting all coefficients $M_{i,j}$ and $M_{j,i}$ to 1 if $j = i$ or ($j < i$ and $\mathsf{h} = p_j$), and 0 otherwise. Note that to do this, each transition of $\mathcal{A}$ with a drop_i action is duplicated into 2^{i-1} disjoint transitions testing which subsets of $\mathsf{p}_1, \ldots, \mathsf{p}_{i-1}$ are present at the head. Only one transition can then be fired, depending on the bit vector generated by the pebbles at the current position. Similarly, the i^{th} pebble p_i can be lifted if and only if $\mathsf{h} = \mathsf{p}_i$ and no larger pebbles are dropped ($i = k$ or $M_{i+1,i+1} = 0$). On lifting p_i, the matrix M is updated by setting all coefficients of the i^{th} row and column to 0. However to ensure the preservation of reverse-determinism, each lift_i transition is duplicated into 2^{i-1} transitions that have complete and disjoint tests for the current bit vector over pebbles $\mathsf{p}_1, \ldots, \mathsf{p}_{i-1}$.

Finally, suppose that $\mathcal{A}$ is reversible. The only extra transitions added to $\mathcal{B}$ are when a pebble is dropped or lifted. However, all the duplicate transitions that have been added in $\mathcal{B}$ have disjoint tests (enforced by disjoint bit vectors). Thus $\mathcal{B}$ is deterministic since $\mathcal{A}$ is. On the other hand, consider we want to reverse a lift_i transition in $\mathcal{A}$. This means that the i^{th} pebble was at the previous position of the reading head, and further, there is only one bit vector (enriching the ith row and column of M) which was valid at the previous step. The correct predecessor matrix is then the only one where the i^{th} row and column is coherent with the bit vector of the previous position. ◀

4 Composition of pebble transducers

The goal of this section is to give an efficient (in the size of the resulting transducer) construction for the composition of pebble transducers. Since the construction is rather involved, we start with two simpler cases. The first one described in Section 4.1 shows how to compose a reversible n-pebble transducer with the reverse function $\mathsf{rev}\colon \Gamma^* \to \Gamma^*$ which takes a string of letters and outputs the sequence of letters in reverse order. For instance, $\mathsf{rev}(abac) = caba$. The second simpler case is the composition of a reversible n-pebble transducer with a deterministic 0-pebble transducer. This case is addressed in Section 4.2. Finally, the general case is solved in Section 4.3.

4.1 Reversing the output of a reversible pebble transducer

Let $\mathcal{T} = (Q, \Sigma, \delta, n, q_i, q_f, \Gamma, \mu)$ be a reversible n-pebble transducer, possibly with equality tests. The goal is to construct a reversible transducer $\mathcal{T}^r = (Q', \Sigma, \delta', n, q'_i, q'_f, \Gamma, \mu')$ which outputs the reverse of the string produced by $\mathcal{T}$: $\mathsf{dom}(\mathcal{T}^r) = \mathsf{dom}(\mathcal{T})$ and for all $u \in \mathsf{dom}(\mathcal{T})$, the string $[\![\mathcal{T}^r]\!](u)$ is the reverse of the string $[\![\mathcal{T}]\!](u)$.

The set of states of $\mathcal{T}^r$ is $Q' = Q$ but the polarity of the states is reversed: $Q'_{+1} = Q_{-1}$, $Q'_0 = Q_0$ and $Q'_{-1} = Q_{+1}$. The initial and final states are exchanged: $q'_i = q_f$ and $q'_f = q_i$. The transitions are also reversed in the following way. Let $t = (q, a, \varphi, \mathsf{op}, q')$ be a transition of $\mathcal{T}$. Then, $t^r = (q', a, (\varphi, \mathsf{op})^r, q)$ is a transition of $\mathcal{T}^r$ with $\mu'(t^r) = \mu(t)$. We define $(\varphi, \mathsf{op})^r = \mathsf{op}(\varphi), \mathsf{op}^r$: the operation is simply op^r but the test $\mathsf{op}(\varphi)$ depends on both φ and op. It remains to define $\mathsf{op}(\varphi)$. We want that $\mathsf{peb}, \mathsf{h} \models \varphi$ iff $\mathsf{op}(\mathsf{peb}, \mathsf{h}), \mathsf{h} \models \mathsf{op}(\varphi)$, assuming that $\mathsf{op}(\mathsf{peb}, \mathsf{h})$ is defined. We let $\mathsf{op}(\varphi_1 \wedge \varphi_2) = \mathsf{op}(\varphi_1) \wedge \mathsf{op}(\varphi_2)$ and $\mathsf{op}(\neg\varphi) = \neg\mathsf{op}(\varphi)$. We let $\mathsf{nop}(\varphi) = \varphi$, and the remaining cases are given below

$$\mathsf{drop}_\ell(\mathsf{h} = \mathsf{p}_i) = \begin{cases} (\mathsf{h} = \mathsf{p}_i) & \text{if } i < \ell \\ \mathsf{false} & \text{otherwise} \end{cases} \qquad \mathsf{drop}_\ell(\mathsf{p}_i = \mathsf{p}_j) = \begin{cases} (\mathsf{p}_i = \mathsf{p}_j) & \text{if } i, j < \ell \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\mathsf{lift}_\ell(\mathsf{h} = \mathsf{p}_i) = \begin{cases} (\mathsf{h} = \mathsf{p}_i) & \text{if } i < \ell \\ \mathsf{true} & \text{if } i = \ell \\ \mathsf{false} & \text{otherwise} \end{cases} \qquad \mathsf{lift}_\ell(\mathsf{p}_i = \mathsf{p}_j) = \begin{cases} (\mathsf{p}_i = \mathsf{p}_j) & \text{if } i, j < \ell \\ (\mathsf{h} = \mathsf{p}_i) & \text{if } i < j = \ell \\ (\mathsf{h} = \mathsf{p}_j) & \text{if } j < i = \ell \\ \mathsf{true} & \text{if } i = j = \ell \\ \mathsf{false} & \text{otherwise} \end{cases}$$

We can easily check that $\mathsf{peb}, \mathsf{h} \models \varphi$ iff $\mathsf{op}(\mathsf{peb}, \mathsf{h}), \mathsf{h} \models \mathsf{op}(\varphi)$ when $\mathsf{op}(\mathsf{peb}, \mathsf{h})$ is defined. Recall that, when $\mathsf{peb}' = \mathsf{op}(\mathsf{peb}, \mathsf{h})$ is defined, then $\mathsf{op}^r(\mathsf{peb}', \mathsf{h})$ is defined and we have $\mathsf{peb} = \mathsf{op}^r(\mathsf{peb}', \mathsf{h})$. With $t = (q, a, \varphi, \mathsf{op}, q')$ and $t^r = (q', a, \mathsf{op}(\varphi), \mathsf{op}^r, q)$, we deduce that

$$(q, \mathsf{peb}, \mathsf{h}) \xrightarrow{t} (q', \mathsf{peb}', \mathsf{h}') \quad \text{iff} \quad (q', \mathsf{peb}', \mathsf{h}' - q') \xrightarrow{t^r} (q, \mathsf{peb}, \mathsf{h} - q) \tag{1}$$

We deduce that, for $u \in \Sigma^*$, the following sequence is a run of $\mathcal{T}$ on $\#u$

$$(q_1, \mathsf{peb}^1, \mathsf{h}_1) \xrightarrow{t_1} (q_2, \mathsf{peb}^2, \mathsf{h}_2) \xrightarrow{t_2} (q_3, \mathsf{peb}^3, \mathsf{h}_3) \cdots \xrightarrow{t_{m-1}} (q_m, \mathsf{peb}^m, \mathsf{h}_m)$$

if and only if the following sequence is a run of $\mathcal{T}^r$ on $\#u$

$$(q_1, \mathsf{peb}^1, \mathsf{h}_1 - q_1) \xleftarrow{t_1^r} (q_2, \mathsf{peb}^2, \mathsf{h}_2 - q_2) \xleftarrow{t_2^r} (q_3, \mathsf{peb}^3, \mathsf{h}_3 - q_3) \cdots \xleftarrow{t_{m-1}^r} (q_m, \mathsf{peb}^m, \mathsf{h}_m - q_m)$$

Now, the run of $\mathcal{T}$ is accepting iff $(q_1, \mathsf{peb}^1, \mathsf{h}_1) = (q_i, \varepsilon, 0)$ and $(q_m, \mathsf{peb}^m, \mathsf{h}_m) = (q_f, \varepsilon, 0)$. Since $q_i, q_f \in Q_0$, we deduce that the run of $\mathcal{T}$ is accepting if and only if the corresponding run of $\mathcal{T}^r$ is accepting, i.e., $(q_m, \mathsf{peb}^m, \mathsf{h}_m - q_m) = (q'_i, \varepsilon, 0) = (q_f, \varepsilon, 0)$ and $(q_1, \mathsf{peb}^1, \mathsf{h}_1 - q_1) = (q'_f, \varepsilon, 0) = (q_i, \varepsilon, 0)$.

We deduce that $\mathsf{dom}(\mathcal{T}^r) = \mathsf{dom}(\mathcal{T})$ and, by definition of μ', for $u \in \mathsf{dom}(\mathcal{T})$, the string $[\![\mathcal{T}^r]\!](u)$ is the reverse of the string $[\![\mathcal{T}]\!](u)$.

To show that $\mathcal{T}^r$ is deterministic, consider a pair of transitions $t_1^r = (q', a, (\varphi_1, \mathsf{op}_1)^r, q_1)$ and $t_2^r = (q', a, (\varphi_2, \mathsf{op}_2)^r, q_2)$ where $t_1 = (q_1, a, \varphi_1, \mathsf{op}_1, q')$ and $t_2 = (q_2, a, \varphi_2, \mathsf{op}_2, q')$ are transitions of $\mathcal{T}$. Assume that t_1^r is enabled at some configuration $C' = (q', \mathsf{peb}', \mathsf{h}')$. Using (1), we deduce that t_1 is reverse-enabled at configuration $C = (q', \mathsf{peb}', \mathsf{h}' + q')$. Similarly, if t_2^r is enabled at C' then t_2 is reverse-enabled at C. Since $\mathcal{T}$ is reverse-deterministic, we deduce that $\mathcal{T}^r$ is deterministic.

We can prove similarly that $\mathcal{T}^r$ is reverse-deterministic using the fact that $\mathcal{T}$ is determinisitic. In particular, we show using (1) that transition t^r is reversed-enabled at some configuration $C' = (q, \mathsf{peb}, \mathsf{h})$ if and only if transition t is enabled at configuration $C = (q, \mathsf{peb}, \mathsf{h} + q)$. We deduce that $\mathcal{T}^r$ is reversible.

4.2 Composition: Simple case

In this subsection, we prove the following result.

