Hiding Pebbles When the Output Alphabet Is Unary

Gaëtan Douéneau-Tabot
IRIF, Université Paris Cité & Direction générale de l'armement – Ingénierie de projets, France

Abstract

Pebble transducers are nested two-way transducers which can drop marks (named "pebbles") on their input word. Blind transducers have been introduced by Nguyên et al. as a subclass of pebble transducers, which can nest two-way transducers but cannot drop pebbles on their input.

In this paper, we study the classes of functions computed by pebble and blind transducers, when the output alphabet is unary. Our main result shows how to decide if a function computed by a pebble transducer can be computed by a blind transducer. We also provide characterizations of these classes in terms of Cauchy and Hadamard products, in the spirit of rational series. Furthermore, pumping-like characterizations of the functions computed by blind transducers are given.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases polyregular functions, pebble transducers, rational series, factorization forests, Cauchy product, Hadamard product

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.120

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2112.10212`

Acknowledgements The author is grateful to Olivier Carton, Lê Thành Dũng Nguyên and Pierre Pradic for discussing about this work.

1 Introduction

Transducers are finite-state machines obtained by adding outputs to finite automata. In this paper, we assume that these machines are always deterministic and have finite inputs, hence they compute functions from finite words to finite words. In particular, a *deterministic two-way transducer* consists of a two-way automaton which can produce outputs. This model computes the class of *regular functions*, which is often considered as one of the functional counterparts of *regular languages*. It has been largely studied for its numerous regular-like properties: closure under composition [4], equivalence with logical transductions [9] or regular transducer expressions [5], decidable equivalence problem [11], etc.

Pebble transducers and blind transducers. Two-way transducers can only describe functions whose output size is at most linear in the input size. A possible solution to overcome this limitation is to consider nested two-way transducers. In particular, the nested model of *pebble transducers* has been studied for a long time (see e.g. [10, 8]).

A *k-pebble transducer* is built by induction on $k \geq 1$. For $k = 1$, a 1-pebble transducer is just a two-way transducer. For $k \geq 2$, a k-pebble transducer is a two-way transducer that, when on any position i of its input word, can launch a $(k-1)$-pebble transducer. This submachine works on the original input where position i is marked by a "pebble". The original two-way transducer then outputs the concatenation of all the outputs returned by the submachines that it has launched along its computation. The intuitive behavior of a 3-pebble transducer is depicted in Figure 2. It can be seen as program with 3 nested loops. The class of word-to-word functions computed by k-pebble transducers for some $k \geq 1$ is known as *polyregular functions*. It has been quite intensively studied over the past few years due to its regular-like properties such as closure under composition [8], equivalence with logical interpretations [3] or other transducer models [2], etc.

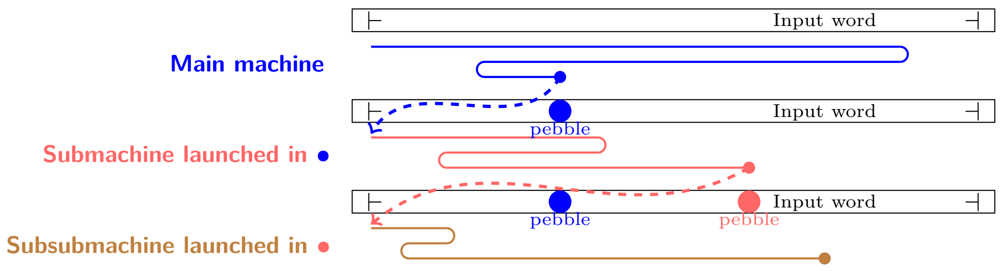


Figure 1 Behavior of a 3-pebble transducer.

A subclass of pebble transducers named *blind transducers* was recently introduced in [13]. For $k = 1$, a 1-blind transducer is just a two-way transducer. For $k \geq 2$, a k-blind transducer is a two-way transducer that can launch a $(k-1)$-blind transducer like a k-pebble transducer. However, there is no pebble marking the input of the submachine (i.e. it cannot see the position i from which it was called). The behavior of a 3-blind transducer is depicted in Figure 2. It can be seen as a program with 3 nested loops which cannot see the upper loop indexes. We call *polyblind functions* the class of functions computed by blind transducers. It is closed under composition and deeply related to lambda-calculus [12].

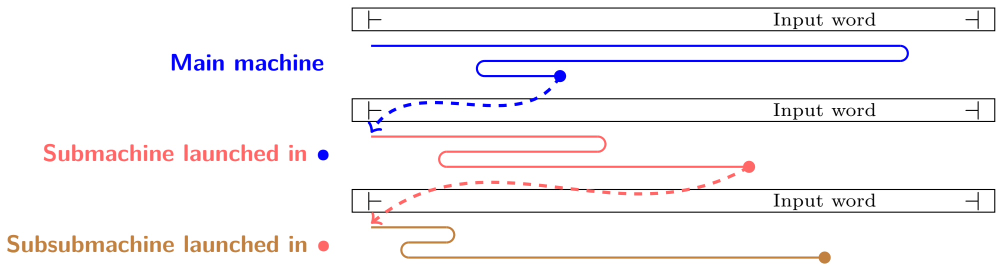


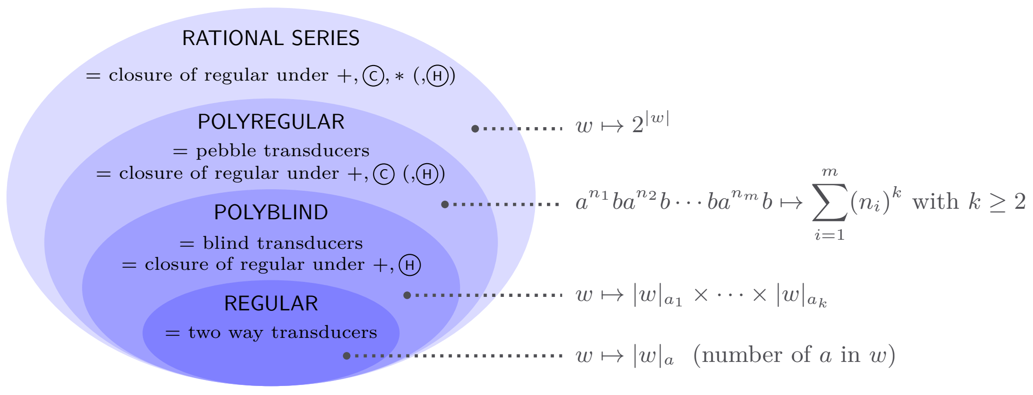
Figure 2 Behavior of a 3-blind transducer.

We study here polyregular and polyblind functions whose output alphabet is unary. Up to identifying a word with its length, we thus consider functions from finite words to $\mathbb{N}$. With this restriction, we show that one can decide if a polyregular function is polyblind, and connect these classes of functions to *rational series*.

Relationship with rational series. Rational series over the semiring $(\mathbb{N}, +, \times)$. are a well-studied class of functions from finite words to $\mathbb{N}$. They can be defined as the closure of (unary output) regular functions under sum $+$, Cauchy product Ⓒ (product for formal power series) and Kleene star $*$ (iteration of Cauchy products). It is also well known that rational series are closed under Hadamard product Ⓗ (component-wise product) [1].

The first result of this paper states that polyregular functions are exactly the subclass of rational series "without star", that is the closure of regular functions under $+$ and Ⓒ (Ⓗ can also be used but it is not necessary). This theorem is obtained by combining several former works. Our second result establishes that polyblind functions are exactly the closure of regular functions under $+$ and Ⓗ. It is shown in a self-contained way.

The aforementioned classes are depicted in Figure 3. All the inclusions are strict and this paper provides a few separating examples (some of them were already known in [6]).



RATIONAL SERIES
= closure of regular under $+$, ⓒ, $*$ (,Ⓗ)
POLYREGULAR
= pebble transducers
= closure of regular under $+$, ⓒ (,Ⓗ)
POLYBLIND
= blind transducers
= closure of regular under $+$, Ⓗ
REGULAR
= two way transducers
$w \mapsto 2^{|w|}$
$a^{n_1}ba^{n_2}b \cdots ba^{n_m}b \mapsto \sum_{i=1}^{m} (n_i)^k$ with $k \geq 2$
$w \mapsto |w|_{a_1} \times \cdots \times |w|_{a_k}$
$w \mapsto |w|_a$ (number of a in w)

Figure 3 Classes of functions from finite words to $\mathbb{N}$ studied in this paper.

Class membership problems. We finally show how to decide whether a polyregular function with unary output is polyblind. It is by far the most involved and technical result of this paper. Furthermore, the construction of a blind transducer is effective, hence this result can be viewed as program optimization. Indeed, given a program with nested loops, our algorithm is able to build an equivalent program using "blind" loops if it exists.

In general, decision problems for transductions are quite difficult to solve, since contrary to regular languages, there are no known "canonical" objects (such as a minimal automaton) to represent (poly)regular functions. It is thus complex to decide an intrinsic property of a function, since it can be described in several seemingly unrelated manners. Nevertheless, the membership problem from rational series to polyregular (resp. regular) functions was shown to be decidable in [7, 6]. It is in fact equivalent to checking if the output of the rational series is bounded by a polynomial (resp. a linear function) in its input's length.

However, both polyregular and polyblind functions can have polynomial growth rates. To discriminate between them, we thus introduce the new notion of *repetitiveness* (which is a pumping-like property for functions) and show that it exactly captures the polyregular functions that are polyblind. The proof is a rather complex induction on the depth $k \geq 1$ of the k-pebble transducer representing the function. We show at the same time that repetitiveness is decidable and that a blind transducer can effectively be built whenever this property holds. Partial results were obtained in [6] to decide "blindness" of the functions computed by 2-pebble transducers. Some of our tools are inspired by this paper, such as the use of bimachines and factorization forests. Nevertheless, our general result requires new proof techniques (e.g. the induction techniques which insulate the term of "highest growth rate" in the function) and concepts (e.g. repetitiveness).

