HAL open science

Comparison-free polyregular functions

Lê Thành Dũng Nguyễn, Camille Noûs, Pierre Pradic

To cite this version:

Lê Thành Dũng Nguyễn, Camille Noûs, Pierre Pradic. Comparison-free polyregular functions. International Colloquium on Automata, Languages and Programming 2021, Jul 2021, Glasgow, United Kingdom. hal-02986228v2

HAL Id: hal-02986228

https://hal.science/hal-02986228v2

Submitted on 13 May 2021 (v2), last revised 21 Feb 2023 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Comparison-free Polyregular Functions

Lê Thành Dũng (Tito) Nguyễn
Laboratoire d'informatique de Paris Nord, Villetaneuse, France

Camille Noûs
Laboratoire Cogitamus

Pierre Pradic
Department of Computer Science, University of Oxford, United Kingdom

Abstract

This paper introduces a new automata-theoretic class of string-to-string functions with polynomial growth. Several equivalent definitions are provided: a machine model which is a restricted variant of pebble transducers, and a few inductive definitions that close the class of regular functions under certain operations. Our motivation for studying this class comes from another characterization, which we merely mention here but prove elsewhere, based on a λ-calculus with a linear type system.

As their name suggests, these *comparison-free polyregular functions* form a subclass of polyregular functions; we prove that the inclusion is strict. We also show that they are incomparable with HDT0L transductions, closed under usual function composition – but not under a certain "map" combinator – and satisfy a comparison-free version of the pebble minimization theorem.

On the broader topic of polynomial growth transductions, we also consider the recently introduced layered streaming string transducers (SSTs), or equivalently k-marble transducers. We prove that a function can be obtained by composing such transducers together if and only if it is polyregular, and that k-layered SSTs (or k-marble transducers) are closed under "map" and equivalent to a corresponding notion of $(k+1)$-layered HDT0L systems.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases pebble transducers, HDT0L systems, polyregular functions

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

Acknowledgements Thanks to Mikołaj Bojańczyk and Sandra Kiefer for inspiring discussions, to Gaëtan Douéneau-Tabot and Amina Doumane for explaining some features of their work to us, to Charles Paperman for his help with bibliography and to the reviewers for their feedback.

1 Introduction

The theory of transducers (as described in the surveys [21, 29]) has traditionally dealt with devices that take as input strings of length n and output strings of length $O(n)$. However, several recent works have investigated function classes going beyond linear growth. We review three classes in this landscape below.

- *Polyregular functions* (§2.3) are thus named because they have (at most) polynomial growth and include regular functions (§2.2) (the most expressive of the traditional string-to-string transduction classes). They were defined in 2018 [4] by four equivalent computational models, one of which – the *pebble transducers* – is the specialization to strings of a tree transducer model that existed previously in the literature [28] (this specialization had been investigated earlier in [17, 14]). A subsequent work [8] gave a logical characterization based on Monadic Second-Order logic (MSO). They enjoy two nice properties:
  - *preservation of regular languages (by preimage)*: if $f : \Gamma^* \to \Sigma^*$ is polyregular and $L \subseteq \Sigma^*$ is regular, then $f^{-1}(L) \subseteq \Gamma^*$ is regular;

  - *closure under function composition*: if $f : \Gamma^* \to \Delta^*$ and $g : \Delta^* \to \Sigma^*$ are both polyregular, then so is $g \circ f : \Gamma^* \to \Sigma^*$.
- *HDT0L transductions* (§2.1) form another superclass of regular functions, whose output size may be at most exponential in the input size. They are older than polyregular functions, and we shall discuss their history in Section 2.1; suffice to say for now, they also admit various equivalent characterizations scattered in several papers [20, 22, 13]. These functions preserve regular languages by preimage, but are *not* closed under composition (the growth rate of a composition of HDT0L transductions may be a tower of exponentials).
- Very recently, the polynomially bounded HDT0L transductions (§2.3) have been characterized using two transducer models [13]. One of them, the *k-marble transducers* (where $k \in \mathbb{N}$ depends on the function to be computed), is obtained by putting a syntactic constraint on the model of *(unbounded) marble transducers* [13] which computes HDT0L transductions. But it can also be seen as a restricted variant of pebble transducers; it follows (although this is not explicitly stated in [13]) that a HDT0L transduction has polynomial growth if and only if it is polyregular. Moreover, as claimed in [13, Section 6], the functions computed by k-marble transducers are not closed under composition either, and thus form a strict subclass of polyregular functions.