▶ **Theorem 4.1.** *Let $\mathcal{T} = (Q, \Sigma, \delta, n, q_i, q_f, \Gamma, \mu)$ be a* reversible *n-pebble transducer (possibly with equality tests) computing a function $f : \Sigma^* \to \Gamma^*$. Let $\mathcal{T}' = (Q', \Gamma, \delta', 0, q'_i, q'_f, \Delta, \mu')$ be a* deterministic *0-pebble transducer computing a function $g : \Gamma^* \to \Delta^*$. We can construct a* deterministic *n-pebble transducer $\mathcal{T}''$ with at most $2 \cdot |Q| \cdot [Q'|$ many states computing the composition $g \circ f : \Sigma^* \to \Delta^*$. Moreover, if $\mathcal{T}'$ is* reversible *then so is $\mathcal{T}''$.*

Proof. Wlog, we assume that transducer $\mathcal{T}$ outputs at most one letter on each transition, i.e., $\mu(t) \in \Gamma \cup \{\varepsilon\}$ for each transition t of $\mathcal{T}$. This can be done while preserving reversibility in the following way: a transition $t = (q, a, \varphi, \mathsf{op}, s)$ producing $v_1 \cdots v_n$ $(n \geq 2)$ is decomposed into n transitions $t_1 = (q, a, \varphi \wedge \overline{\mathsf{op}}, \mathsf{nop}, (q, v_1))$, $t_n = ((q, v_1 \dots v_{n-1}), a, \varphi, \mathsf{op}, s)$ and for $1 < i < n$, $t_i = ((q, v_1 \dots v_{i-1}), a, \varphi \wedge \overline{\mathsf{op}}, \mathsf{nop}, (q, v_1 \dots v_i))$. We set $\mu(t_i) = v_i$. Since we maintain the tests φ and $\overline{\mathsf{op}}$ in the sequence of transitions t_i, even if a state (q, v) happens to be used by several transitions they should be disjoint due to the reversibility of the inital transducer. It will be also more convenient to assume that, when running on $\#u$ with $u \in \Sigma^*$, the transducer $\mathcal{T}$ writes $\#v$ with $v \in \Gamma^*$. This is achieved by setting $\mu(t_0) = \#$ where $t_0 = (q_i, \#, \mathsf{true}, -, -)$ is the unique transition enabled at the initial configuration $C_0 = (q_i, \varepsilon, 0)$ of $\mathcal{T}$ (pebble stack empty and head on position 0). Finally, we also assume that the transducer $\mathcal{T}'$ always fully read its input. This will allow us to ensure that the transducer $\mathcal{T}''$ only accepts words that belong to $\mathsf{dom}(\mathcal{T})$. This can be achieved by addding to $\mathcal{T}'$ a single state $r \in Q'_{+1}$, and if $(q'_i, \#, \varphi, \mathsf{op}, s)$ is the initial transition of δ', we replace it with $(q'_i, \#, \mathsf{true}, \mathsf{nop}, r)$, $(r, a, \mathsf{true}, \mathsf{nop}, r)$ for $a \neq \#$ and $(r, \#, \varphi, \mathsf{op}, s)$. This construction preserves reversibility.

We construct the n-pebble transducer $\mathcal{T}'' = (Q'', \Sigma, \delta'', n, q''_i, q''_f, \Delta, \mu'')$ as a synchronized product of $\mathcal{T}$ and $\mathcal{T}'$. The set of states is $Q'' = (Q \times Q') \cup (\widehat{Q} \times (Q'_{-1} \cup Q'_{+1}))$ where $\widehat{Q} = \{\hat{q} \mid q \in Q\}$ is a disjoint copy of Q. We have $|Q''| \leq 2 \cdot |Q| \cdot [Q'|$. We define

$$\begin{aligned} Q''_0 &= (Q \times Q') \cup (\widehat{Q_0} \times (Q'_{-1} \cup Q'_{+1})) \\ Q''_{+1} &= (\widehat{Q_{+1}} \times Q'_{+1}) \cup (\widehat{Q_{-1}} \times Q'_{-1}) \\ Q''_{-1} &= (\widehat{Q_{+1}} \times Q'_{-1}) \cup (\widehat{Q_{-1}} \times Q'_{+1}) \,. \end{aligned}$$

The initial state is $q''_i = (q_i, q'_i) \in Q''_0$ and the final state is $q''_f = (q_i, q'_f) \in Q''_0$.

The intuition is that, in a state $(q, q') \in Q \times Q'$, transducer $\mathcal{T}''$ will synchronize a pair (t, t') of transitions of $\mathcal{T}$ and $\mathcal{T}'$ where the output $\mu(t) \in \Gamma$ of t is the input letter of t'. On the other hand, when $\mathcal{T}''$ is in a state $(\hat{q}, q')$, it will simulate the computation of $\mathcal{T}$ forward if $q' \in Q'_{+1}$ (resp. backward if $q' \in Q'_{-1}$) using transitions t of $\mathcal{T}$ producing $\mu(t) = \varepsilon$ until the computation reaches a transition t of $\mathcal{T}$ producing a letter $\mu(t) \in \Gamma$.

To handle the fact that the input of $\mathcal{T}$ is a circular word, we extend δ with the transition $t_{f,i} = (q_f, \#, \neg(\mathsf{p}_1 = \mathsf{p}_1), \mathsf{nop}, q_i)$ which is only enabled in the final configuration $(q_f, \varepsilon, 0)$ of $\mathcal{T}$ and moves to the initial configuration $(q_i, \varepsilon, 0)$. We let $\mu(t_{f,i}) = \varepsilon$.

We define now the transitions δ'' and the output function μ''. Let $t = (q, a, \varphi, \mathsf{op}, s)$ be a transition of $\mathcal{T}$. If $\mu(t) = a' \in \Gamma$, for each transition $t' = (q', a', s')$ of $\mathcal{T}'$, we introduce in $\mathcal{T}''$ the synchronized transition t'' defined in Equations ($\mathsf{tr\text{-}a}$)–($\mathsf{tr\text{-}c}$) with $\mu''(t'') = \mu'(t')$.

Firstly, we synchronize a transition and switch to the simulation mode that will search for either the next or previous production, depending on the polarity of the destination state. If $\mathcal{T}'$ is going to the right, we advance in the run of $\mathcal{T}$ and apply the transition t. If $\mathcal{T}'$ is going to the left, we need to start rewinding $\mathcal{T}$ to compute the transition which led to state q:

$$t'' = (q, q') \xrightarrow{a,\varphi,\mathsf{op}} (\hat{s}, s') \qquad \text{if } s' \in Q'_{+1} \tag{tr-a}$$

$$t'' = (q, q') \xrightarrow{a,\varphi\wedge\overline{\mathsf{op}},\mathsf{nop}} (\hat{q}, s') \qquad \text{if } s' \in Q'_{-1} \tag{tr-b}$$

$$t'' = (q, q') \xrightarrow{a,\varphi\wedge\overline{\mathsf{op}},\mathsf{nop}} (q, s') \qquad \text{if } s' \in Q'_{0} \tag{tr-c}$$

Secondly, if we are in simulation mode of $\mathcal{T}$, we need to keep simulating until we reach a non empty production of $\mathcal{T}$. Hence if $\mu(t) = \varepsilon$ then we stay in the simulation mode of $\mathcal{T}$. For each $q' \in Q'_{-1} \cup Q'_{+1}$, we introduce in $\mathcal{T}''$ the following transitions t'' with $\mu''(t'') = \varepsilon$. Note that depending on the polarity of q', we either need to advance or rewind the computation of $\mathcal{T}$. Rewinding is done following the constructions detailled in Section 4.1.

$$t'' = (\hat{q}, q') \xrightarrow{a,\varphi,\mathsf{op}} (\hat{s}, q') \qquad \text{if } q' \in Q'_{+1} \tag{mv-a}$$

$$t'' = (\hat{s}, q') \xrightarrow{a,(\varphi,\mathsf{op})^r} (\hat{q}, q') \qquad \text{if } q' \in Q'_{-1} \tag{mv-b}$$

Finally, if $\mu(t) = a' \in \Gamma$ we found the transition of $\mathcal{T}$ which produces the input letter a' to be read by $\mathcal{T}'$. Hence, we switch from the simulation mode of $\mathcal{T}$ to its synchronization mode. For each $q' \in Q_{-1} \cup Q_{+1}$, we add the following transitions t'' with $\mu''(t'') = \varepsilon$.

$$t'' = (\hat{q}, q') \xrightarrow{a,\varphi\wedge\overline{\mathsf{op}},\mathsf{nop}} (q, q') \qquad \text{if } q' \in Q'_{+1} \tag{sw-a}$$

$$t'' = (\hat{s}, q') \xrightarrow{a,(\varphi,\mathsf{op})^r} (q, q') \qquad \text{if } q' \in Q'_{-1} \tag{sw-b}$$

We prove in Appendix A.1 that the constructed transducer $\mathcal{T}''$ is deterministic (Claim A.1), and that $\mathcal{T}''$ is reverse-deterministic if $\mathcal{T}'$ is reversible (Claim A.2). We prove now the correctness of the construction, i.e., that $\mathcal{T}''$ computes the function $g \circ f$. We fix $u \in \mathsf{dom}(f) \subseteq \Sigma^*$ and $v = f(u) \in \Gamma^*$. We consider the accepting run of $\mathcal{T}$ on $\#u$ producing $\#v$:

$$C_0 = (q_i, \varepsilon, 0) \xrightarrow{t_0} C_1 \xrightarrow{t_1} C_2 \xrightarrow{t_2} \cdots C_N \xrightarrow{t_N} C_{N+1} = (q_f, \varepsilon, 0)$$

where $C_\ell = (q_\ell, \mathsf{peb}_\ell, \mathsf{h}_\ell)$ are configurations and $t_\ell \in \delta \setminus \{t_{f,i}\}$ are transitions of $\mathcal{T}$. Notice that the configurations in this accepting run are pairwise distinct. This follows from the fact that the automaton is deterministic and there are no transitions (other than $t_{f,i}$) starting from the final state q_f.

For each position $i \in \mathsf{epos}(v)$, let $\mathsf{encode}(i)$ be the index of the transition in the above run producing the letter at position i in $\#v$. Since $\mu(t_0) = \#$ we have $j_0 = 0$. Also, $\mu(t_\ell) = \varepsilon$ if $\mathsf{encode}(i) < \ell < \mathsf{encode}(i+1)$ and $\mu(t_{\mathsf{encode}(i)}) \in \Gamma \cup \{\#\}$ if $i \in \mathsf{epos}(v)$.

The main idea is to encode the head position h' of $\mathcal{T}'$ by the configuration $C_{\mathsf{encode}(\mathsf{h}')}$. More precisely, the encoding of a configuration $C' = (q', \mathsf{h}')$ of $\mathcal{T}'$ on $\#v$ is defined as $\mathsf{encode}(C') = ((q_\ell, q'), \mathsf{peb}_\ell, \mathsf{h}_\ell)$ where $\ell = \mathsf{encode}(\mathsf{h}')$. The proof of Claim 4.2 is in Appendix A.1.