Outline. We first describe in Section 2 the notions of pebble and blind transducers. In the case of unary outputs, we recall the equivalent models of pebble, marble and blind bimachines introduced in [6]. These bimachines are easier to handle in the proofs, since they manipulate a monoid morphism instead of having two-way moves. In Section 3 we recall the definitions of sum $+$, Cauchy product ⓒ, Hadamard product Ⓗ and Kleene star $*$ for rational series. We then show how to describe polyregular and polyblind functions with these operations. Finally, we claim in Section 4 that the membership problem from polyregular to polyblind functions is decidable. The proof of this technical result is sketched in sections 5 and 6. Due to space constraints we focus on the most significant lemmas.

2 Preliminaries

$\mathbb{N}$ is the set of nonnegative integers. If $i \leq j$, the set $[i{:}j]$ is $\{i, i{+}1, \dots, j\} \subseteq \mathbb{N}$ (empty if $j < i$). The capital letter A denotes an alphabet, i.e. a finite set of letters. The empty word is denoted by ε. If $w \in A^*$, let $|w| \in \mathbb{N}$ be its length, and for $1 \leq i \leq |w|$ let $w[i]$ be its i-th letter. If $I = \{i_1 < \cdots < i_\ell\} \subseteq [1{:}|w|]$, let $w[I] \coloneqq w[i_1] \cdots w[i_\ell]$. If $a \in A$, let $|w|_a$ be the number of letters a occurring in w. We assume that the reader is familiar with the basics of automata theory, in particular one-way and two-way automata, and monoid morphisms.

Two-way transducers. A deterministic two-way transducer is a deterministic two-way automaton (with input in A^*) enhanced with the ability to produce outputs (from B^*) when performing a transition. The output of the transducer is defined as the concatenation of these productions along the unique accepting run on the input word (if it exists): it thus describes a (partial) function $A^* \to B^*$. Its behavior is depicted in Figure 4. A formal definition can be found e.g. in [6]. These machines compute the class of *regular functions*.

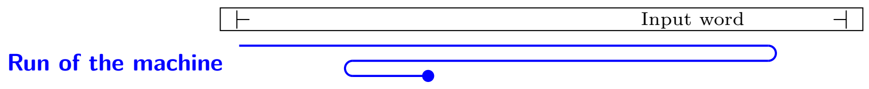


Figure 4 Behavior of a two-way transducer.

▶ **Example 2.1.** The function $a_1 \cdots a_n \mapsto a_1 \cdots a_n \# a_n \cdots a_1$ can be computed by a two-way transducer which reads its input word from left to right and then from right to left.

From now on, the output alphabet B of our machines will always be a singleton. By identifying B^* and $\mathbb{N}$, we assume that the functions computed have type $A^* \to \mathbb{N}$.

▶ **Example 2.2.** Given $a \in A$, the function $\mathsf{nb}_a : A^* \to \mathbb{N}, w \mapsto |w|_a$ is regular.

Blind and pebble transducers. Blind and pebble transducers extend two-way transducers by allowing to "nest" such machines. A 1-blind (resp. 1-pebble) transducer is just a two-way transducer. For $k \geq 2$, a k-blind (resp. k-pebble) transducer is a two-way transducer which, when performing a transition from a position $1 \leq i \leq |w|$ of its input $w \in A^*$, can launch a $(k{-}1)$-blind (resp. $(k{-}1)$-pebble) transducer with input w (resp. $w[1{:}i{-}1]\underline{w[i]}w[i{+}1{:}|w|]$ i.e. w where position i is marked). The two-way transducer then uses the output of this submachine as if it was the output produced along its transition. The intuitive behaviors are depicted for $k = 3$ in figures 1 and 2. Formal definitions can be found e.g. in [13, 6].

▶ **Example 2.3.** Let $a_1, \dots, a_k \in A$, then $\mathsf{nb}_{a_1,\dots,a_k} : w \mapsto |w|_{a_1} \times \cdots \times |w|_{a_k}$ can be computed by a k-blind transducer. The main transducer processes its input from left to right, and it calls inductively a $(k{-}1)$-blind transducer for $\mathsf{nb}_{a_1,\dots,a_{k-1}}$ each time it it sees an a_k.

▶ **Example 2.4.** The function $2\text{-}\mathsf{powers} : a^{n_1}b \cdots a^{n_m}b \mapsto \sum_{i=1}^{m} n_i^2$ can be computed by a 2-pebble transducer. Its main transducer ranges over all the a of the input, and calls a 1-pebble (= two-way) transducer for each a, which produces n_i if the a is in the i-th block (it uses the pebble to detect which block is concerned). Similarly, the function $k\text{-}\mathsf{powers} : a^{n_1}b \cdots a^{n_m}b \mapsto \sum_{i=1}^{m} n_i^k$ for $k \geq 1$ can be computed by a k-pebble transducer.

▶ **Definition 2.5.** *We define the class of* polyregular functions *(resp.* polyblind functions*) as the class of functions computed by a k-pebble (resp. k-blind) transducer for some $k \geq 1$.*

It is not hard to see that polyblind functions are a subclass of polyregular functions. Indeed, a blind transducer is just a pebble transducer "without pebbles".

Bimachines. In this paper, we shall describe formally the regular, polyregular and polyblind functions with another computation model. A *bimachine* is a transducer which makes a single left-to-right pass on its input, but it can use a morphism into a finite monoid to check regular properties of the prefix (resp. suffix) ending (resp. starting) in the current position. This notion of bimachines enable us to easily use algebraic techniques in the proofs, and in particular *factorization forests* over finite monoids.

▶ **Definition 2.6.** *A bimachine $\mathcal{M} := (A, M, \mu, \lambda)$ is:*

- *an input alphabet A, a finite monoid M and a monoid morphism $\mu : A^* \to M$;*
- *an output function $\lambda : M \times A \times M \to \mathbb{N}$.*

$\mathcal{M}$ computes $f : A^* \to \mathbb{N}$ defined by $f(w) := \sum_{i=1}^{|w|} \lambda(\mu(w[1{:}i{-}1]), w[i], \mu(w[i{+}1{:}|w|]))$ for $w \in A^*$. The production of each term of this sum is depicted intuitively in Figure 5.

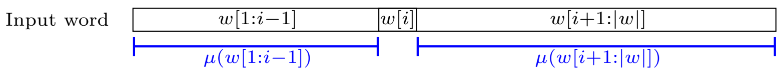


Figure 5 Behavior of a bimachine when producing $\lambda(\mu(w[1{:}i{-}1]), w[i], \mu(w[i{+}1{:}|w|))$.

When dealing with bimachines, we consider without loss of generality total functions such that $f(\varepsilon) = 0$ (the domains of two-way transducers are regular languages [14], and the particular image of ε does not matter). In this context, it is well known that regular functions of type $A^* \to \mathbb{N}$ are exactly those computed by bimachines (in other words, it means that regular functions and rational functions are the same). Now we recall how [6] has generalized this result to k-blind and k-pebble transducers. Intuitively, a *bimachine with external functions* is a bimachine enriched with the possibility to launch a submachine for each letter of the input (it outputs the sum of all the outputs returned by these submachines).

▶ **Definition 2.7** ([6]). *A* bimachine with external pebble (*resp.* external blind, *resp.* external marble) functions $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ *consists of:*

- *an input alphabet A;*
- *a finite monoid M and a morphism $\mu : A^* \to M$;*
- *a finite set $\mathfrak{H}$ of external functions $\mathfrak{h} : (A \uplus \underline{A})^* \to \mathbb{N}$ (resp. $A^* \to \mathbb{N}$, resp. $A^* \to \mathbb{N}$);*
- *an output function $\lambda : M \times A \times M \to \mathfrak{H}$.*

Given $1 \leq i \leq |w|$ a position of $w \in A^*$, let $\mathfrak{h}_i := \lambda(\mu(w[1{:}i{-}1]), w[i], \mu(w[i{+}1{:}|w|])) \in \mathfrak{H}$. A bimachine $\mathcal{M}$ with external pebble functions computes a function $f : A^* \to \mathbb{N}$ defined by $f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{h}_i(w[1{:}i{-}1]\underline{w[i]}w[i{+}1{:}|w|])$. Intuitively, it means that for each position $1 \leq i \leq |w|$, $\mathcal{M}$ calls an external function $\mathfrak{h}_i$ (depending on a regular property of $w[1{:}i{-}1]$,$w[i]$ and $w[i{+}1{:}|w|]$) with input $w[1{:}i{-}1]\underline{w[i]}w[i{+}1{:}|w|]$ (that is "w with a pebble on position i"), and uses the result $\mathfrak{h}_i(w[1{:}i{-}1]\underline{w[i]}w[i{+}1{:}|w|])$ of this function in its own output.

For a bimachine with external blind (resp. marble) functions, the output is defined by $f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{h}_i(w)$ (resp. $f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{h}_i(w[1{:}i]))$. In this case $\mathcal{M}$ calls $\mathfrak{h}_i$ with argument w (resp. the prefix of w ending in position i).

▶ **Definition 2.8** ([6]). *Given $k \geq 1$, a k-pebble (resp. k-blind, resp. k-marble) bimachine is:*

- *for $k = 1$, a bimachine (withtout external functions, see Definition 2.6);*
- *for $k \geq 2$, it a bimachine with external pebble (resp. external blind, resp. external marble) functions (see Definition 2.7) which are computed by $(k{-}1)$-pebble (resp. $(k{-}1)$-blind, resp. $(k{-}1)$-marble) bimachines. These $(k{-}1)$-bimachines are implicitly fixed and given by the external functions of the main bimachine.*

▶ Remark 2.9. For pebble bimachines, a natural question is whether the inner bimachines can ask which pebble was dropped by which ancestor, or whether they can only see that "there is a pebble". Both models are in fact equivalent, since the number of pebbles is bounded.