A new subclass of polyregular functions In this paper, we start by proving a few results on the above classes (Section 3). For instance, we supply a proof for the aforementioned claim of [13, Section 6], and show that the polyregular functions are exactly those computable by compositions of k-marble transducers. Those complements are not particularly difficult nor surprising and are included mostly for the sake of giving a complete picture.

But our main contribution is the introduction of a new class, giving its title to the paper; as we show, it admits three equivalent definitions:

- two ways to inductively generate the class (Sections 4 and 6 respectively):
  - by closing regular functions under a certain "composition by substitution" operation;
  - by combining regular functions and a certain kind of *squaring* functions (less powerful than the squaring plus underlining functions used to characterize general poyregular functions) with usual function composition;
- a restriction on pebble transducers (Section 5) – we disallow comparing the positions of a transducer's multiple reading heads, hence the name *comparison-free polyregular functions* (henceforth abbreviated as *cfp*).

Properties By the third definition above, comparison-free polyregular functions are indeed polyregular, while the second one implies that our new class contains the regular functions and is closed under composition. (In fact, in the proof that our first definition is equivalent to the second one, most of the work goes into showing that the former enjoys closure under composition.) We rule out inclusions involving the other classes that we mentioned by proving some *separation results* (Section 8): there exist

- comparison-free polyregular functions that are not HDT0L (we take one example from [13]),
- and polynomially bounded HDT0L transductions which are not comparison-free:
  - one of our examples follows from a precise characterization of cfp functions over unary input alphabets (extending a known result for regular functions with unary inputs [10]), which we give in Section 9;
  - another example shows that unlike (poly)regular functions, cfp functions are *not* closed under a certain counterpart of the "map" operation in functional programming.

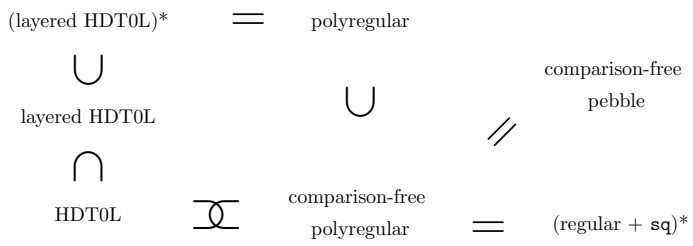


Figure 1 Summary of the known relationships between superlinear transduction classes, taking our results into account. Inclusions $\subset$ are strict, and $\not\subset\!\not\supset$ means that there is no inclusion either way. Finally C^* denotes the composition closure of the class C.

We summarize the inclusions and separations between classes that we get in Figure 1.

Finally, we show in Section 7 that the number of pebbles required to compute a function using a comparison-free transducer is related to its growth rate. The analogous result for pebble transducers was proved recently, with a whole paper dedicated to it [25]; we adapt its arguments to our setting, resulting in our longest and most technical proof. There is a similar property for k-marble transducers [13], but it is proved using very different tools.

Motivations Although this is the first proper paper to introduce comparison-free pebble transducers, we were told that they had already been considered by several colleagues (Mikołaj Bojańczyk, personal communication). But in fact, the starting point in our investigation was a characterization of regular functions using a linear λ-calculus (in the sense of linear logic) that we had previously obtained [30]; this was part of a research programme relating automata and functional programming that we initiated in [31]. As we reported in a previous version of the present paper (version 1 of the full paper archived on HAL), by tweaking a parameter in this characterization, one gets the cfp functions instead; we initially defined the latter using composition by substitution, and only later realized the connection with pebble transducers. One interesting feature of the λ-calculus characterization is that it is trivially closed under composition, and this led us to take inspiration from the category-theoretic machinery that we used in [30] for our standalone composition proof in this paper.