▷ Claim 4.2. There is a transition $(q', \mathsf{h}') \xrightarrow{t'} (s', \mathsf{h}' + s')$ of $\mathcal{T}'$ on $\#v$ if and only if there is a nonempty run $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{+} \mathsf{encode}((s', \mathsf{h}' + s'))$ of $\mathcal{T}''$ on $\#u$ which does not use an intermediate state in $Q \times Q'$.

Now, we can show that the transducer $\mathcal{T}''$ computes the function $g \circ f$. With the notation above, assuming that $v \in \mathsf{dom}(g)$ we consider the accepting run ρ' of $\mathcal{T}'$ on $\#v$:

$$C'_0 = (q'_i, 0) \xrightarrow{t'_0} C'_1 \xrightarrow{t'_1} C'_2 \cdots C'_{N'} \xrightarrow{t'_{N'}} C'_{N'+1} = (q'_f, 0)\,. \tag{2}$$

Using Claim 4.2 we obtain the accepting run ρ'' of $\mathcal{T}''$ on $\#u$

$$\mathsf{encode}(C'_0) = ((q_i, q'_i), \varepsilon, 0) \xrightarrow{+} \mathsf{encode}(C'_1) \xrightarrow{+} \mathsf{encode}(C'_2) \cdots \mathsf{encode}(C'_{N'}) \xrightarrow{+} \mathsf{encode}(C'_{N'+1}) = ((q_i, q'_f), \varepsilon, 0)\,.$$

It is easy to see that ρ'' produces the same output string $g(v)$ as ρ'. We get $[\![\mathcal{T}'']\!](u) = g(v) = g \circ f(u)$.

Conversely, an accepting run ρ'' of $\mathcal{T}''$ on $\#u$ can be split according to its visits to states in $Q \times Q'$:

$$C''_0 = ((q_i, q'_i), \varepsilon, 0) \xrightarrow{+} C''_1 \xrightarrow{+} C''_2 \cdots C''_{N'} \xrightarrow{+} C''_{N'+1} = ((q_i, q'_f), \varepsilon, 0)\,.$$

where the C''_i are the configurations with state in $Q \times Q'$. We have $C''_0 = \mathsf{encode}(C'_0)$. By induction, and using Claim 4.2, we can easily show that, for $0 \le i \le N'$, there are transitions t'_i and configurations C'_{i+1} such that $C'_i \xrightarrow{t'_i} C'_{i+1}$ on $\#v$ and $C''_{i+1} = \mathsf{encode}(C'_{i+1})$. We deduce that (2) gives an accepting run ρ' of $\mathcal{T}'$ on $\#v$. Again, the output string $[\![\mathcal{T}'']\!](u)$ of ρ'' is the same as the output string $g(v)$ of ρ'. Because we assumed that $\mathcal{T}'$ reads its whole input, by emulating $\mathcal{T}'$ we know that the transducer $\mathcal{T}''$ fully simulates $\mathcal{T}$, which ensures that it accepts an input u if, and only if, it belongs to $\mathsf{dom}(\mathcal{T})$ and $\mathcal{T}(u)$ belongs to $\mathsf{dom}(\mathcal{T}')$.

This concludes the proof of Theorem 4.1. ◀

▶ **Example 4.3.** We illustrate the construction from Theorem 4.1, with $\mathcal{T}$ being a slight modification of the *squaring* function realized by the transducer from Figure 1, and $\mathcal{T}'$ is the function *iterated reverse* realized by the transducer from Figure 5. The *squaring* function is modified as such: instead of outputting marked letters $\underline{a}$ when reading the letter on which the pebble is placed, from state q_4, it outputs !. Now the output of this automaton is of a form that is expected for the function *iterated reverse*.

We show a fragment of the run of $\mathcal{T}''$ on the input $\#u = \#bcd$. On this input, $\mathcal{T}$ produces the output $\#v = \#!cdb!dbc!$, on which $\mathcal{T}'$ produces $!bdc!cdb$. The fragment of the run illustrated below starts when transducer $\mathcal{T}'$ reads the second ! of v, and needs to rewind the computation of $\mathcal{T}$ in order to process the infix cdb of u. Notice that to produce the second ! of v, $\mathcal{T}$ is producing the second copy of u, hence the pebble placed on the c in the initial configuration of the run. The fragment ends when $\mathcal{T}'$ is done reversing the current infix, and is about to find the previous ! symbol.

Transitions of the type ($\mathsf{mv\text{-}b}$) are represented by $\rightsquigarrow$, those of the type ($\mathsf{sw\text{-}b}$) by $\dashrightarrow$, and those of the type ($\mathsf{tr\text{-}b}$) by $\rightarrow$. Notice that in this fragment of run, no transition where $\mathcal{T}'$ needs to go right on $\#v$ are represented. The run is as follow:

$$\begin{aligned}((q_4, q'_1), 2, 2) &\xrightarrow[1]{c} ((\hat{q}_4, q'_2), 2, 1) \overset{b}{\underset{2}{\dashrightarrow}} ((q_4, q'_2), 2, 1) \xrightarrow[3]{b|b} ((\hat{q}_4, q'_2), 2, 0) \overset{\#}{\underset{4}{\rightsquigarrow}} ((\hat{q}_3, q'_2), 2, 3)\\ &\overset{d}{\underset{5}{\rightsquigarrow}} ((\hat{q}_3, q'_2), 2, 2) \overset{c}{\underset{6}{\rightsquigarrow}} ((\hat{q}_1, q'_2), \varepsilon, 1) \overset{b}{\underset{7}{\rightsquigarrow}} ((\hat{q}_5, q'_2), 1, 0) \overset{\#}{\underset{8}{\rightsquigarrow}} ((\hat{q}_4, q'_2), 1, 3)\\ &\overset{d}{\underset{9}{\dashrightarrow}} ((q_4, q'_2), 1, 3) \xrightarrow[10]{d|d} ((\hat{q}_4, q'_2), 1, 2) \overset{c}{\underset{11}{\dashrightarrow}} ((q_4, q'_2), 1, 2) \xrightarrow[12]{c|c} ((\hat{q}_4, q'_2), 1, 1)\end{aligned}$$

Notice how after transition 3, because $\mathcal{T}'$ still requires to move left, the computation of $\mathcal{T}$ is rewound and goes back to the first copy of u produced by $\mathcal{T}$. Transition 6 undoes the drop_1 operation, hence the ε in the following configuration, and transition 7 undoes the lift_1, effectively moving the pebble one position to the left.

4.3 Composition: General case

Now, we prove the general case of composition of pebble transducers.

▶ **Theorem 4.4.** *Let $\mathcal{T} = (Q, \Sigma, \delta, n, q_i, q_f, \Gamma, \mu)$ be a* reversible *n-pebble transducer (possibly with equality tests) computing a function $f\colon \Sigma^* \to \Gamma^*$. Let $\mathcal{T}' = (Q', \Gamma, \delta', m, q'_i, q'_f, \Delta, \mu')$ be a* deterministic *m-pebble transducer computing a function $g\colon \Gamma^* \to \Delta^*$. Let $r = (n+1)(m+1)-1$. We can construct a* deterministic *r-pebble transducer $\mathcal{T}'' = (Q'', \Sigma, \delta'', r, q''_i, q''_f, \Delta, \mu'')$* with equality tests *computing the composition $g \circ f\colon \Sigma^* \to \Delta^*$. The number of states of $\mathcal{T}''$ is at most $\mathcal{O}(|Q|^{m+2} \cdot |Q'| \cdot (n+1)^{m+3})$. Moreover, if $\mathcal{T}'$ is* reversible *then so is $\mathcal{T}''$.*

To simplify the construction, we first show that we may restrict to reversible pebble transducers that do not move their head when dropping or lifting a pebble.

▶ **Lemma 4.5.** *For each reversible n-pebble transducer $\mathcal{T} = (Q, \Sigma, \delta, n, q_i, q_f, \Gamma, \mu)$, we can construct a reversible n-pebble transducer $\mathcal{T}' = (Q', \Sigma, \delta', n, q'_i, q'_f, \Gamma, \mu')$ with $|Q'| \leq 3|Q|$, computing the same function, and such that every transition in $\mathcal{T}'$ which moves the head has operation* nop.

Proof. Let $Q_{\mathsf{op}} = Q \times \{\mathsf{drop}, \mathsf{lift}\}$. We let $Q' = Q \cup Q_{\mathsf{op}}$, $q'_i = q_i$, $q'_f = q_f$, $Q'_0 = Q_0 \cup Q_{\mathsf{op}}$, $Q'_{+1} = Q_{+1}$ and $Q'_{-1} = Q_{-1}$. We define now δ' and μ'. Each transition $t = (q, a, \varphi, \mathsf{nop}, s)$ of $\mathcal{T}$ also appears in $\mathcal{T}'$, and for each transition $t = (q, a, \varphi, \mathsf{op}, s)$ of $\mathcal{T}$ where $\mathsf{op} \neq \mathsf{nop}$, we define the following two transitions of $\mathcal{T}'$:

$$t' = (q, a, \varphi, \mathsf{op}, (q, \mathsf{op}))$$
$$t'' = ((q, \mathsf{op}), \mathsf{op}(\varphi) \wedge \overline{\mathsf{op}^r}, \mathsf{nop}, s)$$

with output $\mu'(t') = \mu(t)$ and $\mu'(t'') = \varepsilon$. Note that we abuse notation as we forget the index of the action op in the state (q, op). We claim that $\mathcal{T}'$ is reversible (see [11]).

Finally, it is easy to see that there is a one-to-one correspondence between the accepting runs of $\mathcal{T}$ and the accepting runs of $\mathcal{T}'$, moreover this correspondence preserves the output string produced. Hence, $[\![\mathcal{T}]\!] = [\![\mathcal{T}']\!]$. ◀

The rest of the section is devoted to the proof of Theorem 4.4.

As in Section 4.2, we assume that transducer $\mathcal{T}$ outputs at most one letter on each transition, and that it writes $\#v$ with $v \in \Gamma^*$ when running on $\#u$ with $u \in \Sigma^*$. Using Lemma 4.5, we also assume that transitions $t' = (q', a', \varphi', \mathsf{op}', s')$ of $\mathcal{T}'$ do not both drop/lift a pebble and move the head: if $\mathsf{op}' \neq \mathsf{nop}$ then $s' \in Q'_0$.

We fix $u \in \mathsf{dom}(f) \subseteq \Sigma^*$ and $v = f(u) \in \Gamma^*$. We consider the accepting run of $\mathcal{T}$ on $\#u$ producing $\#v$:

$$C_0 \xrightarrow{t_0} C_1 \xrightarrow{t_1} C_2 \xrightarrow{t_2} \cdots C_N \xrightarrow{t_N} C_{N+1} \qquad (3)$$

where $C_\ell = (q_\ell, \mathsf{peb}_\ell, \mathsf{h}_\ell)$ are configurations and $t_\ell \in \delta$ are transitions of $\mathcal{T}$.