Now we recall in Theorem 2.10 that pebble and blind bimachines are respectively equivalent to the aforementioned pebble and blind transducers. More interestingly, the marble bimachines (which call "prefixes") also correspond to pebble transducers. In our proofs, we shall preferentially use the marble model to represent polyregular functions.

▶ **Theorem 2.10** ([6]). *For all $k \geq 1$, k-pebble transducers, k-pebble bimachines and k-marble bimachines compute the same class of functions. Furthermore, k-blind transducers and k-blind bimachines compute the same class of functions.*

3 From pebbles to rational series

The class of *rational series* (which are total functions $A^* \to \mathbb{N}$) is the closure of regular functions under sum, Cauchy product and Kleene star. It is well known that it can be described by *weighted automata*, furthermore this class is closed under Hadamard product (see e.g. [1, Theorem 5.5]). Let us recall the definition of these operations for $f, g : A^* \to \mathbb{N}$:

- the sum $f + g : w \mapsto f(w) + g(w)$;
- the Cauchy product $f \circledC g : w \mapsto \sum_{i=0}^{|w|} f(w[1{:}i])g(w[i{+}1{:}|w|])$;
- the Hadamard product $f \circledH g : w \mapsto f(w)g(w)$;
- if and only if $f(\varepsilon) = 0$, the Kleene star $f^* := \sum_{n \geq 0} f^n$ where $f^0 : \varepsilon \mapsto 1, u \neq \varepsilon \mapsto 0$ is neutral for Cauchy product and $f^{n+1} := f \circledC f^n$.

▶ **Example 3.1.** In Example 2.3 we have $\mathsf{nb}_{a_1,\dots,a_k} = \mathsf{nb}_{a_1} \circledH \cdots \circledH \mathsf{nb}_{a_k}$.

▶ **Example 3.2.** Let $f, g : \{a,b\}^* \to \mathbb{N}$ defined by $f(wa) = 2$ for $w \in A^*$ and $f(w) = 0$ otherwise; $g(a^n bw) = n$ for $w \in A^*$ and $g(w) = 0$ otherwise. It is easy to see that $f \circledC g(a^{n_1}b \cdots a^{n_m}b) = \sum_{i=1}^m n_i(n_i{-}1)$. Hence $2\text{-}\mathsf{powers} = \mathsf{nb}_a + f \circledC g$ (see Example 2.4).

We are now ready to state the first main results of this paper. The first one shows that polyregular functions correspond to the subclass of rational series where the use of star is disallowed (and it also corresponds to rational series whose "growth" is polynomial).

▶ **Theorem 3.3.** *Let $f : A^* \to \mathbb{N}$, the following conditions are (effectively) equivalent:*
1. *f is polyregular;*
2. *f is a rational series with polynomial growth, i.e. $f(w) = \mathcal{O}(|w|^k)$ for some $k \geq 1$;*
3. *f belongs to the closure of regular functions under sum and Cauchy product;*
4. *f belongs to the closure of regular functions under sum, Cauchy and Hadamard products.*

Proof sketch. 1 ⇔ 2 follows from [7, 6]. For 2 ⇒ 3, it is shown in [1, Exercise 1.2 of Chapter 9] that the rational series of polynomial growth can be obtained by sum and Cauchy products from the characteristic series of rational languages (which are regular functions). Having 3 ⇒ 4 is obvious. Finally for 4 ⇒ 2, it suffices to note that regular functions have polynomial growth, and this property is preserved by $+, \circledC$ and $\circledH$. ◀

Note that Hadamard products do not increase the expressive power of this class. However, removing Cauchy products gives polyblind functions, as shown in Theorem 3.4.

▶ **Theorem 3.4.** *Let $f : A^* \to \mathbb{N}$, the following conditions are equivalent:*
1. *f is polyblind;*
2. *f belongs to the closure of regular functions under sum and Hadamard product.*

Proof sketch. We first show 1 $\Rightarrow$ 2 by induction on $k \geq 1$ when f is computed by a k-blind bimachine. For $k = 1$, it is obvious. Let us describe the induction step from $k \geq 1$ to $k{+}1$. Let f be computed by a bimachine $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$, with external blind functions computed by k-blind bimachines. Given $\mathfrak{h} \in \mathfrak{H}$, let $f'_{\mathfrak{h}}$ be the function which maps $w \in A^*$ to the cardinal $|\{1 \leq i \leq |w| : \lambda(\mu(w[1{:}i{-}1]), w[i], \mu(w[i{+}1{:}|w|])) = \mathfrak{h}\}|$. Then $f'_{\mathfrak{h}}$ is a regular function and $f = \sum_{\mathfrak{h} \in \mathfrak{H}} f'_{\mathfrak{h}} \circledH \mathfrak{h}$. The result follows by induction hypothesis.

For 2 $\Rightarrow$ 1, we show that functions computed by blind bimachines are closed under sum and Hadamard product. For the sum, we use the blind transducer model to simulate successively the computation of the two terms of a sum. For the Hadamard product, we show by induction on $k \geq 1$ that if f is computed by a k-blind bimachine and g is polyblind, then $f \circledH g$ is polyblind. If $k = 1$, we transform the bimachine for f in a blind bimachine computing $f \circledH g$ by replacing each output $n \in \mathbb{N}$ by a call to a machine computing $n \times g$. If $k \geq 2$, using the notations of the previous paragraph we have $f = \sum_{\mathfrak{h} \in \mathfrak{H}} f'_{\mathfrak{h}} \circledH \mathfrak{h}$, thus $f \circledH g = \sum_{\mathfrak{h} \in \mathfrak{H}} f'_{\mathfrak{h}} \circledH (\mathfrak{h} \circledH g)$. By induction hypothesis, each $\mathfrak{h} \circledH g$ is polyblind. Hence we compute $f \circledH g$ by replacing in $\mathcal{M}$ each external function $\mathfrak{h}$ by the function $\mathfrak{h} \circledH g$. ◀

We conclude this section by recalling that polyregular (resp. polyblind) functions with unary output enjoy a "pebble minimization" property, which allows to reduce the number of nested layers depending on the growth rate of the output.

▶ **Definition 3.5.** *We say that a function $f : A^* \to \mathbb{N}$ has* growth at most k *if $f(w) = \mathcal{O}(|w|^k)$.*

▶ **Theorem 3.6** ([7, 6, 13]). *A a polyregular (resp. polyblind) function f can be computed by a k-pebble (resp. k-blind) transducer if and only it has growth at most k. Furthermore this property can be decided and the construction is effective.*

▶ Remark 3.7. The result also holds for blind transducers with non-unary outputs [13]. However, it turns out to be false for pebble transducers with non-unary outputs (unpublished work of the author of [2]).

In the next section, we complete the decidability picture by solving the membership problem from polyregular to polyblind functions.

4 Membership problem from polyregular to polyblind

In this section, we state the most technical and interesting result of this paper, which consists in deciding if a polyregular function is polyblind. We also give in Theorem 4.6 a semantical characterization of polyregular functions which are polyblind, using the notion of *repetitiveness.* Intuitively, a k-repetitive function is a function which, when given two places in a word where the same factor is repeated, cannot distinguish between the first and the second place. Hence its output only depends on the total number of repetitions of the factor.

▶ **Definition 4.1** (Repetitive function). *Let $k \geq 1$. We say that $f : A^* \to \mathbb{N}$ is* k-repetitive *if there exists $\eta \geq 1$, such that the following holds for all $\alpha, \alpha_0, u_1, \alpha_1, \dots, u_k, \alpha_k, \beta \in A^*$ and $\omega \geq 1$ multiple of η. Let $W : \mathbb{N}^k \to A^*$ defined by:*

$$W : X_1, \dots, X_k \mapsto \left(\alpha_0 \prod_{i=1}^{k} {u_i}^{\omega X_i} \alpha_i\right).$$

and let $w := W(1, \dots, 1)$. Then there exists a function $F : \mathbb{N}^k \to \mathbb{N}$ such that for all $\overline{X} := X_1, \dots, X_k \geq 3$ and $\overline{Y} := Y_1, \dots, Y_k \geq 3$, we have:

$$f(\alpha w^{2\omega-1} W(\overline{X}) w^{\omega-1} W(\overline{Y}) w^{\omega} \beta) = F(X_1 + Y_1, \dots, X_k + Y_k).$$

▶ Remark 4.2. If f is k-repetitive, f is also $(k-1)$-repetitive, by considering an empty u_k.

Let us now give a few examples in order so see when this criterion holds, or not.

▶ **Example 4.3.** The function $\mathsf{nb}_{a_1,\dots,a_k}$ (see Example 2.3) is k-repetitive for all $k \geq 1$.

▶ **Example 4.4.** If $f : A^* \to \mathbb{N}$ where $A = \{a\}$ is a singleton, then f is k-repetitive for all $k \geq 1$. Indeed, $\alpha w^{2\omega-1} W(\overline{X}) w^{\omega-1} W(\overline{Y}) w^{\omega} \beta \in A^*$ is itself a function of $X_1+Y_1, \dots, X_k+Y_k$, hence so is its image $f(\alpha w^{2\omega-1} W(\overline{X}) w^{\omega-1} W(\overline{Y}) w^{\omega})$.