2 Preliminaries

Notations The set of natural numbers is $\mathbb{N} = \{0, 1, \dots\}$. We write $|w|$ for the length of a string $w \in \Sigma^*$; for $\Pi \subseteq \Sigma$, we write $|s|_\Pi$ for the number of occurrences of letters from Π in w; and for $c \in \Sigma$, we abbreviate $|w|_{\{c\}}$ as $|w|_c$. The i-th letter of $w \in \Sigma^*$ is denoted by either w_i or $w[i]$ (for $i \in \{1, \dots, |w|\}$). Given monoids M and N, $\mathrm{Hom}(M, N)$ is the set of monoid morphisms. We write ε for the empty word and $\underline{\Sigma} = \{\underline{a} \mid a \in \Sigma\}$ for a disjoint copy of the alphabet Σ made of "underlined" letters.

2.1 HDT0L transductions and streaming string transducers

L-systems were originally introduced by Lindenmayer [26] in the 1960s as a way to generate formal languages, with motivations from biology. While this language-centric view is still predominant, the idea of considering variants of L-systems as specifications for string-to-string functions – whose range are the corresponding languages – seems to be old. For instance, in a paper from 1980 [18], one can find (multi-valued) string functions defined by ET0L systems.

More recently, Ferté, Marin and Sénizergues [20] provided alternative characterizations[1] (by catenative recurrent equations and higher-order pushdown transducers of level 2) of the string-to-string functions that *HDT0L systems* can express – what we call here *HDT0L transductions*. Later work by Filiot and Reynier [22] and then by Douéneau-Tabot, Filiot and Gastin [13] – that does not build on [36, 20] – proved the equivalence with, respectively, copyful SSTs (Definition 2.3) and unbounded marble transducers (not presented here).

▶ **Definition 2.1** (following [22]). *A* HDT0L system *consists of:*
- *an input alphabet* Γ*, an output alphabet* Σ*, and a working alphabet* Δ *(all* finite*);*
- *an initial word* $d \in \Delta^*$*;*
- *for each* $c \in \Gamma$*, a* monoid morphism $h_c \in \mathrm{Hom}(\Delta^*, \Delta^*)$*;*
- *a final morphism* $h' \in \mathrm{Hom}(\Delta^*, \Sigma^*)$*.*

It defines the transduction taking $w = w_1 \dots w_n \in \Gamma^*$ *to* $h' \circ h_{w_1} \circ \dots \circ h_{w_n}(d) \in \Sigma^*$*.*

(The definition of HDT0L systems given in [36, 20] makes slightly different choices of presentation[2].) To define the equivalent model of copyful streaming string transducers, we must first introduce the notion of register assignment.

▶ **Definition 2.2.** *Fix a finite alphabet* Σ*. Let* R *and* S *be two finite sets* disjoint from Σ*; we shall consider their elements to be "register variables".*

For any word $\omega \in (\Sigma \cup R)^*$*, we write* $\omega^\dagger : (\Sigma^*)^R \to \Sigma^*$ *for the map that sends* $(u_r)_{r \in R}$ *to* ω *in which every occurrence of a register variable* $r \in R$ *is replaced by* u_r *– formally, we apply to* ω *the morphism* $(\Sigma \cup R)^* \to \Sigma^*$ *that maps* $c \in \Sigma$ *to itself and* $r \in R$ *to* u_r*.*

A register assignment[3] α *from* R *to* S *(over* Σ*) is a map* $\alpha : S \to (\Sigma \cup R)^*$*. It induces the action* $\alpha^\dagger : \vec{u} \in (\Sigma^*)^R \mapsto (\alpha(s)^\dagger(\vec{u}))_{s \in S} \in (\Sigma^*)^S$ *(which indeed goes "from* R *to* S*").*