As in Section 4.2, for each position $i \in \mathsf{epos}(v)$, we let $\mathsf{encode}(i)$ be the index of the transition in the above run producing the letter at position i in $\#v$. Again, the main idea is to encode a position $i \in \mathsf{epos}(v)$ by the configuration $C_{\mathsf{encode}(i)}$. Since $\mathcal{T}'$ has pebbles, this encoding is used not only for the head h' of $\mathcal{T}'$, but also for the pebbles dropped by $\mathcal{T}'$.

Recall that the ℓ-th configuration in (3) is $C_\ell = (q_\ell, \mathsf{peb}_\ell, \mathsf{h}_\ell)$. Consider a configuration $C' = (q', \mathsf{peb}', \mathsf{h}')$ of $\mathcal{T}'$ on $\#v$. Let $k = |\mathsf{peb}'|$, $j_i = \mathsf{encode}(\mathsf{peb}'_i)$ for $1 \leq i \leq k$, and $j = \mathsf{encode}(\mathsf{h}')$. Let $[n+1] = \{1, \ldots, n+1\}$. Define $\mathsf{encode}(C') = C'' = (q'', \mathsf{peb}'', \mathsf{h}'')$ where

$$q'' = (q_j, q', (q_{j_i})_{1 \leq i \leq k}, (1 + |\mathsf{peb}_{j_i}|)_{1 \leq i \leq k}) \in Q \times Q' \times Q^k \times [n+1]^k \subseteq Q''_0$$
$$\mathsf{peb}'' = \mathsf{peb}_{j_1}\mathsf{h}_{j_1} \cdots \mathsf{peb}_{j_k}\mathsf{h}_{j_k}\mathsf{peb}_j \qquad \text{and} \qquad \mathsf{h}'' = \mathsf{h}_j\,.$$

The set Q''_0 of 0-states of $\mathcal{T}''$ contains $\bigcup_{k=0}^{m} Q \times Q' \times Q^k \times [n+1]^k$. The initial and final states are $q''_i = (q_i, q'_i, (), ())$ and $q''_f = (q_i, q'_f, (), ())$ respectively. It remains to define the other states in Q'', the transition function δ'' and the output function μ''.

We explain below how a transition $C' \xrightarrow{t'} C'_1$ of $\mathcal{T}'$ will be simulated in $\mathcal{T}''$ by a sequence of transitions of the form $\mathsf{encode}(C') \xrightarrow{+} \mathsf{encode}(C'_1)$.

Consider a state $q'' = (q, q', \overline{x}, \overline{y}) \in Q \times Q' \times Q^k \times [n+1]^k$ for some $0 \le k \le m$, with $\overline{x} = (x_i)_{1\le i\le k}$ and $\overline{y} = (y_i)_{1\le i\le k}$. Let $t = (q, a, \varphi, \mathsf{op}, s)$ be a transition of $\mathcal{T}$ which produces an output letter $a' = \mu(t) \in \Gamma$. Consider a transition $t' = (q', a', \varphi', \mathsf{op}', s')$ of $\mathcal{T}'$ which reads the output produced by t. The goal is to synchronize the pair of transitions (t, t') as we did in Section 4.2. We first explain how to write a test ξ checking whether both t and t' are "enabled" at a configuration $C'' = (q'', \mathsf{peb}'', \mathsf{h})$ of $\mathcal{T}''$. This is intended to be used in particular when C'' is of the form $\mathsf{encode}(C')$ for some configuration C' of $\mathcal{T}'$.

We introduce some notation. For each $0 \le \ell \le k$, we let $d_\ell = y_1 + \cdots + y_\ell$ which can be recovered from the state q'' of $\mathcal{T}''$. Given an offset $d \ge 0$ and a test φ of $\mathcal{T}$, we write φ^{+d} the test obtained by adding d to the pebble indices: $\mathsf{p}_i = \mathsf{p}_j$ is replaced with $\mathsf{p}_{i+d} = \mathsf{p}_{j+d}$ and $\mathsf{h} = \mathsf{p}_j$ is replaced with $\mathsf{h} = \mathsf{p}_{j+d}$. Finally, let $\xi_0 = (\mathsf{p}_0 = \mathsf{p}_0) \wedge \neg(\mathsf{p}_{n+1} = \mathsf{p}_{n+1})$ which will be used, shifted by some offset, to check the size of the pebble stack.

▷ Claim 4.6. Let $C'' = (q'', \mathsf{peb}'', \mathsf{h})$ be a configuration of $\mathcal{T}''$ with $q'' = (q, q', \overline{x}, \overline{y})$. Let ψ be a test of $\mathcal{T}$ and ψ' be a test of $\mathcal{T}'$.

1. $\mathsf{peb}'', \mathsf{h} \models \xi_0^{+d_k}$ if and only if $d_k \le |\mathsf{peb}''| \le d_k + n$.
   We assume below that we are in this case and we write $\mathsf{peb}'' = \mathsf{peb}^1\mathsf{h}^1 \cdots \mathsf{peb}^k\mathsf{h}^k\mathsf{peb}$ with $|\mathsf{peb}^\ell\mathsf{h}^\ell| = y_\ell$ for $1 \le \ell \le k$ and $0 \le |\mathsf{peb}| \le n$.
2. $\mathsf{peb}'', \mathsf{h} \models \psi^{+d_k}$ if and only if $\mathsf{peb}, \mathsf{h} \models \psi$.
3. We can construct a formula $\overline{\xi}(q'', \psi')$ such that, for all $\mathsf{peb}', \mathsf{h}'$ with $|\mathsf{peb}'| = k$ and (a) $\mathsf{peb}'_i = \mathsf{peb}'_j$ iff $x_i = x_j$, $\mathsf{peb}^i = \mathsf{peb}^j$ and $\mathsf{h}^i = \mathsf{h}^j$; and (b) $\mathsf{h}' = \mathsf{peb}'_j$ iff $q = x_j$, $\mathsf{peb} = \mathsf{peb}^j$ and $\mathsf{h} = \mathsf{h}^j$, we have $\mathsf{peb}'', \mathsf{h} \models \overline{\xi}(q'', \psi')$ if and only if $\mathsf{peb}', \mathsf{h}' \models \psi'$.

Proof. Items 1 and 2 are clear. For Item 3 we let $\overline{\xi}(q'', \psi')$ be the formula ψ' in which we replace each atom of the form $\mathsf{h}' = \mathsf{p}'_i$ by

- false if $i > k$ or $x_i \ne q$,
- $\left(\bigwedge_{\ell=1}^{y_i-1} \mathsf{p}_{\ell+d_k} = \mathsf{p}_{\ell+d_i-1}\right) \wedge (\mathsf{h} = \mathsf{p}_{d_i}) \wedge \neg(\mathsf{p}_{y_i+d_k} = \mathsf{p}_{y_i+d_k})$ otherwise;

and each atom of the form $\mathsf{p}'_i = \mathsf{p}'_j$ by

- false if $i > k$ or $j > k$ or $x_i \ne x_j$ or $y_i \ne y_j$,
- $\bigwedge_{\ell=1}^{y_i} \mathsf{p}_{\ell+d_{i-1}} = \mathsf{p}_{\ell+d_j-1}$ otherwise. ◁

Let $\xi = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+d_k} \wedge \overline{\xi}(q'', \varphi' \wedge \overline{\mathsf{op}'})$. From Claim 4.6, assuming that $C'' = \mathsf{encode}(C')$, we see that $C'' \models \xi$ if and only if t is enabled at $C = (q, \mathsf{peb}, \mathsf{h})$ and t' is enabled at $C' = (q', \mathsf{peb}', \mathsf{h}')$. We explain now how to synchronize the pair of transitions (t, t') from state q''. Given an offset $d \ge 0$ and an operation op of $\mathcal{T}$, we write op^{+d} the operation shifted by d: $\mathsf{drop}_i^{+d} = \mathsf{drop}_{i+d}$, $\mathsf{lift}_i^{+d} = \mathsf{lift}_{i+d}$ and $\mathsf{nop}^{+d} = \mathsf{nop}$.

Case nop. We have $t' = (q', a', \varphi', \mathsf{nop}, s')$ with $s' \in Q'$. We check whether the pair (t, t') is enabled with ξ and we implement the move induced by s' of the head of $\mathcal{T}'$ on $\#v$ as we did in Section 4.2. To do so, we introduce the following transition with $\mu''(t'') = \mu'(t')$:

$$t'' = (q, q', \overline{x}, \overline{y}) \xrightarrow{a, \xi, \mathsf{op}^{+d_k}} (\hat{s}, s', \overline{x}, \overline{y}) \qquad \text{if } s' \in Q'_{+1} \qquad \text{(gtr-a)}$$

$$t'' = (q, q', \overline{x}, \overline{y}) \xrightarrow{a, \xi, \mathsf{nop}} (\hat{q}, s', \overline{x}, \overline{y}) \qquad \text{if } s' \in Q'_{-1} \qquad \text{(gtr-b)}$$

$$t'' = (q, q', \overline{x}, \overline{y}) \xrightarrow{a, \xi, \mathsf{nop}} (q, s', \overline{x}, \overline{y}) \qquad \text{if } s' \in Q'_{0} \qquad \text{(gtr-c)}$$

Here, $\widehat{Q} = \{\hat{q} \mid q \in Q\}$ is a disjoint copy of Q. The polarity of a simulation state $(\hat{q}, q', \overline{x}, \overline{y}) \in \widehat{Q} \times (Q'_{-1} \cup Q'_{+1}) \times Q^k \times [n+1]^k$ with $0 \le k \le m$ is the product of the polarity of q and the polarity of q'. When in a simulation state $(\hat{q}, q', \overline{x}, \overline{y})$, we simulate $\mathcal{T}$ forward or backward depending on the polarity of q' using transitions producing ε until we reach a transition producing a symbol from Γ. For each transition $t = (q, a, \varphi, \mathsf{op}, s)$ of $\mathcal{T}$ with $\mu(t) = \varepsilon$ we introduce the following transition t'' in $\mathcal{T}''$ with $\mu''(t'') = \varepsilon$:

$$t'' = (\hat{q}, q', \overline{x}, \overline{y}) \xrightarrow{a,\varphi^{+d_k},\mathsf{op}^{+d_k}} (\hat{s}, q', \overline{x}, \overline{y}) \qquad \text{if } q' \in Q'_{+1} \qquad \text{(gmv-a)}$$

$$t'' = (\hat{s}, q', \overline{x}, \overline{y}) \xrightarrow{a,((\varphi,\mathsf{op})^r)^{+d_k}} (\hat{q}, q', \overline{x}, \overline{y}) \qquad \text{if } q' \in Q'_{-1} \qquad \text{(gmv-b)}$$

and for each transition $t = (q, a, \varphi, \mathsf{op}, s)$ of $\mathcal{T}$ with $\mu(t) \in \Gamma$ we introduce the following transition t'' in $\mathcal{T}''$ with $\mu''(t'') = \varepsilon$ in order to switch back from the simulation mode to the synchronization mode:

$$t'' = (\hat{q}, q', \overline{x}, \overline{y}) \xrightarrow{a,(\varphi\wedge\overline{\mathsf{op}})^{+d_k},\mathsf{nop}} (q, q', \overline{x}, \overline{y}) \qquad \text{if } q' \in Q'_{+1} \qquad \text{(gsw-a)}$$

$$t'' = (\hat{s}, q', \overline{x}, \overline{y}) \xrightarrow{a,((\varphi,\mathsf{op})^r)^{+d_k}} (q, q', \overline{x}, \overline{y}) \qquad \text{if } q' \in Q'_{-1} \qquad \text{(gsw-b)}$$

As in the proof of Theorem 4.1, we can show that $\mathcal{T}''$ is reversible at simulation states of the form $(\hat{q}, q', \overline{x}, \overline{y})$. We can also prove similarly the following analog of Claim 4.2.