▶ **Example 4.5.** For all $k \geq 2$, the function $k\text{-}\mathsf{powers} : a^{n_1}b \cdots a^{n_m}b \mapsto \sum_{i=1}^{m} n_i^k$ is not 1-repetitive. Let us choose any $\omega \geq 1$ and fix $\alpha = \beta := \varepsilon$, $u_1 := a$ and $\alpha_0 = \alpha_1 := b$, then:

$$\begin{aligned} k\text{-}\mathsf{powers}(W(X_1,Y_1)) &= k\text{-}\mathsf{powers}\left((ba^{\omega}b)^{2\omega-1} ba^{\omega X_1} b (ba^{\omega}b)^{\omega-1} ba^{\omega Y_1} b (ba^{\omega}b)^{\omega}\right) \\ &= \omega^k(4\omega-2) + \omega^k X_1^k + \omega^k Y_1^k \end{aligned}$$

which is *not* a function of $X_1 + Y_1$ for $k \geq 2$.

We are now ready to state our main result of this paper.

▶ **Theorem 4.6.** *Let $k \geq 1$ and $f : A^* \to \mathbb{N}$ be computed by a k-pebble transducer. Then f is polyblind if and only if it is k-repetitive. Furthermore, this property can be decided and one can effectively build a blind transducer for f when it exists.*

▶ Remark 4.7. By Theorem 3.6, we can even build a k-blind transducer.

Theorem 4.6 provides a tool to show that some polyregular functions are not polyblind:

▶ **Example 4.8.** By Example 4.5, the polyregular function $k\text{-}\mathsf{powers}$ is not k-repetitive for $k \geq 2$. Therefore it is not polyblind. Also note that it is computable by a k-pebble transducer, but not by an ℓ-pebble transducer for $\ell < k$ (Theorem 3.6).

An immediate consequence of Theorem 4.6 is obtained below for functions which have both a unary output alphabet and a unary input alphabet. This result was already obtained by the authors of [13] using other techniques, in an unpublished work.

▶ **Corollary 4.9.** *Polyblind and polyregular functions over a unary input alphabet coincide.*

Proof. Polyregular functions with unary input are k-repetitive by Example 4.4. ◀

5 Repetitive functions and permutable bimachines

The rest of this paper is devoted to the proof of Theorem 4.6. Since k-pebble transducers and k-marble bimachines compute the same functions (Theorem 2.10), it follows directly from Theorem 5.1 below (the notions used are defined in the next sections).

From now on, we assume that a k-marble bimachine uses the same morphism $\mu : A^* \to M$ in its main bimachine, and inductively in all its submachines computing the external functions. We do not lose any generality here, since it is always possible to get this situation by taking the product of all the morphisms used. We also assume that μ is surjective (we do no lose any generality, since it is possible to replace M by the image $\mu(M)$).

▶ **Theorem 5.1.** *Let $k \geq 2$ and $f : A^* \to \mathbb{N}$ be computed by $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ a k-marble bimachine. The following conditions are equivalent:*

1. *f is polyblind;*
2. *f is k-repetitive;*
3. *$\mathcal{M}$ is $2^{3|M|}$-permutable (see Definition 5.11) and the function f'' (built from Proposition 6.1) is polyblind.*

Furthermore this property is decidable and the construction is effective.

Let us now give the skeleton of the proof of Theorem 5.1, which is rather long and involved. The propositions on which it relies are shown in the two following sections.

Proof of Theorem 5.1. $1 \Rightarrow 2$ follows from Proposition 5.2. For $2 \Rightarrow 3$, if f is k-repetitive then $\mathcal{M}$ is $2^{3|M|}$-permutable by Proposition 5.12. Then Proposition 6.1 gives $f = f' + f''$ where f' is polyblind and f'' is computable by a $(k-1)$-marble bimachine. By Proposition 5.2, f' is $(k-1)$-repetitive. Since f is also $(k-1)$-repetitive, then $f'' = f - f'$ is also $(k-1)$-repetitive by Claim 5.3. Thus by induction hypothesis f'' is polyblind. For $3 \Rightarrow 1$, since in Proposition 6.1 we have $f = f' + f''$ where f' and f'' are polyblind, then f is (effectively) polyblind. The decidability is also obtained by induction: one has to check that $\mathcal{M}$ is $2^{3|M|}$-permutable (which is decidable) and inductively that f'' is polyblind. ◀

Now we present the tools used in this proof. We first show that polyblind functions are repetitive. Then we introduce the notion of *permutability*, and show that repetitive functions are computed by permutable marble bimachines.

5.1 Polyblind functions are repetitive

Intuitively, a blind transducer cannot distinguish between two “similar” iterations of a given factor in a word, since it cannot drop a pebble for doing so. We get the following using technical but conceptually easy pumping arguments.

▶ **Proposition 5.2.** *A polyblind function is k-repetitive for all $k \geq 1$.*

Proof idea. We first show that a regular function (computed by a bimachine without external functions) is k-repetitive for all $k \geq 1$. In this case, η is chosen as the idempotent index of the monoid used by the bimachine. Then, it is easy to conclude by noting that k-repetitiveness is preserved under sums and Hadamard products. ◀

The induction which proves Theorem 5.1 also requires the following result. Its proof is obvious since k-repetitiveness is clearly preserved under subtractions.

▷ Claim 5.3. If f, g are k-repetitive and $f - g \geq 0$, then it is k-repetitive.

5.2 Repetitive functions are computed by permutable machines

In this subsection, we show that k-marble bimachines which compute k-repetitive functions have a specific property named *permutability* (which turns out to be decidable).

Productions. We first introduce the notion of *production*. In the rest of this paper, the notation $\{\!\{\cdots\}\!\}$ represents a multiset (i.e. a set with multiplicities). If S is a set and m is a multiset, we write $m \subseteq S$ to say that each element of m belongs to S (however, there can be multiplicities in m but not in S). For instance $\{\!\{1, 1, 2, 3, 3\}\!\} \subseteq \mathbb{N}$.

▶ **Definition 5.4** (Production). *Let $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ be a k-marble bimachine, $w \in A^*$. We define the* production *of $\mathcal{M}$ on $\{\!\{i_1, \dots, i_k\}\!\} \subseteq [1{:}|w|]$ as follows if $i_1 \leq \cdots \leq i_k$:*

- *if $k = 1$, $\mathsf{prod}^w_{\mathcal{M}}(\{\!\{i_1\}\!\}) := \lambda(\mu(w[1{:}i_1{-}1]), w[i_1], \mu(w[i_1{+}1{:}|w|])) \in \mathbb{N}$;*
- *if $k \geq 2$, let $\mathfrak{h} := \lambda(\mu(w[1{:}i_k{-}1]), w[i_k], \mu(w[i_k{+}1{:}|w|])) \in \mathfrak{H}$ and $\mathcal{M}_{\mathfrak{h}}$ be the $(k{-}1)$-marble bimachine computing $\mathfrak{h}$. Then $\mathsf{prod}^w_{\mathcal{M}}(\{\!\{i_1, \cdots, i_k\}\!\}) := \mathsf{prod}^{w[1:i_k]}_{\mathcal{M}_{\mathfrak{h}}}(\{\!\{i_1, \dots, i_{k-1}\}\!\})$.*

The value $\mathsf{prod}^w_{\mathcal{M}}(\{\!\{i_1, \cdots, i_k\}\!\})$ with $i_1 \leq \cdots \leq i_k$ thus corresponds to the value output when doing a call on position i_k, then on i_{k-1}, etc. Now let $\{\!\{I_1, \dots, I_k\}\!\}$ be a multiset of sets of positions of w (i.e. for all $1 \leq i \leq k$ we have $I_i \subseteq [1{:}|w|]$). We define the production of $\mathcal{M}$ on $\{\!\{I_1, \dots, I_k\}\!\}$ as the combination of all possible productions on positions among these sets:

$$\mathsf{prod}^w_{\mathcal{M}}(\{\!\{I_1, \dots, I_k\}\!\}) := \sum_{\substack{\{\!\{i_1, \dots, i_k\}\!\} \subseteq [1:|w|] \\ \text{with } i_j \in I_j \text{ for } 1 \leq j \leq k}} \mathsf{prod}^w_{\mathcal{M}}(\{\!\{i_1, \dots, i_k\}\!\}).$$

▶ Remark 5.5. Note that we no longer have $i_1 \leq \cdots \leq i_k$.

By rewriting the sum which defines the function f from $\mathcal{M}$, we get the following.

▶ **Lemma 5.6.** *Let $\mathcal{M}$ be a k-marble bimachine computing a function $f : A^* \to \mathbb{N}$. Let $w \in A^*$ and $J_1, \dots J_n$ be a partition of $[1{:}|w|]$:*

$$f(w) = \sum_{\{\!\{I_1, \dots, I_k\}\!\} \subseteq \{J_1, \dots, J_n\}} \mathsf{prod}^w_{\mathcal{M}}(\{\!\{I_1, \dots, I_k\}\!\}).$$

Following the definition of bimachines, the production $\mathsf{prod}^w_{\mathcal{M}}(\{\!\{i_1, \cdots, i_k\}\!\})$ should only depend on $w[i_1], \dots, w[i_k]$ and of the image under μ of the factors between these positions. Now we formalize this intuition in a more general setting.

▶ **Definition 5.7** (Multicontext). *Given $x \geq 0$, an x-*multicontext *consists of two sequences of words $v_0, \cdots, v_x \in A^*$ and $u_1, \cdots, u_x \in A^*$. It is denoted $v_0[\![u_1]\!]v_1 \cdots [\![u_x]\!]v_x$.*

Let $w := v_0 u_1 \cdots u_k v_k \in A^*$. For $1 \leq i \leq k$, let $I_i \subseteq [1{:}|w|]$ be the set of positions corresponding to u_i. We define the production on the multicontext $v_0[\![u_1]\!]v_1 \cdots [\![u_k]\!]v_k$ by:

$$\mathsf{prod}_{\mathcal{M}}(v_0[\![u_1]\!]v_1 \cdots [\![u_k]\!]v_k) := \mathsf{prod}^w_{\mathcal{M}}(\{\!\{I_1, \dots, I_k\}\!\}). \tag{1}$$

As stated in Proposition-Definition 5.8, this quantity only depends on the u_i and the images of the v_i under the morphism μ of $\mathcal{M}$, which leads to a new notion of productions.