▶ **Definition 2.3** ([22]). *A (deterministic copyful)* streaming string transducer *(SST) with input alphabet* Γ *and output alphabet* Σ *is a tuple* $\mathcal{T} = (Q, q_0, R, \delta, \vec{u}_I, F)$ *where*
- Q *is a finite set of* states *and* $q_0 \in Q$ *is the* initial state*;*
- R *is a finite set of* register variables*, that we require to be* disjoint from Σ*;*
- $\delta : Q \times \Gamma \to Q \times (R \to (\Sigma \cup R)^*)$ *is the* transition function *– we abbreviate* $\delta_{\mathrm{st}} = \pi_1 \circ \delta$ *and* $\delta_{\mathrm{reg}} = \pi_2 \circ \delta$*, where* π_i *is the projection from* $X_1 \times X_2$ *to its* i*-th component* X_i*;*
- $\vec{u}_I \in (\Sigma^*)^R$ *describes the* initial register values*;*
- $F : Q \to (\Sigma \cup R)^*$ *describes how to recombine the final values of the registers, depending on the final state, to produce the output.*

The function $\Gamma^* \to \Sigma^*$ *computed by* $\mathcal{T}$ *is*

$$w_1 \dots w_n \quad \mapsto \quad F(q_n)^\dagger \circ \delta_{\mathrm{reg}}(q_{n-1}, w_n)^\dagger \circ \dots \circ \delta_{\mathrm{reg}}(q_0, w_1)^\dagger(\vec{u}_I)$$

where the sequence of states $(q_i)_{0 \le i \le n}$ *(sometimes called the* run *of the transducer over the input word) is inductively defined, starting from the fixed initial state* q_0*, by* $q_i = \delta_{\mathrm{st}}(q_{i-1}, w_i)$*.*

---

[1] Those characterizations had previously been announced in an invited paper by Sénizergues [36]. Some other results announced in [36] are proved in [9].

[2] The family $(h_c)_{c \in \Gamma}$ is presented as a morphism $H : \Gamma^* \to \mathrm{Hom}(\Delta^*, \Delta^*)$ (whose codomain is indeed a monoid for function composition). And an initial *letter* is used instead of an initial word; this is of no consequence regarding the functions that can be expressed (proof sketch: consider $\Delta' = \Delta \cup \{x\}$ with a new letter $x \notin \Delta$, take x as the initial letter and let $h_c(x) = h_c(w)$, $h'(x) = h'(w)$).

[3] Some papers e.g. [11, 13] call register assignments *substitutions*. We avoid this name since it differs from its meaning in the context of our "composition by substitution" operation.

▶ **Example 2.4.** Let $\Sigma = \Gamma \cup \underline{\Gamma}$. We consider a SST $\mathcal{T}$ with $Q = \{q\}$, $R = \{X, Y\}$ and

$$\vec{u}_I = (\varepsilon)_{r \in R} \qquad F(q) = Y \qquad \forall c \in \Gamma,\ \delta(q, c) = (q,\ (X \mapsto cX,\ Y \mapsto \underline{c}XY))$$

If we write (v, w) for the family $(u_r)_{r \in R}$ with $u_X = v$ and $u_Y = w$, then the action of the register assignments may be described as $(X \mapsto cX,\ Y \mapsto \underline{c}XY)^\dagger(v, w) = (c \cdot v,\ \underline{c} \cdot v \cdot w)$.

Let $\mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4} \in \Gamma$. After reading $\mathtt{1234} \in \Gamma^*$, the values stored in the registers of $\mathcal{T}$ are

$$(X \mapsto \mathtt{4}X,\ Y \mapsto \underline{\mathtt{4}}XY)^\dagger \circ \ldots \circ (X \mapsto \mathtt{1}X,\ Y \mapsto \underline{\mathtt{1}}XY)^\dagger(\varepsilon, \varepsilon) = (\mathtt{4321}, \underline{\mathtt{4}}\mathtt{321}\underline{\mathtt{3}}\mathtt{21}\underline{\mathtt{2}}\mathtt{1}\underline{\mathtt{1}})$$

Since $F(q) = Y$, the function defined by $\mathcal{T}$ maps $\mathtt{1234}$ to $\underline{\mathtt{4}}\mathtt{321}\underline{\mathtt{3}}\mathtt{21}\underline{\mathtt{2}}\mathtt{1}\underline{\mathtt{1}} \in (\Gamma \cup \underline{\Gamma})^* = \Sigma^*$.