▷ Claim 4.7. Assuming that the operation of t' is nop, there is a transition of $\mathcal{T}'$ $C' = (q', \mathsf{peb}', \mathsf{h}') \xrightarrow{t'} C'_1 = (s', \mathsf{peb}', \mathsf{h}' + s')$ on $\#v$ if and only if there is a nonempty run $\mathsf{encode}(C') \xrightarrow{+} \mathsf{encode}(C'_1)$ of $\mathcal{T}''$ on $\#u$ where intermediate states are simulation states.

Case lift. We have $t' = (q', a', \varphi', \mathsf{lift}_{k'}, s')$ with $1 \le k' \le m$ and $s' \in Q'_0$ from our assumption on $\mathcal{T}'$ (Lemma 4.5). We still check that the pair (t, t') is enabled with $\xi = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+d_k} \wedge \overline{\xi}(q'', \varphi' \wedge \overline{\mathsf{lift}_{k'}})$. Thanks to $\overline{\xi}(q'', \overline{\mathsf{lift}_{k'}})$, we get $k' = k$, $q = x_k$, $\mathsf{peb}^k = \mathsf{peb}$ and $\mathsf{h}^k = \mathsf{h}$. We use the lift-gadget given in Figure 3 to pop the top y_k pebbles, i.e., $\mathsf{h}^k\mathsf{peb}$ (in reverse order). We enter and leave the lift-gadget with the following transitions where $s'' = (q, s', (x_1, \ldots, x_{k-1}), (y_1, \ldots, y_{k-1})) \in Q''_0$, $\xi' = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+(d_k - y_k)} \wedge \overline{\xi}(s'', \mathsf{op}'(\varphi') \wedge \overline{\mathsf{op}'^r})$, $\mu''(t''_{\mathsf{first}}) = \mu'(t')$ and $\mu''(t''_{\mathsf{last}}) = \varepsilon$:

$$t''_{\mathsf{first}} = q'' \xrightarrow{a,\xi,\mathsf{nop}} (q'', -1) \qquad \text{(lift-a)}$$

$$t''_{\mathsf{last}} = (q'', 0) \xrightarrow{a,\xi',\mathsf{nop}} s'' \qquad \text{(lift-b)}$$

The blue tests are added to ensure reverse determinism at the internal states. Also, when the before last transition is taken we see p_{d_k}. Hence, the head is on the same position at the beginning and at the end of the gadget.

Case drop. The operation of transition t' is $\mathsf{op}' = \mathsf{drop}_{k'}$ $(1 \le k' \le m)$. We still check that the pair (t, t') is enabled with $\xi = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+d_k} \wedge \overline{\xi}(q'', \varphi' \wedge \overline{\mathsf{drop}_{k'}})$. Thanks to $\overline{\xi}(q'', \overline{\mathsf{drop}_{k'}})$, we get $k' = k + 1$. We use the drop-gadget given in Figure 4 to push h and then peb (in this order) on the pebble stack. We enter and leave the drop-gadget with the following transitions where $1 \le z \le n+1$, $s'' = (q, s', (x_1, \ldots, x_k, q), (y_1, \ldots, y_k, z)) \in Q''_0$, $\xi' = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+(d_k+z)} \wedge \overline{\xi}(s'', \mathsf{op}'(\varphi') \wedge \overline{\mathsf{op}'^r})$, $\mu''(t''_{\mathsf{first}}) = \mu'(t')$ and $\mu''(t''_{\mathsf{last}}) = \varepsilon$:

$$t''_{\mathsf{first}} = q'' \xrightarrow{a,\xi,\mathsf{drop}_{d_k+z}} (q'', z, 1) \qquad \text{(drop-a)}$$

$$t''_{\mathsf{last}} = (q'', z, z) \xrightarrow{a,\xi',\mathsf{nop}} s'' \qquad \text{(drop-b)}$$

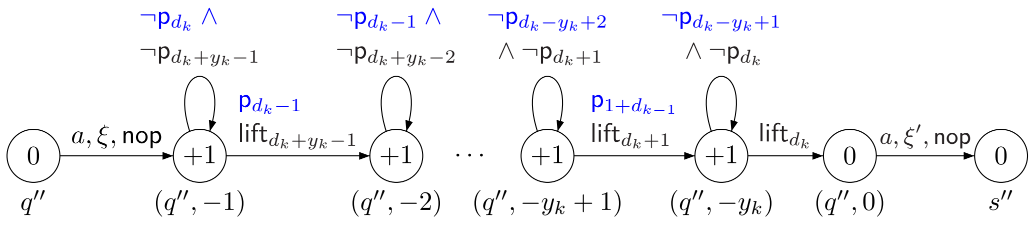

Figure 3 Simulation of lift_k. The first state is $q'' = (q, q', (x_1, \ldots, x_k), (y_1, \ldots, y_k)) \in Q''_0$ and $d_k = y_1 + \cdots + y_k$. The internal states are of the from $(q'', -\ell) \in Q''_{+1}$ for $0 \le \ell \le y_k$. The last state is $s'' = (q, s', (x_1, \ldots, x_{k-1}), (y_1, \ldots, y_{k-1})) \in Q''_0$. The test on the first transition is $\xi = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+d_k} \wedge \bar{\xi}(q'', \varphi' \wedge \overline{\mathsf{op}'})$. The test on the last transition is $\xi' = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+(d_k-y_k)} \wedge \bar{\xi}(s'', \mathsf{op}'(\varphi') \wedge \overline{\mathsf{op}'^r})$.

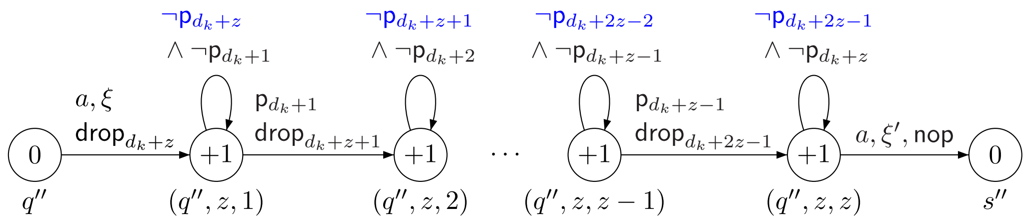

Figure 4 Simulation of drop_{k+1} with z any number such that $1 \le z \le n+1$. The first state is $q'' = (q, q', (x_1, \ldots, x_k), (y_1, \ldots, y_k)) \in Q''_0$ and $d_k = y_1 + \cdots + y_k$. The internal states are of the from $(q'', z, \ell) \in Q''_{+1}$ for $1 \le \ell \le z$. The last state is $s'' = (q, s', (x_1, \ldots, x_k, q), (y_1, \ldots, y_k, z)) \in Q''_0$. The test on the first transition is $\xi = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+d_k} \wedge \bar{\xi}(q'', \varphi' \wedge \overline{\mathsf{op}'})$. The test on the last transition is $\xi' = (\xi_0 \wedge \varphi \wedge \overline{\mathsf{op}})^{+(d_k+z)} \wedge \bar{\xi}(s'', \mathsf{op}'(\varphi') \wedge \overline{\mathsf{op}'^r})$.

On the first transition of the gadget, the test ξ makes sure that the pebble stack is of the form $\mathsf{peb}'' = \mathsf{peb}^1\mathsf{h}^1 \cdots \mathsf{peb}^k\mathsf{h}^k\mathsf{peb}$ with $|\mathsf{peb}^\ell\mathsf{h}^\ell| = y_\ell$ for $1 \le \ell \le k$ and $0 \le |\mathsf{peb}| \le n$. The operation drop_{d_k+z} on the first transition is only possible if $z = |\mathsf{peb}| + 1$ and it allows to determine the size of peb. Since $\mathsf{op}'^r = \mathsf{lift}_{k+1}$, the test ξ' on the last transition contains $\bar{\xi}(s'', \overline{\mathsf{lift}_{k+1}})$ which contains the test p_{d_k+z}. This test makes sure that at the end of the gadget, the head of $\mathcal{T}''$ is on the same position of the input word $\#u$ as it was at the beginning. The drop gadget is deterministic at the internal states. The blue tests are added to make this gadget reverse deterministic at the internal states.

We claim that $\mathcal{T}''$ is deterministic, and that if $\mathcal{T}'$ is reverse-deterministic then $\mathcal{T}''$ is reverse-deterministic (see [11] for detailed proofs).

5 Generators for Pebble Transducers

Set of generators. In [5], the class of polyregular functions is defined as the smallest class of functions closed under composition that contains the sequential functions, the squaring function and the iterated reverse function. The iterated reverse function acts on an alphabet Σ enriched with a special symbol ! and maps a word $u_0!u_1!\ldots!u_n$ to $u_0^r!u_1^r!\ldots!u_n^r$. By proving that each of these generators can easily be realized by a reversible pebble transducer and using Theorem 4.4, we prove that reversible pebble transducers realize all polyregular functions. This also gives a way to generate them using these basic blocks and composition.

Using [10, Theorem 2] as well as Remark 2.1, we get that any sequential function, realized by a transducer T with n states, can be realized by a reversible 0-pebble transducer with $\mathcal{O}(n^2)$ states. A reversible 1-pebble transducer for the squaring function was given in Example 2.2. Finally, we give a reversible 0-pebble transducer for the iterated reverse in Figure 5.

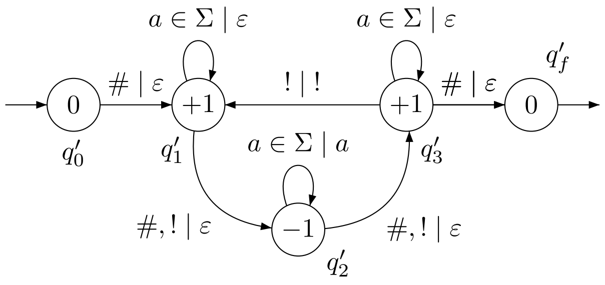

Figure 5 A 0-pebble transducer realizing the iterated reverse function. As there is no pebble, we omitted the tests and the pebble actions.