▶ **Proposition-Definition 5.8.** *Let $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ be a k-marble bimachine. Let $v_0 u_1 \cdots u_k v_k \in A^*$ and $v'_0 u_1 \cdots u_k v'_k \in A^*$ be such that $\mu(v_i) = \mu(v'_i)$ for all $0 \leq i \leq k$. Then: $\mathsf{prod}_{\mathcal{M}}(v_0[\![u_1]\!]v_1 \cdots [\![u_k]\!]v_k) = \mathsf{prod}_{\mathcal{M}}(v'_0[\![u_1]\!]v'_1 \cdots [\![u_k]\!]v'_k)$. Let $m_i := \mu(v_i) = \mu(v'_i)$, we define $\mathsf{prod}_{\mathcal{M}}(m_0[\![u_1]\!]m_1 \cdots [\![u_k]\!]m_k)$ as the previous value.*

▶ Remark 5.9. In the following, we shall directly manipulate multicontexts of the form $m_0[\![u_1]\!]m_1 \cdots [\![u_x]\!]m_x$ with $m_i \in M$ and $u_i \in A^*$. Note that an x-multicontext and a y-multicontext can be concatenated to obtain an $(x+y)$-multicontext.

We finally introduce the notion of (x, K)-*iterator*, which corresponds to an x-multicontext in which the words u_i have length at most $K \geq 0$ and have idempotent images under μ. Recall that $e \in M$ is *idempotent* if and only if $ee = e$.

▶ **Definition 5.10** (Iterator). *Let $x, K \geq 0$ and $\mu : A^* \to M$. An (x, K)-*iterator *for μ is an x-multicontext of the form $m_0 \left(\prod_{i=1}^{x} e_i[\![u_i]\!]e_i m_i\right)$ such that $m_0, \dots, m_x \in M$ and $u_1, \dots, u_x \in A^*$ are such that for all $1 \leq i \leq x$, $|u_i| \leq K$ and $e_i = \mu(u_i) \in M$ is idempotent.*

Permutable k-marble bimachines. We are now ready to state the definition of *permutability* for marble bimachines. Intuitively, this property means that, under some idempotency conditions, $\mathsf{prod}_{\mathcal{M}}(m_0[\![u_1]\!]m_1 \cdots [\![u_k]\!]m_k)$ only depends on the 1-multicontexts of each $[\![u_i]\!]$, which are $m_0\mu(u_1)\cdots m_i[\![u_i]\!]m_{i+1}\mu(u_{i+1})\cdots m_k \in M$ for $1 \leq i \leq k$. In particular, it does not depend on the relative position of the u_i nor on the m_i which separate them. Hence, it will be possible to simulate a permutable k-marble bimachine by a k-blind bimachine which can only see the 1-multicontext of one position at each time.

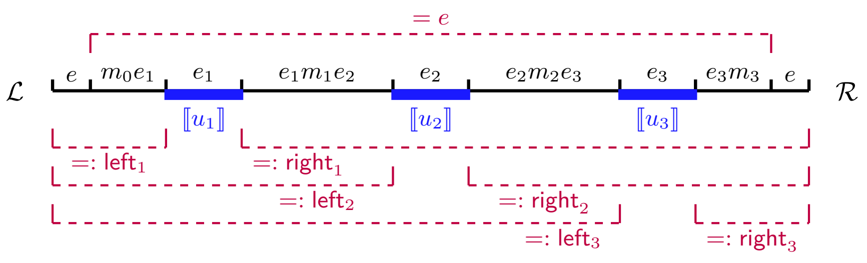
▶ **Definition 5.11.** *Let $\mathcal{M}$ be a k-marble bimachine using a surjective morphism $\mu : A^* \to M$. Let $K \geq 0$, we say that $\mathcal{M}$ is K-*permutable *if the following holds whenever $\ell + x + r = k$:*
- *for all (ℓ, K)-iterator $\mathcal{L}$ and (r, K)-iterator $\mathcal{R}$;*
- *for all (x, K)-iterator $m_0 \left(\prod_{i=1}^{x} e_i[\![u_i]\!]e_i m_i\right)$ such that $e := m_0\left(\prod_{i=1}^{x} e_i m_i\right)$ idempotent;*
- *for all $1 \leq j \leq x$, define the left and right contexts:*

$$\mathsf{left}_j := e\left(\prod_{i=1}^{j} m_{i-1}e_i\right) \quad \text{and} \quad \mathsf{right}_j := \left(\prod_{i=j}^{x} e_i m_i\right) e.$$

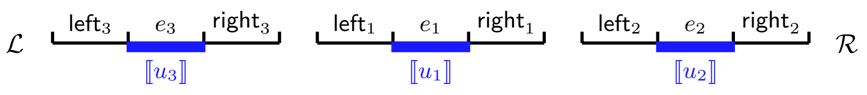
Then if σ is a permutation of $[1{:}x]$, we have:

$$\mathsf{prod}_{\mathcal{M}}\left(\mathcal{L}\; e m_0 \left(\prod_{i=1}^{x} e_i[\![u_i]\!]e_i m_i\right) e\; \mathcal{R}\right) = \mathsf{prod}_{\mathcal{M}}\left(\mathcal{L}\left(\prod_{i=1}^{x} \mathsf{left}_{\sigma(i)}\, [\![u_{\sigma(i)}]\!]\, \mathsf{right}_{\sigma(i)}\right)\mathcal{R}\right).$$

An visual description of permutability is depicted in Figure 6.



(a) Initial multicontext and definition of the left_j / right_j for $1 \leq j \leq 3$.



(b) Multicontext which separates the factors using the left_j / right_j and σ.

Figure 6 Productions which must be equal in Definition 5.11, with $x = 3$ and $\sigma = (3, 1, 2)$.

The next result follows from a technical proof based on iteration techniques (it can be understood as a kind of pumping lemma on iterators).

▶ **Proposition 5.12.** *Let $\mathcal{M}$ be a k-marble bimachine computing a k-repetitive function. Then $\mathcal{M}$ is K-permutable for all $K \geq 0$.*

Let us finally note that being K-permutable (for a fixed K) is a decidable property. Indeed, it suffices to range over all $\ell + x + r = k$ and (ℓ, K)-, (x, K)- and (r, K)-iterators (there are finitely many of them, since they correspond to bounded sequences which alternate between monoid elements and words of bounded lengths), and compute their productions.

6 From permutable bimachines to polyblind functions

The purpose of this section is to show Proposition 6.1, which allows us to perform the induction step in the proof of Theorem 5.1, by going from k to $k{-}1$ marbles.

▶ **Proposition 6.1.** *Let $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ be a $2^{3|M|}$-permutable k-marble bimachine computing a function $f : A^* \to \mathbb{N}$. One can build a polyblind function $f' : A^* \to \mathbb{N}$ and a function $f'' A^* \to \mathbb{N}$ computed by a $(k{-}1)$-marble bimachine such that $f = f' + f''$.*

Proof overview. It follows from Equation 2 and Lemma 6.14 that:

$$\begin{aligned} f &= (\mathsf{sum\text{-}dep}_{\mathcal{M}} + \mathsf{sum\text{-}ind}_{\mathcal{M}}) \circ \mathsf{forest}_{\mu} \quad \text{(see Theorem 6.5 for the definition of } \mathsf{forest}_{\mu}\text{)} \\ &= \underbrace{\mathsf{sum\text{-}ind}'_{\mathcal{M}} \circ \mathsf{forest}_{\mu}}_{=:f'} + \underbrace{(\mathsf{sum\text{-}dep}_{\mathcal{M}} + \mathsf{sum\text{-}ind}''_{\mathcal{M}}) \circ \mathsf{forest}_{\mu}}_{=:f''} \end{aligned}$$

By Lemma 6.14 one has $f'' \geq 0$. Furthermore, $\mathsf{sum\text{-}ind}'_{\mathcal{M}}$ is polyblind, hence f' is polyblind since the class of polyblind functions is closed under pre-composition by regular functions (even when the outputs are not unary, see [13]). Similarly, it follows from lemmas 6.12 and 6.14 that $\mathsf{sum\text{-}dep}_{\mathcal{M}}$ and $\mathsf{sum\text{-}ind}''_{\mathcal{M}}$ are polyregular with growth at most $k{-}1$ (see Definition 3.5), hence so is their sum and its pre-composition by a regular function [2]. Thus f'' is polyregular and has growth at most $k{-}1$. By theorems 2.10 and 3.6, it can be computed by a $(k{-}1)$-marble bimachine. ◀

Now we give the statements and the proofs of lemmas 6.12 and 6.14. An essential tool is the notion of *factorization forest*, which is recalled below.

6.1 Factorization forests

If $\mu : A^* \to M$ is a morphism into a finite monoid and $w \in A^*$, a *factorization forest* (called *μ-forest* in the following) of w is an unranked tree structure defined as follows.