This gives us an example of HDT0L transduction $\Gamma^* \to (\Gamma \cup \underline{\Gamma})^*$, since:

▶ **Theorem 2.5** ([22]). *A function $\Gamma^* \to \Sigma^*$ can be computed by a copyful SST if and only if it can be specified by a HDT0L system.*

▶ Remark 2.6. As observed in [22, Lemma 3.3], there is a natural translation from HDT0L systems to SSTs whose range is composed precisely of the *single-state* SSTs whose transitions and final output function *do not access the letters of their output alphabet* – those are called *simple SSTs* in [13, §5.1]. This involves a kind of reversal: the initial register values correspond to the final morphisms, while the final output function corresponds to the initial word. Thus, Theorem 2.5 is essentially a state elimination result; a direct translation from SSTs to single-state SSTs has also been given by Benedikt et al. [2, Proposition 8]. However, it does *not* preserve the subclass of *copyless* SSTs (this would contradict Proposition 3.7).

The lookahead elimination theorem for macro tree transducers [19, Theorem 4.21] arguably generalizes this to trees. Indeed, while those transducers are generally presented as a top-down model, their formal definition can also be read as bottom-up register tree transducers in the style of [7, §4], and top-down lookahead corresponds to bottom-up states.

2.2 Regular functions

▶ **Definition 2.7** (Alur and Černý [1]). *A register assignment $\alpha : S \to (\Sigma \cup R)^*$ from R to S is said to be* copyless *when each $r \in R$* occurs at most once among all the strings $\alpha(s)$ for *$s \in S$, i.e. it does not occur at least twice in some $\alpha(s)$, nor at least once in $\alpha(s)$ and at least once in $\alpha(s')$ for some $s \neq s'$. (This restriction does not apply to the letters in Σ.)*

A streaming string transducer is copyless *if all the assignments in the image of its transition function are copyless. In this paper, we take computability by copyless SSTs as the definition of* regular functions *(but see Theorem 5.3 for another standard definition).*

▶ Remark 2.8. Thanks to Theorem 2.5, every regular function is a HDT0L transduction.

▶ Remark 2.9. The SST of Example 2.4 is *not* copyless: in a transition $\alpha = \delta_{\text{reg}}(q, c)$, the register X appears twice, once in $\alpha(X) = cX$ and once in $\alpha(Y) = \underline{c}XY$; in other words, its value is *duplicated* by the action $\alpha^\dagger$. In fact, it computes a function whose output size is quadratic in the input size, while regular functions have linearly bounded output.

▶ **Example 2.10** (Iterated reverse [4, p. 1]). The following single-state SST is copyless:

$$\Gamma = \Sigma \text{ with } \# \in \Sigma \qquad Q = \{q\} \qquad R = \{X, Y\} \qquad \vec{u}_I = (\varepsilon)_{r \in R} \qquad F(q) = XY$$

$$\delta(q, \#) = (q,\ (X \mapsto XY\#,\ Y \mapsto \varepsilon)) \qquad \forall c \in \Sigma \setminus \{\#\},\ \delta(q, c) = (q,\ (X \mapsto X,\ Y \mapsto cY))$$

For $u_1, \ldots, u_n \in (\Sigma \setminus \{\#\})^*$, it maps $u_1\# \ldots \#u_n$ to $\mathtt{reverse}(u_1)\# \ldots \#\mathtt{reverse}(u_n)$.

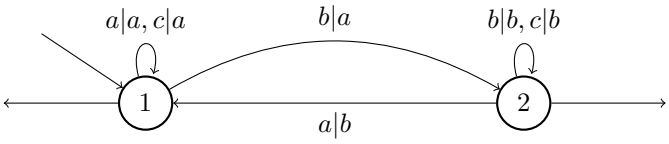


Figure 2 An example of sequential transducer.

The concrete SSTs (copyless or not) that we have seen for now are all single-state. As a source of stateful copyless SSTs, one can consider the translations of *sequential transducers*. These are usual finite automata, whose transitions additionally produce a word catenated to the end of the would-be output function. For instance, the one in Figure 2 computes the function $\{a, b, c\}^* \to \{a, b\}^*$ that replaces each c in its input by the closest non-c letter on its left (or a if no such letter exists). We do not give a detailed definition (which can be found e.g. in [33, Chapter V]) here, but for our purpose, it suffices to observe any sequential transducer can be translated into a copyless SST with the same set of states and a single register.