Uniformizing Pebble transducers. Another way to generate reversible pebble transducers is to start from a possibly non deterministic pebble transducer and uniformize it by a reversible pebble transducer. This section provides a procedure to do this, while preserving the number of pebbles used by the given transducer. By uniformizing a relation R, we mean extract a function f such that $\mathsf{dom}(f) = \mathsf{dom}(R)$ and $f \subseteq R$.

▶ **Theorem 5.1.** *Given a k-pebble transducer $\mathcal{T}$ with n states (possibly with equality tests), one can construct a k-pebble reversible transducer R with $2^{\mathcal{O}((kn)^2)}$ states such that $[\![R]\!]$ is a uniformization of $[\![\mathcal{T}]\!]$.*

Proof. The proof relies on the composition of reversible pebble transducers, the uniformization result from [10] and lemmas formally stated and proven in Appendix B. $\mathcal{T}$ can be decomposed into a basic reversible k-pebble transducer $\mathcal{C}_k$, a reversible 0-pebble transducer $\mathcal{C}_k^=$ enabling equality tests, and a 0-pebble transducer $\mathcal{T}_0$ with respectively $\mathcal{O}(k)$, $\mathcal{O}(2^{k^2})$ and $\mathcal{O}(kn)$ states. The transducer $\mathcal{C}_k$ associates to a word u the word of length $|\#u|^{|\#u|^k}$ that is the sequence of possible configurations of a k pebble transducer over the word u. The transducer $\mathcal{C}_k^=$ adds the truth values of equality tests to the configurations. The 0-pebble transducer $\mathcal{T}_0$ uses this information to simulate $\mathcal{T}$. Then, by Remark 2.1, $\mathcal{T}_0$ is transformed into a two-way transducer $\mathcal{T}'_0$ with $\mathcal{O}(kn)$ states. We can then use [10, Theorem 4] to obtain a reversible two-way transducer RT with $2^{\mathcal{O}((kn)^2)}$ states. Using Remark 2.1 back, RT is transformed into a 0-pebble transducer RP with $2^{\mathcal{O}((kn)^2)}$ states. We can conclude by composing $\mathcal{C}_k$ with $\mathcal{C}_k^=$ and composing the result with RP using Theorem 4.1 twice. We obtain a reversible k-pebble transducer R with $2^{\mathcal{O}((kn)^2)}$ states. ◀

References

1 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *30th International Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, volume 8 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 1–12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPICS.FSTTCS.2010.1.

2 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 9:1–9:10. ACM, 2014. doi:10.1145/2603088.2603151.

3 Nicolas Baudru and Pierre-Alain Reynier. From two-way transducers to regular function expressions. In Mizuho Hoshi and Shinnosuke Seki, editors, *22nd International Conference on Developments in Language Theory, DLT 2018*, volume 11088 of *Lecture Notes in Computer Science*, pages 96–108. Springer, 2018. doi:10.1007/978-3-319-98654-8_8.

4 Mikolaj Bojanczyk. Polyregular functions. *CoRR*, abs/1810.08760, 2018. `doi:10.48550/arXiv.1810.08760`.

5 Mikolaj Bojanczyk. Transducers of polynomial growth. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 1:1–1:27. ACM, 2022. `doi:10.1145/3531130.3533326`.

6 Mikolaj Bojanczyk, Laure Daviaud, and Shankara Narayanan Krishna. Regular and first-order list functions. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 125–134, 2018. `doi:10.1145/3209108.3209163`.

7 Mikolaj Bojanczyk, Sandra Kiefer, and Nathan Lhote. String-to-string interpretations with polynomial-size output. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 106:1–106:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPICS.ICALP.2019.106`.

8 Michal P. Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *Automata, languages and programming (Fourth Colloq., Univ. Turku, Turku, 1977)*, pages 135–137. Lecture Notes in Comput. Sci., Vol. 52. Springer, Berlin, 1977.

9 Bruno Courcelle. Monadic second-order definable graph transductions: a survey [see MR1251992 (94f:68009)]. *Theoret. Comput. Sci.*, 126(1):53–75, 1994. Seventeenth Colloquium on Trees in Algebra and Programming (CAAP '92) and European Symposium on Programming (ESOP) (Rennes, 1992). `doi:10.1016/0304-3975(94)90268-2`.

10 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 113:1–113:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.ICALP.2017.113`.

11 Luc Dartois, Paul Gastin, Loïc Germerie Guizouarn, and Shankaranarayanan Krishna. Reversible pebble transducers. *CoRR*, 2025. `doi:10.48550/arXiv.2506.11334`.

12 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular Transducer Expressions for Regular Transformations. In Martin Hofmann, Anuj Dawar, and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic In Computer Science (LICS'18)*, pages 315–324, Oxford, UK, July 2018. ACM Press. `doi:10.1145/3209108.3209182`.

13 Joost Engelfriet. Two-way pebble transducers for partial functions and their composition. *Acta Informatica*, 52(7-8):559–571, 2015. `doi:10.1007/S00236-015-0224-3`.

14 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic*, 2(2):216–254, 2001. `doi:10.1145/371316.371512`.

15 Joost Engelfriet and Sebastian Maneth. Two-way finite state transducers with nested pebbles. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 234–244. Springer, 2002. `doi:10.1007/3-540-45687-2_19`.

A Composition of pebble transducers

A.1 Composition: Simple case

▷ Claim 4.2. There is a transition $(q', \mathsf{h}') \xrightarrow{t'} (s', \mathsf{h}' + s')$ of $\mathcal{T}'$ on $\#v$ if and only if there is a nonempty run $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{+} \mathsf{encode}((s', \mathsf{h}' + s'))$ of $\mathcal{T}''$ on $\#u$ which does not use an intermediate state in $Q \times Q'$.

Proof. ($\Longrightarrow$) Assume that $\mathcal{T}''$ has a transition $(q', \mathsf{h}') \xrightarrow{t'} (s', \mathsf{h}' + s')$ on $\#v$. Let a' be the letter of $\#v$ at position h'. Let $\ell = \mathsf{encode}(\mathsf{h}')$ and $\ell' = \mathsf{encode}(\mathsf{h}' + s')$. We have $\mu(t_\ell) = a'$ and $t' = (q', a', s')$. Let t''_ℓ be the transition of $\mathcal{T}''$ constructed from the pair (t, t') using Equations ($\mathsf{tr\text{-}a}$)–($\mathsf{tr\text{-}c}$).

If $s' \in Q'_0$ then $\ell' = \ell$ and $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{t''_\ell} \mathsf{encode}((s', \mathsf{h}' + s'))$: we are done.

Assume that $s' \in Q'_{+1}$. Then $\ell' > \ell$.[2] For each $\ell < i < \ell'$, let t''_i be the transition of $\mathcal{T}''$ constructed from t_i by ($\mathsf{mv\text{-}a}$). Let $t''_{\ell'}$ be the transition of $\mathcal{T}''$ constructed from $t_{\ell'}$ by ($\mathsf{sw\text{-}a}$). We can easily check that

$$((q_\ell, q'), \mathsf{peb}_\ell, \mathsf{h}_\ell) \xrightarrow{t''_\ell} ((\widehat{q_{\ell+1}}, s'), \mathsf{peb}_{\ell+1}, \mathsf{h}_{\ell+1}) \xrightarrow{t''_{\ell+1}\cdots t''_{\ell'-1}} ((\widehat{q_{\ell'}}, s'), \mathsf{peb}_{\ell'}, \mathsf{h}_{\ell'})$$
$$\ldots \xrightarrow{t''_{\ell'}} ((q_{\ell'}, s'), \mathsf{peb}_{\ell'}, \mathsf{h}_{\ell'})$$

Therefore, $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{t''_\ell t''_{\ell+1}\cdots t''_{\ell'-1} t''_{\ell'}} \mathsf{encode}((s', \mathsf{h}' + s'))$.

The third case is $s' \in Q'_{-1}$. Then $\ell' < \ell$.[3] For each $\ell' < i < \ell$, let t''_i be the transition of $\mathcal{T}''$ constructed from t_i by ($\mathsf{mv\text{-}b}$). Let $t''_{\ell'}$ be the transition of $\mathcal{T}''$ constructed from $t_{\ell'}$ by ($\mathsf{sw\text{-}b}$). We can easily check that

$$((q_\ell, q'), \mathsf{peb}_\ell, \mathsf{h}_\ell) \xrightarrow{t''_\ell} ((\hat{q}_\ell, s'), \mathsf{peb}_\ell, \mathsf{h}_\ell - q_\ell) \xrightarrow{t''_{\ell-1}} ((\widehat{q_{\ell-1}}, s'), \mathsf{peb}_{\ell-1}, \mathsf{h}_{\ell-1} - q_{\ell-1})$$
$$\cdots \xrightarrow{t''_{\ell'+1}} ((\widehat{q_{\ell'+1}}, s'), \mathsf{peb}_{\ell'+1}, \mathsf{h}_{\ell'+1} - q_{\ell'+1}) \xrightarrow{t''_{\ell'}} ((q_{\ell'}, s'), \mathsf{peb}_{\ell'}, \mathsf{h}_{\ell'} - 0)$$

We deduce that $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{t''_\ell t''_{\ell-1}\cdots t''_{\ell'+1} t''_{\ell'}} \mathsf{encode}((s', \mathsf{h}' + s'))$ as desired.

($\Longleftarrow$) The converse can be shown similarly by noting that, depending on the polarity of s', in a run $\mathsf{encode}((q', \mathsf{h}')) \xrightarrow{+} \mathsf{encode}((s', \mathsf{h}' + s'))$ which has no intermediate states in $Q \times Q'$, the sequence of transitions used is one of

(a) a transition constructed with ($\mathsf{tr\text{-}a}$), followed by a (possibly empty) sequence of transitions constructed with ($\mathsf{mv\text{-}a}$), followed by a transition constructed with ($\mathsf{sw\text{-}a}$),

(b) a transition constructed with ($\mathsf{tr\text{-}b}$), followed by a (possibly empty) sequence of transitions constructed with ($\mathsf{mv\text{-}b}$), followed by a transition constructed with ($\mathsf{sw\text{-}b}$),

(c) a single transition constructed with ($\mathsf{tr\text{-}c}$). ◁

▷ Claim A.1. The transducer $\mathcal{T}''$ from Theorem 4.1 is deterministic.

Proof. Consider two transitions t''_1 and t''_2 starting from some state q'', reading some $a \in \Sigma$ and both enabled at some configuration $C'' = (q'', \mathsf{peb}, \mathsf{h})$ of $\mathcal{T}''$.

Assume first that $q'' = (\hat{q}, q')$ with $q' \in Q'_{+1}$. For each $i \in \{1, 2\}$, let $t_i = (q, a, \varphi_i, \mathsf{op}_i, s_i)$ be the transition of $\mathcal{T}$ giving rise to t''_i with ($\mathsf{mv\text{-}a}$) or ($\mathsf{sw\text{-}a}$). Since t''_i is enabled at configuration C'', we deduce that t_i is enabled at configuration $C = (q, \mathsf{peb}, \mathsf{h})$. Hence, t_1, t_2 are both enabled at C. We get $t_1 = t_2$ by determinism of $\mathcal{T}$ and therefore $t''_1 = t''_2$.