▶ **Definition 6.2** (Factorization forest [15]). *Given a morphism $\mu : A^* \to M$ and $w \in A^*$, we say that $\mathcal{F}$ is a μ-forest of w if:*

- *either $\mathcal{F} = a$ and $w = a \in A$;*
- *or $\mathcal{F} = \langle \mathcal{F}_1 \rangle \cdots \langle \mathcal{F}_n \rangle$, $w = w_1 \cdots w_n$ and for all $1 \leq i \leq n$, $\mathcal{F}_i$ is a μ-forest of $w_i \in A^+$. Furthermore, if $n \geq 3$ then $\mu(w_1) = \cdots = \mu(w_n)$ is an idempotent of M.*

▶ Remark 6.3. If $\langle \mathcal{F}_1 \rangle \cdots \langle \mathcal{F}_n \rangle$ is a μ-forest, then so is $\langle \mathcal{F}_i \rangle \langle \mathcal{F}_{i+1} \rangle \dots \langle \mathcal{F}_j \rangle$ for $1 \leq i \leq j \leq n$. The empty forest ε is the unique μ-forest of the empty word ε.

We shall use the standard tree vocabulary of "height" (a leaf is a tree of height 1), "child", "sibling", "descendant" and "ancestor" (both defined in a non-strict way: a node is itself one of its ancestors/descendants), etc. We denote by $\mathsf{Nodes}(\mathcal{F})$ the set of nodes of $\mathcal{F}$. In order to simplify the statements, we identify a node $\mathsf{t} \in \mathsf{Nodes}(\mathcal{F})$ with the subtree rooted in this node. Thus $\mathsf{Nodes}(\mathcal{F})$ can also be seen as the set of subtrees of $\mathcal{F}$, and $\mathcal{F} \in \mathsf{Nodes}(\mathcal{F})$. We say that a node is *idempotent* if is has at least 3 children (see Definition 6.2).

Given $\mu : A^* \to M$, we denote by $\mathsf{Forests}_{\mu}(w)$ the set of μ-forests of $w \in A^*$. If $K \geq 0$, let $\mathsf{Forests}^{K}_{\mu}(w)$ be the μ-forests of w of height at most K. Note that $\mathsf{Forests}_{\mu}(w)$ is a set of tree structures which can also be seen as a language over $\widehat{A} := A \uplus \{\langle, \rangle\}$. Indeed, a forest of w can also be seen as "the word w with brackets" in Definition 6.2.

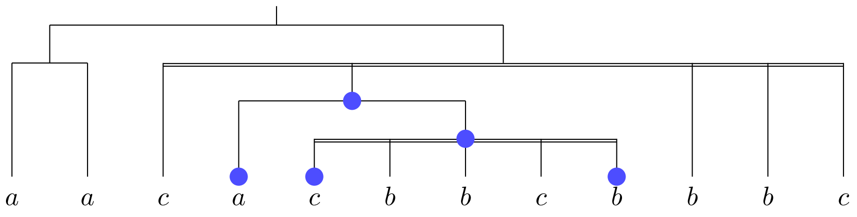


Figure 7 The μ-forest $\langle aa\rangle\langle c\langle a\langle cbbcb\rangle\rangle bbc\rangle$ on $aacacbbcbbbc$.

▶ **Example 6.4.** Let $A = \{a, b, c\}$, $M{=}(\{1, -1, 0\}, \times)$ with $\mu(a) := -1$ and $\mu(b) := \mu(c) := 0$. Then $\mathcal{F} := \langle aa\rangle\langle c\langle a\langle cbbcb\rangle\rangle bbc\rangle \in \mathsf{Forests}^5_\mu(aacacbbcbbbc)$ (we dropped the brackets around single letters for more readability) is depicted in Figure 7. Double lines are used to denote idempotent nodes (i.e. with more than 3 children).

A celebrated result states that for any word, a forest of bounded height always exist and it can be computed by a bimachine (with non-unary output alphabet), or a two-way transducer. The following theorem can also be found in [2, Lemma 6.5].

▶ **Theorem 6.5** ([15]). *Given a morphism $\mu : A^* \to M$, we have $\mathsf{Forests}^{3|M|}_\mu(w) \neq \varnothing$ for all $w \in A^*$. Furthermore, one can build a two-way transducer (with a non-unary output alphabet) which computes a total function $\mathsf{forest}_\mu : A^* \to (\widehat{A})^*, w \mapsto \mathcal{F} \in \mathsf{Forests}^{3|M|}_\mu(w)$.*

6.2 Iterable nodes and productions

We define the iterable nodes $\mathsf{Iters}(\mathcal{F}) \subseteq \mathsf{Nodes}(\mathcal{F})$ as the set of nodes which have both a left and a right sibling. Such nodes are thus exactly the middle children of idempotent nodes.

▶ **Definition 6.6.** *Let $\mathcal{F} \in \mathsf{Forests}_\mu(w)$, we define inductively the* iterable nodes *of $\mathcal{F}$:*
- *if $\mathcal{F} = a \in A$ is a leaf, $\mathsf{Iters}(\mathcal{F}) := \varnothing$;*
- *otherwise if $\mathcal{F} = \langle\mathcal{F}_1\rangle \cdots \langle\mathcal{F}_n\rangle$, then:*

$$\mathsf{Iters}(\mathcal{F}) := \{\mathcal{F}_i : 2 \leq i \leq n{-}1\} \cup \bigcup_{1\leq i\leq n} \mathsf{Iters}(\mathcal{F}_i).$$

Now we define a notion of *skeleton* which selects the right-most and left-most children.

▶ **Definition 6.7.** *Let $\mathcal{F} \in \mathsf{Forests}_\mu(w)$ and $\mathfrak{t} \in \mathsf{Nodes}(\mathcal{F})$, we define the* skeleton *of $\mathfrak{t}$ by:*
- *if $\mathfrak{t} = a \in A$ is a leaf, then $\mathsf{Skel}(\mathfrak{t}) := \{\mathfrak{t}\}$;*
- *otherwise if $\mathfrak{t} = \langle\mathcal{F}_1\rangle \cdots \langle\mathcal{F}_n\rangle$, then $\mathsf{Skel}(\mathfrak{t}) := \{\mathfrak{t}\} \cup \mathsf{Skel}(\mathcal{F}_1) \cup \mathsf{Skel}(\mathcal{F}_n)$.*

Intuitively, $\mathsf{Skel}(\mathfrak{t}) \subseteq \mathsf{Nodes}(\mathcal{F})$ contains all the descendants of $\mathfrak{t}$ except those which are descendant of a middle child. We then define the frontier of $\mathfrak{t}$, denoted $\mathsf{Fr}_\mathcal{F}(\mathfrak{t}) \subseteq \{1, \dots, |w|\}$ as the set of positions of w which belong to $\mathsf{Skel}(\mathfrak{t})$ (when seen as leaves of $\mathcal{F}$).

▶ **Example 6.8.** In Figure 7, the top-most blue node $\mathfrak{t}$ is iterable. Furthermore $\mathsf{Skel}(\mathfrak{t})$ is the set of blue nodes, $\mathsf{Fr}_\mathcal{F}(\mathfrak{t}) = \{4, 5, 9\}$ and $w[\mathsf{Fr}_\mathcal{F}(\mathfrak{t})] = acb$.

Using the frontiers, we can naturally lift the notion of productions of a k-marble bimachine from multisets of sets of positions to multisets of nodes in a forest.

▶ **Definition 6.9.** *Let $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ be a k-marble bimachine, $w \in A^*$, $\mathcal{F} \in \mathsf{Forests}_\mu(w)$ and $\mathfrak{t}_1, \dots, \mathfrak{t}_k \in \mathsf{Nodes}(\mathcal{F})$. We let $\mathsf{prod}^\mathcal{F}_\mathcal{M}(\{\!\{\mathfrak{t}_1, \dots, \mathfrak{t}_k\}\!\}) := \mathsf{prod}^w_\mathcal{M}(\{\!\{\mathsf{Fr}_\mathcal{F}(\mathfrak{t}_1), \dots, \mathsf{Fr}_\mathcal{F}(\mathfrak{t}_k)\}\!\})$.*

Using Lemma 5.6, we can recover the function f from the productions over the nodes. Lemma 6.10 roughly ranges over all possible tuples of calling positions of $\mathcal{M}$.

▶ **Lemma 6.10.** *Let $f : A^* \to \mathbb{N}$ be computed by a k-marble bimachine $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$. If $w \in A^*$ and $\mathcal{F} \in \mathsf{Forests}_\mu(w)$, we have:*

$$f(w) = \sum_{\{\!\{\mathsf{t}_1,\dots,\mathsf{t}_k\}\!\} \subseteq \mathsf{Iters}(\mathcal{F}) \cup \{\mathcal{F}\}} \mathsf{prod}^{\mathcal{F}}_{\mathcal{T}}\left(\{\!\{\mathsf{t}_1, \dots, \mathsf{t}_k\}\!\}\right).$$

Proof. It follows from [6] that for all $w \in A^*$ and $\mathcal{F} \in \mathsf{Forests}_\mu(w)$, the set of frontiers $\{\mathsf{Fr}_{\mathcal{F}}(\mathsf{t}) : \mathsf{t} \in \mathsf{Iters}(\mathcal{F}) \cup \{\mathcal{F}\}\}$ is a partition of $[1{:}|w|]$. We then apply Lemma 5.6. ◀

6.3 Dependent multisets of nodes

The multisets $\{\!\{\mathsf{t}_1, \dots, \mathsf{t}_k\}\!\} \subseteq \mathsf{Iters}(\mathcal{F}) \cup \{\mathcal{F}\}$ of Lemma 6.10 will be put into two categories. The *independent* multisets are those whose nodes are distinct and "far enough" in $\mathcal{F}$. The remaining ones are said *dependent*; their number is bounded by a polynomial of degree $k-1$.