2.3 Polynomial growth transductions

Next, we recall one way to define Bojańczyk's *polyregular functions* [4].

▶ **Definition 2.11** ([4]). *The class of* polyregular functions *is the smallest class of string-to-string functions closed under composition containing:*

- *the functions computed by sequential transducers (for instance, the one of Figure 2);*
- *the* iterated reverse *function of Example 2.10, over any finite alphabet containing #;*
- *the* squaring with underlining *functions* $\texttt{squaring}_\Gamma : \Gamma^* \to (\Gamma \cup \underline{\Gamma})^*$*, for any finite* Γ*, illustrated by* $\texttt{squaring}_\Gamma(1234) = \underline{1}2341\underline{2}3412\underline{3}4123\underline{4}$.

As mentioned in the introduction, the intersection between the above class and HDT0L transductions has been recently characterized by Douéneau-Tabot et al. [13].

▶ **Theorem 2.12** ([13]). *Let* $f : \Gamma^* \to \Sigma^*$*. The following conditions are equivalent:*

- *f is both a polyregular function and a HDT0L transduction;*
- *f is a HDT0L transduction and has at most polynomial growth:* $f(|w|) = |w|^{O(1)}$*;*
- *there exists $k \in \mathbb{N}$ such that f is computed by some k-*layered SST*, defined below.*

(Another equivalent model, the *k-marble transducers*, was mentioned in the introduction, but we will not use it in the rest of the paper.) Those k-layered SST propose a compromise between copyful and copyless SSTs: duplication is controlled, but not outright forbidden.

▶ **Definition 2.13** ([13]). *A register assignment* $\alpha : R \to (\Sigma \cup R)^*$ *is* k-layered *(for $k \in \mathbb{N}$) with respect to a partition $R = R_0 \sqcup \ldots \sqcup R_k$ when for $0 \leq i \leq k$,*

- *for $r \in R_i$, we have $\alpha(r) \in (\Sigma \cup R_0 \cup \ldots \cup R_i)^*$;*
- *each register variable in R_i appears at most once among all the $\alpha(r)$ for $r \in R_i$ (however, those from $R_0 \sqcup \ldots \sqcup R_{i-1}$ may appear an arbitrary number of times).*

A SST is k-layered *if its registers can be partitioned in such a way that all assignments in the transitions of the SST are k-layered.*

Beware: with this definition, the registers of a k-layered SST are actually divided into $k + 1$ layers, not k. In particular, a SST is copyless if and only if it is 0-layered. (We chose this convention for backwards compatibility with [13]; see also Remark 5.4.)

For instance, the transducer of Example 2.4 is 1-layered with $R_0 = \{X\}$ and $R_1 = \{Y\}$. There also exist register assignments that cannot be made k-layered no matter the choice of partition, such as $X \mapsto XX$. Using such assignments, one can indeed build SSTs that compute functions f such that e.g. $|f(w)| = 2^{|w|}$.

▶ Remark 2.14. There is arguably an old precursor to this recent characterization of HDT0L transductions with polynomial growth by a syntactic "layering" condition: Schützenberger's theorem on polynomially bounded $\mathbb{Z}$-rational series, which dates back to the 1960s (see for instance [3, Chapter 9, Section 2] – the preface of the same book describes this theorem as "one of the most difficult results in the area"). Let us give a brief exposition.

A $\mathbb{Z}$-*rational series* $f : \Sigma^* \to \mathbb{Z}$ is a function of the form $f : w \in \Sigma^* \mapsto X^T \cdot \Phi(w) \cdot Y$ where $X, Y \in \mathbb{Z}^R$ and Φ is a morphism from Σ^* to the multiplicative monoid of R-indexed square matrices over $\mathbb{Z}$, where R is a finite set. This data (X, Φ, Y) has a clear interpretation as a "simple SST" (cf. Remark 2.6) with register set R, whose register values are integers rather than strings. Schützenberger's theorem says that any $\mathbb{Z}$-rational series f with polynomial growth (i.e. $|f(w)| = |w|^{O(1)}$ where $|\cdot|$ on the left is the absolute value) can be written as $f : w \mapsto X^T \cdot \Phi(w) \cdot Y$ where

(i) the image of Φ has a block triangular structure;
(ii) the projection of this image on each diagonal block is a finite monoid.