Assume now that $q'' = (\hat{s}, q')$ with $q' \in Q'_{-1}$. For each $i \in \{1, 2\}$, let $t_i = (q_i, a, \varphi_i, \mathsf{op}_i, s)$ be the transition of $\mathcal{T}$ giving rise to t''_i with ($\mathsf{mv\text{-}b}$) or ($\mathsf{sw\text{-}b}$). Since t''_i is enabled at configuration C'', we deduce that t^r_i is enabled at configuration $C = (s, \mathsf{peb}, \mathsf{h})$. Therefore, by (1), transition t_i is reverse-enabled at configuration $C' = (s, \mathsf{peb}, \mathsf{h} + s)$. Hence, t_1, t_2 are both reverse-enabled at C'. We get $t_1 = t_2$ by reverse-determinism of $\mathcal{T}$. Hence, $t''_1 = t''_2$.

[2] Or $\ell = |v|$ and $\ell' = 0$ which can be handled similarly using the extra transition $t_{f,i}$ producing ε allowing to move from the final configuration of $\mathcal{T}$ to its initial configuration.

[3] Or $\ell = 0$ and $\ell' = |v|$ which can be handled similarly using the extra transition $t_{f,i}$.

Finally, assume that $q'' = (q, q')$. For each $i \in \{1, 2\}$, let $t_i = (q, a, \varphi_i, \mathsf{op}_i, s_i)$, $a'_i = \mu(t_i)$ and $t'_i = (q', a'_i, s'_i)$ be the transitions of $\mathcal{T}$ and $\mathcal{T}'$ giving rise to t''_i with (tr-a), (tr-b) or (tr-c). Since t''_i is enabled at C'', we deduce that t_i is enabled at configuration $C = (q, \mathsf{peb}, \mathsf{h})$. Hence, t_1, t_2 are both enabled at C and we get $t_1 = t_2$ by determinism of $\mathcal{T}$. It follows that $a'_1 = \mu(t_1) = \mu(t_2) = a'_2$ and we get $t'_1 = t'_2$ by determinism of $\mathcal{T}'$. Therefore $t''_1 = t''_2$. ◁

▷ Claim A.2. If the transducer $\mathcal{T}'$ from Theorem 4.1 is reversible, then $\mathcal{T}''$ is reverse-deterministic.

Proof. Consider two transitions t''_1 and t''_2 of $\mathcal{T}''$ ending in some state q'', reading some $a \in \Sigma$ and both reverse-enabled at some configuration $C'' = (q'', \mathsf{peb}, \mathsf{h})$.

Assume first that $q'' = (\hat{s}, s')$ with $s' \in Q'_{+1}$. For each $i \in \{1, 2\}$, let $t_i = (q_i, a, \varphi_i, \mathsf{op}_i, s)$ be the transition of $\mathcal{T}$ giving rise to t''_i with (mv-a) or (tr-a). Since t''_i is reverse-enabled at configuration C'', we deduce that t_i is reverse-enabled at configuration $C = (s, \mathsf{peb}, \mathsf{h})$. Hence, t_1, t_2 are both reverse-enabled at C and we get $t_1 = t_2 = t$ by reverse-determinism of $\mathcal{T}$. Now, either $\mu(t) = \varepsilon$ and both t''_1 and t''_2 are constructed from t and s' with (mv-a). We get $t''_1 = t''_2$. Or $\mu(t) = a' \in \Gamma$ and t''_i is constructed from t and some transition $t'_i = (q'_i, a', s')$ of $\mathcal{T}'$ with (tr-a). Using the reverse-determinism of $\mathcal{T}'$, we deduce that $t'_1 = t'_2$ and $t''_1 = t''_2$.

Assume now that $q'' = (q, q') \in Q''_0$. Depending on the polarity of q', *both* transitions t''_1 and t''_2 are constructed with the *same* case (sw-a), (sw-b) or (tr-c) from some transitions t_1 and t_2 of $\mathcal{T}$. In the cases (sw-a) or (tr-c), the pebble stack does not change when executing t''_i whose operation is nop, and the head does not move since $q'' \in Q''_0$. Since t''_i is reverse-enabled at C'', we deduce that $\mathsf{peb}, \mathsf{h} \models \varphi_i \wedge \overline{\mathsf{op}_i}$. Hence, t_1 and t_2 are both enabled at configuration $C = (q, \mathsf{peb}, \mathsf{h})$. It follows that $t_1 = t_2$ using the determinism of $\mathcal{T}$, and then $t''_1 = t''_2$, using the reverse-determinism of $\mathcal{T}'$ in case (tr-c). Consider now the case (sw-b) and let $t_i = (q, a, \varphi_i, \mathsf{op}_i, s_i)$. Since t''_i is reversed-enabled at C'', we deduce that t^r_i is reverse-enabled at $C = (q, \mathsf{peb}, \mathsf{h})$. By (1), this implies that t_i is enabled at $(q, \mathsf{peb}, \mathsf{h} + q)$. We conclude as in the previous case that $t_1 = t_2$ and $t''_1 = t''_2$.

Finally, assume that $q'' = (\hat{q}, q') \in \widehat{Q} \times Q'_{-1}$. For each $i \in \{1, 2\}$, let $t_i = (q, a, \varphi_i, \mathsf{op}_i, s_i)$ be the transition of $\mathcal{T}$ giving rise to t''_i with (mv-b) or (tr-b). In case (mv-b), we see that t^r_i is reverse-enabled at $(q, \mathsf{peb}, \mathsf{h})$ and using (1) we deduce that t_i is enabled at $C = (q, \mathsf{peb}, \mathsf{h} + q)$. In case (tr-b), we note that the pebble stack does not change while executing transition t''_i and the head moves by the polarity of q''. Since t''_i is reverse-enabled at C'', we have $\mathsf{peb}, \mathsf{h} - q'' \models \varphi_i \wedge \overline{\mathsf{op}_i}$. Using $\mathsf{h} - q'' = \mathsf{h} + q$ we deduce that t_i is enabled at $C = (q, \mathsf{peb}, \mathsf{h} + q)$. We have shown that in both cases (mv-b) or (tr-b), transitions t_1 and t_2 are both enabled at C. It follows that $t_1 = t_2$ using the determinism of $\mathcal{T}$, and then $t''_1 = t''_2$, using the reverse-determinism of $\mathcal{T}'$ in case (tr-b). ◁

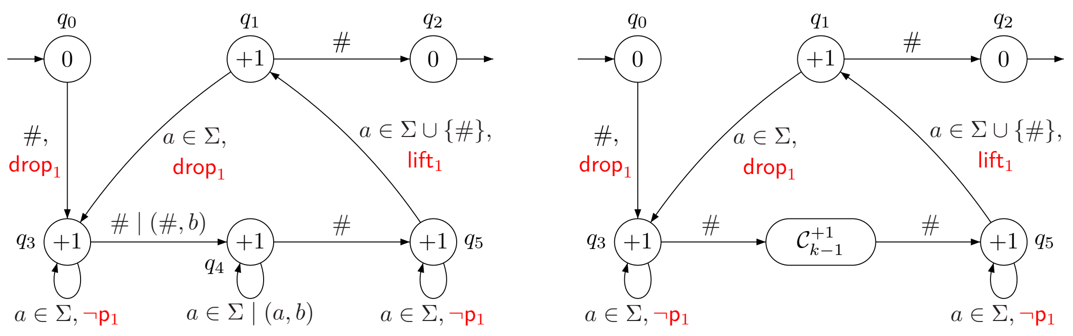
q_0
0
q_1
+1
$\#$
q_2
0
$\#$, drop_1
$a \in \Sigma$, drop_1
$a \in \Sigma \cup \{\#\}$, lift_1
q_3
+1
$\# \mid (\#, b)$
+1
q_4
$\#$
+1
q_5
$a \in \Sigma, \neg\mathsf{p}_1$
$a \in \Sigma \mid (a, b)$
$a \in \Sigma, \neg\mathsf{p}_1$
q_0
0
q_1
+1
$\#$
q_2
0
$\#$, drop_1
$a \in \Sigma$, drop_1
$a \in \Sigma \cup \{\#\}$, lift_1
q_3
+1
$\#$
$\mathcal{C}_{k-1}^{+1}$
$\#$
+1
q_5
$a \in \Sigma, \neg\mathsf{p}_1$
$a \in \Sigma, \neg\mathsf{p}_1$

Figure 6 Reversible transducers for the function $\mathcal{C}_1$ (left) and $\mathcal{C}_k$ (right). The production b of $\mathcal{C}_1$ is the bit corresponding to $\mathsf{h} = \mathsf{p}_1$. The transitions are actually duplicated. The self-loop on q_4 reading $a \in \Sigma$ stands for two transitions $(q_4, a, \mathsf{p}_1, q_4)$ producing $(a, 1)$ and $(q_4, a, \neg\mathsf{p}_1, q_4)$ producing $(a, 0)$. Similarly, the transition from q_3 to q_4 reading $\#$ is duplicated.
In $\mathcal{C}_k$, we use a copy $\mathcal{C}_{k-1}^{+1}$ of $\mathcal{C}_{k-1}$ where all the pebble indices are incremented by 1. The transition in $\mathcal{C}_k$ from q_3 labelled $\#$ goes to the initial state of $\mathcal{C}_{k-1}^{+1}$ which is a 0-state, hence the head does not move. Similarly, the transition in $\mathcal{C}_k$ labelled $\#$ going to q_5 starts from the final state of $\mathcal{C}_{k-1}^{+1}$. When it appears in $\mathcal{C}_k$, b stands for the bit vector giving the truth value of each $\mathsf{h} = \mathsf{p}_i$ for $1 \leq i \leq k$. It is actually a macro for 2^k disjoint transitions.

B Generators for pebble transducers

We give the supporting lemmas and their proof required for Theorem 5.1.

Given an integer $k \geq 1$ and a word u, we define the marking of u for k-configurations as the word $\mathcal{C}_k(u)$ on $(\Sigma \cup \{\#\}) \times \{0,1\}^k$ which is the lexicographically ordered sequence of every possible marking of k positions in $\#u$. For instance, we have $\mathcal{C}_1(ab) = \begin{matrix} \# & a & b & \# & a & b & \# & a & b \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{matrix}$ and

$$\mathcal{C}_2(ab) = \begin{matrix} \# & a & b & \# & a & b & \# & a & b & \# & a & b & \# & a & b & \# & a & b & \# & a & b & \# & a & b & \# & a & b \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{matrix}.$$

▶ **Lemma B.1.** *Given an integer $k \geq 1$ and an alphabet Σ, one can construct a reversible k-pebble transducer $\mathcal{C}_k$ with $\mathcal{O}(k)$ states such that for all input word $u \in \Sigma^*$, $\mathcal{C}_k(u)$ is the marking of u for k-configurations.*

Proof. The machine for $\mathcal{C}_1$ is given in Figure 6 (left). It is a slightly modified version of the squaring function of Figure 1, mainly to accomodate the fact that $\mathcal{C}_1$ also produces the endmarker $\#$ at each iteration.