▶ **Definition 6.11** (Independent multiset). *Let $\mu : A^* \to M$, $w \in A^*$ and $\mathcal{F} \in \mathsf{Forests}_\mu(w)$, we say that a multiset $\mathfrak{T} := \{\!\{\mathsf{t}_1, \dots, \mathsf{t}_k\}\!\} \subseteq \mathsf{Iters}(\mathcal{F})$ is* independent *if for all $1 \le i \neq j \le k$:*

- *t_i is not an ancestor of t_j;*
- *t_i is not the immediate left sibling of an ancestor of t_j;*
- *t_i is not the immediate right sibling of an ancestor of t_j.*

Note that if $\mathfrak{T}$ is independent, then $\mathcal{F} \notin \mathfrak{T}$ since it is not an iterable node. We denote by $\mathsf{Ind}^k(\mathcal{F})$ the set of independent multisets. Conversely, let $\mathsf{Dep}^k(\mathcal{F})$ be the set of multisets $\{\!\{\mathsf{t}_1, \dots, \mathsf{t}_k\}\!\} \subseteq \mathsf{Iters}(\mathcal{F}) \cup \{\mathcal{F}\}$ which are *dependent* (i.e. not independent). By Lemma 6.10, if $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ computes $f : A^* \to \mathbb{N}$ and $\mathcal{F} \in \mathsf{Forests}_\mu(w)$, then:

$$f(w) = \sum_{\mathfrak{T} \in \mathsf{Ind}^k(\mathcal{F})} \mathsf{prod}^{\mathcal{F}}_{\mathcal{M}}(\mathfrak{T}) + \sum_{\mathfrak{T} \in \mathsf{Dep}^k(\mathcal{F})} \mathsf{prod}^{\mathcal{F}}_{\mathcal{M}}(\mathfrak{T}) \tag{2}$$

The idea is now to compute these two sums separately. We begin with the second one.

▶ **Lemma 6.12.** *Given a k-marble bimachine $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$, the following function is (effectively) polyregular and has growth at most $k-1$:*

$$\mathsf{sum\text{-}dep}_{\mathcal{M}} : (\widehat{A})^* \to \mathbb{N}, \mathcal{F} \mapsto \begin{cases} \displaystyle\sum_{\mathfrak{T} \in \mathsf{Dep}^k(\mathcal{F})} \mathsf{prod}^{\mathcal{F}}_{\mathcal{M}}(\mathfrak{T}) & \text{if } \mathcal{F} \in \mathsf{Forests}^{3|M|}_\mu(w) \text{ for some } w \in A^*; \\ 0 \text{ otherwise.} \end{cases}$$

▶ Remark 6.13. For this result, we do not need to assume that $\mathcal{M}$ is permutable.

6.4 Independent multisets of nodes

In order to complete the description of f from Equation 2, it remains to treat the productions over independent multisets of nodes. When $\{\!\{\mathsf{t}_1, \dots, \mathsf{t}_k\}\!\} \in \mathsf{Ind}^k(\mathcal{F})$, all the t_i must be distinct, hence we shall denote it by a set $\{\mathsf{t}_1, \dots, \mathsf{t}_k\}$. We define the counterpart of $\mathsf{sum\text{-}dep}_{\mathcal{M}}$:

$$\mathsf{sum\text{-}ind}_{\mathcal{M}} : (\widehat{A})^* \to \mathbb{N}, \mathcal{F} \mapsto \begin{cases} \displaystyle\sum_{\mathfrak{T} \in \mathsf{Ind}^k(\mathcal{F})} \mathsf{prod}^{\mathcal{F}}_{\mathcal{M}}(\mathfrak{T}) & \text{if } \mathcal{F} \in \mathsf{Forests}^{3|M|}_\mu(w) \text{ for some } w \in A^*; \\ 0 \text{ otherwise.} \end{cases}$$

▶ **Lemma 6.14.** *Given a k-marble bimachine $\mathcal{M}$ which is $2^{3|M|}$-permutable, one can build a polyblind function* $\mathsf{sum\text{-}ind}'_{\mathcal{M}} : (\widehat{A})^* \to \mathbb{N}$ *and a polyregular function* $\mathsf{sum\text{-}ind}''_{\mathcal{M}} : (\widehat{A})^* \to \mathbb{N}$ *with growth at most $k-1$, such that* $\mathsf{sum\text{-}ind}_{\mathcal{M}} = \mathsf{sum\text{-}ind}'_{\mathcal{M}} + \mathsf{sum\text{-}ind}''_{\mathcal{M}}$.

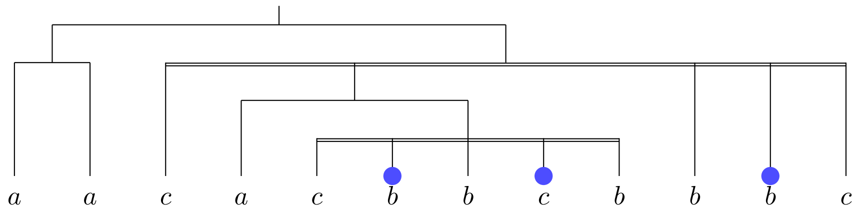
If $\mathsf{sum\text{-}ind}_{\mathcal{M}}$ was a polynomial, then $\mathsf{sum\text{-}ind}'_{\mathcal{M}}$ should roughly be its term of highest degree and $\mathsf{sum\text{-}ind}''_{\mathcal{M}}$ corresponds to a corrective term.

The rest of this section is devoted to the proof of Lemma 6.14. In order to simplify the notations, we extend a morphism μ to its μ-forests by $\mu(\mathcal{F}) := \mu(w)$ when $\mathcal{F} \in \mathsf{Forests}_\mu(w)$. Given $\mathsf{t} \in \mathsf{Nodes}(\mathcal{F})$, we denote by $\mathsf{depth}^{\mathcal{F}}(\mathsf{t})$ its depth in the tree structure $\mathcal{F}$ (the root has depth 1, and it is defined inductively as usual). Now we introduce the notion of *linearization* of $\mathsf{t} \in \mathsf{Nodes}(\mathcal{F})$, which is used to describe $w[\mathsf{Fr}_{\mathcal{F}}(\mathsf{t})]$ as a 1-multicontext.

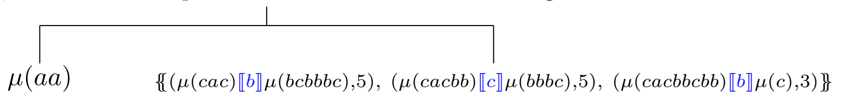
▶ **Definition 6.15** (Linearization). *Let $\mu : A^* \to M$, $w \in A^*$ and $\mathcal{F} \in \mathsf{Forests}_\mu(w)$. The* linearization *of $\mathsf{t} \in \mathsf{Nodes}(\mathcal{F})$ is a 1-multicontext $m[\![u]\!]m'$ defined by induction:*

- *if $\mathsf{t} = \mathcal{F}$ then $\mathsf{lin}^{\mathcal{F}}(\mathsf{t}) := [\![w[\mathsf{Fr}_{\mathcal{F}}(\mathcal{F})]]\!]$;*
- *otherwise $\mathcal{F} = \langle \mathcal{F}_1 \rangle \cdots \langle \mathcal{F}_n \rangle$, and $\mathsf{t} \in \mathsf{Nodes}(\mathcal{F}_i)$ for some $1 \leq i \leq n$. We define:* $\mathsf{lin}^{\mathcal{F}}(\mathsf{t}) := \mu(\mathcal{F}_1) \cdots \mu(\mathcal{F}_{i-1})\, \mathsf{lin}^{\mathcal{F}_i}(\mathsf{t}) \mu(\mathcal{F}_{i+1}) \cdots \mu(\mathcal{F}_n)$.

We finally introduce the notion of *architecture*. Intuitively, it is a simple tree which describes the positions of a set of nodes $\mathfrak{T} \in \mathsf{Ind}^k(\mathcal{F})$ in its forest $\mathcal{F}$. We build it inductively on the example depicted in Figure 8. At the root, we see that there is no node of $\mathfrak{T}$ in the left subtree, hence we replace it by its image under μ. The right subtree is an idempotent node whose leftmost and rightmost subtrees have no node in $\mathfrak{T}$. We thus replace this idempotent node by a leaf containing the multiset of the linearizations and depths of the $\mathsf{t} \in \mathfrak{T}$. Since our machine $\mathcal{M}$ is permutable, this simple information will be enough to recover $\mathsf{prod}^{\mathcal{F}}_{\mathcal{M}}(\mathfrak{T})$.



(a) In blue, a set $\mathfrak{T}$ of 3 independent nodes in the forest from Figure 7.


$\mu(aa)$
$\{\!\{(\mu(cac)[\![b]\!]\mu(bcbbbc),5),\ (\mu(cacbb)[\![c]\!]\mu(bbbc),5),\ (\mu(cacbbcbb)[\![b]\!]\mu(c),3)\}\!\}$

(b) The corresponding architecture.

Figure 8 A set of independent nodes and its architecture.

▶ **Definition 6.16** (Architecture). *Let $w \in A^*$, $\mathcal{F} \in \mathsf{Forests}_\mu(w)$ and $\mathfrak{T} \in \mathsf{Ind}^k(\mathcal{F})$. We define the* architecture *of $\mathfrak{T}$ in $\mathcal{F}$ by induction as follows:*

- *if $\mathcal{F} = \varepsilon$, then $k = 0$. We define $\mathsf{arc}^{\mathcal{F}}(\mathfrak{T}) := \varepsilon$;*
- *if $\mathcal{F} = a$, then $k = 0$. We define $\mathsf{arc}^{\mathcal{F}}(\mathfrak{T}) := \mu(a)$;*
- *otherwise $\mathcal{F} = \langle \mathcal{F}_1 \rangle \cdots \langle \mathcal{F}_n \rangle$ with $n \geq 1$:*
  - *if $k = 0$, we set $\mathsf{arc}^{\mathcal{F}}(\mathfrak{T}) = \langle \mu(\mathcal{F}) \rangle$;*
  - *else if $\mathfrak{T}_1 := \mathfrak{T} \cap \mathsf{Nodes}(\mathcal{F}_1) \neq \varnothing$, then $\mathfrak{T}_1 \in \mathsf{Ind}^{|\mathfrak{T}_1|}(\mathcal{F}_1)$ (since $\mathcal{F}_1 \notin \mathfrak{T}$ by iterability) and $\mathfrak{T} \smallsetminus \mathfrak{T}_1 \in \mathsf{Ind}^{k-|\mathfrak{T}_1|}(\langle \mathcal{F}_2 \rangle \cdots \langle \mathcal{F}_n \rangle)$ (since $\mathcal{F}_2 \notin \mathfrak{T}$ by independence). We set:* $\mathsf{arc}^{\mathcal{F}}(\mathfrak{T}) := \langle \mathsf{arc}^{\mathcal{F}_1}(\mathfrak{T}_1) \rangle\, \mathsf{arc}^{\langle \mathcal{F}_2 \rangle \cdots \langle \mathcal{F}_n \rangle}(\mathfrak{T} \smallsetminus \mathfrak{T}_1)$.

- *else if* $\mathfrak{I}_n := \mathfrak{I} \cap \mathsf{Nodes}(\mathcal{F}_n) \neq \varnothing$, *we define symmetrically:* $\mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) := \mathsf{arc}^{\langle \mathcal{F}_1 \rangle \cdots \langle \mathcal{F}_{n-1} \rangle}(\mathfrak{I} \smallsetminus \mathfrak{I}_n)\langle \mathsf{arc}^{\mathcal{F}_n}(\mathfrak{I}_n)\rangle$
- *else* $\mathfrak{I}_1 = \mathfrak{I}_n = \varnothing$ *but* $k > 0$, *thus* $n \geq 3$ *and* $\mu(\mathcal{F})$ *is idempotent. We define:* $\mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) := \langle \{\!\{ (\mathsf{lin}^{\mathcal{F}}(\mathsf{t}), \mathsf{depth}^{\mathcal{F}}(\mathsf{t})) : \mathsf{t} \in \mathfrak{I} \}\!\} \rangle$

Given a morphism, the set of architectures over bounded-height forests is finite.

▷ Claim 6.17. The set $\mathsf{Arcs}_{\mu}^{3|M|} := \{\mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) : \mathfrak{I} \in \mathsf{Ind}^k(\mathcal{F}), \mathcal{F} \in \mathsf{Forests}_{\mu}^{3|M|}(w), w \in A^*\}$ is finite, given $k \geq 0$ and a morphism $\mu : A^* \to M$.

Proof. The architectures from $\mathsf{Arcs}_{\mu}^{3|M|}$ are tree structures of height at most $3|M|$. Furthermore they have a branching bounded by $k+3$ and their leaves belong to a finite set (they are either idempotents $e \in M$, or multisets of at most k elements of the form $(m[\![u]\!]m', d)$ with $m, m' \in M$, $|u| \leq 2^{3|M|}$ and $1 \leq d \leq 3|M|$). ◁

Using the permutability of the k-marble bimachine, we show that the production over a set of independent nodes only depends on its architecture. This result enables us to define the notion of production over an architecture.

▶ **Proposition-Definition 6.18.** *Let* $\mathcal{M} = (A, M, \mu, \mathfrak{H}, \lambda)$ *be a* $2^{3|M|}$*-permutable* k*-marble bimachine. Let* $w, w' \in A^*$, $\mathcal{F} \in \mathsf{Forests}_{\mu}^{3|M|}(w)$ *and* $\mathcal{F}' \in \mathsf{Forests}_{\mu}^{3|M|}(w')$, $\mathfrak{I} \in \mathsf{Ind}^k(\mathcal{F})$ *and* $\mathfrak{I}' \in \mathsf{Ind}^k(\mathcal{F}')$ *such that* $\mathbb{A} := \mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) = \mathsf{arc}^{\mathcal{F}'}(\mathfrak{I}')$. *Then* $\mathsf{prod}_{\mathcal{M}}^{\mathcal{F}}(\mathfrak{I}) = \mathsf{prod}_{\mathcal{M}}^{\mathcal{F}'}(\mathfrak{I}')$.
We define $\mathsf{prod}_{\mathcal{M}}(\mathbb{A})$ *as the above value.*

By using the previous statements, we get for all $w \in A^*$ and $\mathcal{F} \in \mathsf{Forests}_{\mu}^{3|M|}(w)$:

$$\begin{aligned}\mathsf{sum\text{-}ind}_{\mathcal{M}}(\mathcal{F}) &= \sum_{\mathfrak{I} \in \mathsf{Ind}^k(\mathcal{F})} \mathsf{prod}_{\mathcal{M}}^{\mathcal{F}}(\mathfrak{I}) = \sum_{\mathbb{A} \in \mathsf{Arcs}_{\mu}^{3|M|}} \sum_{\substack{\mathfrak{I} \in \mathsf{Ind}^k(\mathcal{F}) \\ \mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) = \mathbb{A}}} \mathsf{prod}_{\mathcal{M}}^{\mathcal{F}}(\mathfrak{I}) \\ &= \sum_{\mathbb{A} \in \mathsf{Arcs}_{\mu}^{3|M|}} \mathsf{prod}_{\mathcal{M}}(\mathbb{A}) \times \mathsf{count}_{\mathbb{A}}(\mathcal{F})\end{aligned}$$

where $\mathsf{count}_{\mathbb{A}}(\mathcal{F}) := |\{\mathfrak{I} \in \mathsf{Ind}^k(\mathcal{F}) : \mathsf{arc}^{\mathcal{F}}(\mathfrak{I}) = \mathbb{A}\}|$. It describes the number of multisets of independent nodes which have a given architecture. Now we show how to compute this function as a sum of a polyblind function and a polyregular function with lower growth.

▶ **Lemma 6.19.** *Let* $\mu : A^* \to M$. *Given an architecture* $\mathbb{A} \in \mathsf{Arcs}_{\mu}^{3|M|}$, *one can build:*

- *a polyblind function* $\mathsf{count}'_{\mathbb{A}} : (\widehat{A})^* \to \mathbb{N}$;
- *a polyregular function* $\mathsf{count}''_{\mathbb{A}} : (\widehat{A})^* \to \mathbb{N}$ *with growth at most* $k-1$;

such that $\mathsf{count}_{\mathbb{A}}(\mathcal{F}) = \mathsf{count}'_{\mathbb{A}}(\mathcal{F}) + \mathsf{count}''_{\mathbb{A}}(\mathcal{F})$ *for all* $\mathcal{F} \in \mathsf{Forests}_{\mu}^{3|M|}(w)$ *and* $w \in A^*$.

To conclude the proof of Lemma 6.14, we define the function (which is polyblind):

$$\mathsf{sum\text{-}ind}'_{\mathcal{M}} := \sum_{\mathbb{A} \in \mathsf{Arcs}_{\mu}^{3|M|}} \mathsf{prod}_{\mathcal{M}}(\mathbb{A}) \times \mathsf{count}'_{\mathbb{A}}.$$

We define similarly the following function which is polyregular and has growth at most $k-1$:

$$\mathsf{sum\text{-}ind}''_{\mathcal{M}} = \sum_{\mathbb{A} \in \mathsf{Arcs}_{\mu}^{3|M|}} \mathsf{prod}_{\mathcal{M}}(\mathbb{A}) \times \mathsf{count}''_{\mathbb{A}}.$$

7 Outlook

This paper provides a technical solution to a seemingly difficult membership problem. This result can be interpreted both in terms of nested transducers (i.e. programs with visible or blind recursive calls) and in terms of rational series. We conjecture that the new techniques introduced here (especially the induction techniques), and the concepts of productions on words and forests, give an interesting toolbox to tackle other decision problem for transducers such as equivalence or membership issues. It could also be interesting to characterize polyblind functions as the series computed by specific weighted automata over $(\mathbb{N}, +, \times)$.

References

1 Jean Berstel and Christophe Reutenauer. *Noncommutative rational series with applications*, volume 137. Cambridge University Press, 2011.

2 Mikolaj Bojańczyk. Polyregular functions. *arXiv preprint*, 2018. `arXiv:1810.08760`.

3 Mikolaj Bojańczyk, Sandra Kiefer, and Nathan Lhote. String-to-string interpretations with polynomial-size output. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, pages 106:1–106:14, 2019. `doi:10.4230/LIPIcs.ICALP.2019.106`.

4 Michal P Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *4th International Colloquium on Automata, Languages, and Programming, ICALP 1977*, pages 135–147. Springer, 1977.

5 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular transducer expressions for regular transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 315–324. ACM, 2018.

6 Gaëtan Douéneau-Tabot. Pebble transducers with unary output. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, 2021.

7 Gaëtan Douéneau-Tabot, Emmanuel Filiot, and Paul Gastin. Register transducers are marble transducers. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, 2020.

8 Joost Engelfriet. Two-way pebble transducers for partial functions and their composition. *Acta Informatica*, 52(7-8):559–571, 2015.

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

10 Noa Globerman and David Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Computer Science*, 169(2):161–184, 1996.

11 Eitan M Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal on Computing*, 11(3):448–452, 1982.

12 Lê Thành Dung Nguyên. *Implicit automata in linear logic and categorical transducer theory.* PhD thesis, Université Paris 13, 2021.

13 Lê Thành Dung Nguyên, Camille Noûs, and Pierre Pradic. Comparison-free polyregular functions. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, 2021.

14 John C Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

15 Imre Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990. `doi:10.1016/0304-3975(90)90047-L`.