For $k > 1$, the machine $\mathcal{C}_k$ is defined using an enhanced $\mathcal{C}_{k-1}$ as a blackbox. Indeed, the marking for k-configurations over a word u can be defined as the $k-1$ marking iterated $|\#u|$ times, where each iteration is marked by the new pebble on a different position of $\#u$, in order. Then $\mathcal{C}_k$ uses a modified $\mathcal{C}_{k-1}^{+1}$ that outputs its marking plus the one of the new pebble. It is described in Figure 6 (right). ◀

▶ **Lemma B.2.** *Given an integer $k \geq 1$ and an alphabet Σ, one can construct a reversible 0-pebble transducer $\mathcal{C}_k^{=}$ with $\mathcal{O}(2^{k^2})$ states that reads $[\![\mathcal{C}_k]\!](u)$ and adds to each copy of $\#u$ in $[\![\mathcal{C}_k]\!](u)$ a $k \times k$ boolean matrix M such that $M_{i,j} = 1$ if and only if i and j mark the same position in this copy of $\#u$.*

Sketch of proof. Informally, the transducer $\mathcal{C}_k^=$ reads a marking of $\#u$ and computes the matrix M along this copy. Then, when reaching a new pair $(\#, b)$, it goes back to output the copy enriched with the matrix M. Special care has to be taken to ensure reverse-determinism. In particular, the transducer $\mathcal{C}_k^=$ needs to undo the computation done before moving again to the next marked copy of $\#u$. Details can be found in [11]. ◀

▶ **Lemma B.3.** *Given a k-pebble ($k \geq 1$) transducer $\mathcal{T}$ with n states, one can construct a 0-pebble transducer $\mathcal{T}_0$ with $\mathcal{O}(kn)$ states such that $[\![\mathcal{T}]\!] = [\![\mathcal{T}_0]\!] \circ [\![\mathcal{C}_k^=]\!] \circ [\![\mathcal{C}_k]\!]$.*

Proof. The idea is to simulate the moves of the pebbles of $\mathcal{T}$ with moves of the reading head of $\mathcal{T}_0$ along the sequence of k-configurations produced by $\mathcal{C}_k$. The tests φ are validated using the information added by $\mathcal{C}_k^=$.

Informally, the transducer $\mathcal{T}_0$ remembers in its state the number of pebbles dropped by $\mathcal{T}$. It uses the configuration where the unused pebbles are on the same position as the reading head. Then, when the simulation of $\mathcal{T}$ drops a pebble i, $\mathcal{T}_0$ is already reading a configuration where the pebble i is at the current position. The transducer $\mathcal{T}_0$ simply needs to increment the number of pebbles dropped. Conversely, if a pebble i is lifted, it means the reading head is on a position where the pebble i is present. Hence it only needs to decrement its number of pebbles dropped.

The move of the reading head uses the fact that the configurations are output by $\mathcal{C}_k$ in lexicographic order. If the head moves to the left while having i pebbles dropped, $\mathcal{T}_0$ moves left until it sees a position where pebbles $i+1$ to k are all present. The first such position is the previous one from where we started due to the lexicographic order. It is also the correct position to maintain the invariant that all undropped pebbles are placed at the same position as the head. A move right is treated symmetrically.

Because the input word of u is read in a cyclic fashion, this might create issue for $\mathcal{T}_0$ as consecutive endmarkers $\#$ do not belong to the same marking. More precisely, given any position in $\mathcal{C}_k(u)$, its corresponding endmarker is the closest (possibly itself) $\#$-labelled position on its left. So if during the computation $\mathcal{T}_0$ reaches the endmarker $\#$ moving left and need to keep moving left, $\mathcal{T}_0$ needs to move to a position where all pebbles 1 to i are unchanged, but pebbles $i+1$ to k are on the last letter of the word. This is done by reaching the closest position on the right where pebbles $i+1$ to k are on an endmarker $\#$ (or $\dashv$ in the case where we reach the end of the word) then moving to the position on the left.

Symmetrically, if the computation of $\mathcal{T}_0$ reaches an endmarker $\#$ with a single right move, it means that pebbles $i+1$ to k were on the last position of u, then $\mathcal{T}_0$ moves left to the closest position where pebbles $i+1$ to k are all on the endmarker $\#$.

Formally, let $\mathcal{T} = (Q, \Sigma, \delta, k, q_i, q_f, \Gamma, \mu)$. We define $\mathcal{T}_0 = (P, \Sigma', \gamma, 0, p_i, p_f, \Gamma, \nu)$ where $\Sigma' = \Sigma \times \{0,1\}^k \times \{0,1\}^{k^2}$, $P \subseteq (Q \times \{0, \ldots, k\} \times \{s, m_r, m_\ell\}) \cup \{p_i, p_f\}$ where s stands for simulation, and m_ℓ, m_r stand for move left and right respectively. We divide P into

$$\begin{aligned} P_0 &= (Q \times \{0, \ldots, k\} \times \{s\}) \cup \{p_i, p_f\}, \\ P_{+1} &= Q \times \{0, \ldots, k\} \times \{m_r\}, \\ P_{-1} &= Q \times \{0, \ldots, k\} \times \{m_\ell\}\,. \end{aligned}$$

To avoid the more convoluted cases, we assume that in $\mathcal{T}$ transitions with action drop or lift do not move the reading head (this is especially needed for $\#$). Note that this can be enforced by decomposing a drop and move transition into two transitions (see Lemma 4.5).

For $0 \leq i \leq k$, we denote by b^{+i} any vector where $b_j^{+i} = 1$ for all $i < j \leq k$ and we denote by b^{-i} any vector where $b_j^{-i} \neq 1$ for some $j > i$.

Given an integer $0 \leq i \leq k$, a bit vector b and a matrix M, we say that $b, M, i \models (\mathsf{h} = \mathsf{p}_\ell)$ (resp. $\mathsf{p}_\ell = \mathsf{p}_{\ell'}$) if $\ell \leq i$ and $b_\ell = 1$ (resp. $\ell, \ell' \leq i$ and $M_{\ell,\ell'} = 1$).

The transitions in γ and the output function ν are defined below, where a denotes a letter different from $\#$, σ is any letter (possibly $\#$), b is any k-bit vector and M is a $k \times k$ boolean matrix. We omit the tests and operations in transitions as $\mathcal{T}_0$ is pebbleless. All transitions output ε except those of Item 3 below.

1. $(p_i, \vdash\!\dashv, (q_i, 0, m_r))$ and $((q_i, 0, m_r), (\#, b^{+0}, M_1), (q_i, 0, s))$.
2. $((q_f, 0, s), (\#, b^{+0}, M_1), (q_f, 0, m_\ell))$ and $((q_i, 0, m_r), \vdash\!\dashv, p_f)$.
3. $((q, i, s), (\sigma, b^{+i}, M), (q', i', m))$ if there exists a transition $t = (q, \sigma, \varphi, \mathsf{op}, q')$ in $\mathcal{T}$ such that $b^{+i}, M, i \models \varphi$ and $\mathsf{op} \in \{\mathsf{nop}, \mathsf{drop}_{i+1}, \mathsf{lift}_i\}$ and $b_i = 1$ if $\mathsf{op} = \mathsf{lift}_i$. In this case,

$$i' = \begin{cases} i & \text{if } \mathsf{op} = \mathsf{nop} \\ i+1 & \text{if } \mathsf{op} = \mathsf{drop}_{i+1} \\ i-1 & \text{if } \mathsf{op} = \mathsf{drop}_i \end{cases} \qquad \text{and} \qquad m = \begin{cases} s & \text{if } q' \in Q_0 \\ m_\ell & \text{if } q' \in Q_{-1} \wedge \sigma \neq \# \\ m_r & \text{if } q' \in Q_{+1} \text{ or } q' \in Q_{-1} \wedge \sigma = \# \end{cases}$$

   The production of this transition is the same as t.
   Note that, the head of $\mathcal{T}_0$ moves left or right when $q' \in Q_{-1}$ or $q' \in Q_{+1}$. But in general, the head of $\mathcal{T}_0$ does not reach immediately the position where the simulation of $\mathcal{T}$ continues. We have to skip positions with a bit vector of the form $b'^{-i'}$ and handle carefully the endmarker $\#$. This is the purpose of the following transitions.
4. If $q \in Q_{-1}$ then $((q, i, m_\ell), (\sigma, b^{-i}, M), (q, i, m_\ell))$ and $((q, i, m_\ell), (\sigma, b^{+i}, M), (q, i, s))$.
5. If $q \in Q_{-1}$ then $((q, i, m_r), (a, b, M), (q, i, m_r))$ and $((q, i, m_r), (\#, b^{-i}, M), (q, i, m_r))$ and $((q, i, m_r), (\#, b^{+i}, M), (q, i, m_\ell))$ and $((q, i, m_r), \vdash\!\dashv, (q, i, m_\ell))$.
   Note that, when moving right we reach $\#$ with a bit vector of the form b^{+i} then the previous letter also has a bit vector of the form b^{+i}. Hence, the transition taken from (q, i, m_ℓ) will go directly to the simulation mode (q, i, s). If moving right we reach the endmarker $\vdash\!\dashv$ then the previous letter has a bit vector of the form b^{+0}.
6. If $q \in Q_{+1}$ then $((q, i, m_r), (\sigma, b^{-i}, M), (q, i, m_r))$ and $((q, i, m_r), (a, b^{+i}, M), (q, i, s))$.
7. If $q \in Q_{+1}$ then $((q, i, m_r), (\#, b^{+i}, M), (q, i, m_\ell))$ and $((q, i, m_r), \vdash\!\dashv, (q, i, m_\ell))$ and $((q, i, m_\ell), (a, b, M), (q, i, m_\ell))$ and $((q, i, m_\ell), (\#, b^{-i}, M), (q, i, m_\ell))$ and $((q, i, m_\ell), (\#, b^{+i}, M), (q, i, s))$.

We remark that $\mathcal{T}_0$ is obtained from $\mathcal{T}$ by decomposing its transitions into separate sequences of transitions. The bit vectors and the matrices M allow us to check the tests of transitions of $\mathcal{T}$. The producing transitions of $\mathcal{T}_0$ correspond to the ones of $\mathcal{T}$ and the accepting runs of $\mathcal{T}_0$ correspond to the accepting runs of $\mathcal{T}$. Hence for any input word u, $u \in \mathsf{dom}(\mathcal{T})$ if, and only if $[\![\mathcal{C}_k^{=}]\!]([\![\mathcal{C}_k]\!](u)) \in \mathsf{dom}(\mathcal{T}_0)$ and $[\![\mathcal{T}]\!] = [\![\mathcal{T}_0]\!] \circ [\![\mathcal{C}_k^{=}]\!] \circ [\![\mathcal{C}_k]\!]$. ◀