The first item gives us a partition of the register into layers where each layer "depends" only on the ones below them. The finiteness condition in the second item is equivalent to having bounded coefficients, which means that the register assignments within each layer are *bounded-copy*, while in a layered SST, they would be *copyless* instead – but bounded-copy SSTs are known to be equivalent to copyless SSTs (see e.g. [11]). The theorem also states a relationship between the number of blocks and the growth rate; compare this to Remark 7.2.

Via the canonical isomorphism $\{a\}^* \cong \mathbb{N}$, HDT0L transductions with unary output alphabet are the same thing as $\mathbb{N}$-*rational series*. The counterpart of Schützenberger's theorem over $\mathbb{N}$ is thus a corollary of the results of [13] on layered SSTs.

2.4 Transition monoids for streaming string transducers

To wrap up the preliminaries, let us recall some algebraic tools for working with SSTs (this technical section can be safely skipped on a first reading). Let us start by putting a monoid structure on register assignments (Definition 2.2).

▶ **Definition 2.15.** *Let $\mathcal{M}_{R,\Sigma} = R \to (\Sigma \cup R)^*$ for $R \cap \Sigma = \varnothing$. We endow it with the following composition operation, that makes it into a monoid:*

$$\alpha \bullet \beta = \alpha^{\odot} \circ \beta \quad \textit{where } \alpha^{\odot} \in \mathrm{Hom}((\Sigma \cup R)^*, (\Sigma \cup R)^*),\ \alpha^{\odot}(x) = \begin{cases} \alpha(x) & \textit{for } x \in R \\ x & \textit{for } x \in \Sigma \end{cases}$$

The monoid $\mathcal{M}_{R,\Sigma}$ thus defined is isomorphic to a submonoid of $\mathrm{Hom}((\Sigma \cup R)^*, (\Sigma \cup R)^*)$ with function composition. It admits a submonoid of *copyless* assignments.

▶ **Definition 2.16.** *We write $\mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$ for the set of all $\alpha \in \mathcal{M}_{R,\Sigma}$ such that each letter $r \in R$ occurs at most once among all the $\alpha(r')$ for $r' \in R$.*

▶ **Proposition 2.17.** *$\mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$ is a submonoid of $\mathcal{M}_{R,\Sigma}$. In other words, copylessness is preserved by composition (and the identity assignment is copyless).*

The following proposition ensures that this composition does what we expect. Recall from Definition 2.2 that $(-)^\dagger$ sends $\mathcal{M}_{R,\Sigma}$ to $(\Sigma^*)^R \to (\Sigma^*)^R$.

▶ **Proposition 2.18.** *For all $\alpha, \beta \in \mathcal{M}_{R,\Sigma}$, we have $(\alpha \bullet \beta)^\dagger = \beta^\dagger \circ \alpha^\dagger$.*

To incorporate information concerning the states of an SST, we define below a special case of the *wreath product* of transformation monoids.

▶ **Definition 2.19.** *Let M be a monoid whose multiplication is denoted by $m, m' \mapsto m \cdot m'$. We define $M \wr Q$ as the monoid whose set of elements is $Q \to Q \times M$ and whose monoid multiplication is, for $\mu, \mu' : Q \to Q \times M$,*

$$(\mu \bullet \mu') : q \mapsto (\pi_1 \circ \mu' \circ \pi_1 \circ \mu(q),\ (\pi_2 \circ \mu(q)) \cdot (\pi_2 \circ \mu' \circ \pi_1 \circ \mu(q)))$$

where $\pi_1 : Q \times M \to Q$ and $\pi_2 : Q \times M \to M$ are the projections.

For instance, if M is the trivial monoid with one element, $Q \wr M$ is isomorphic to $Q \to Q$ with *reverse* composition as the monoid multiplication: $f \bullet g = g \circ f$.

▶ **Proposition 2.20.** *Let $(Q, q_0, R, \delta, \vec{u}_I, F)$ be an SST that computes $f : \Gamma^* \to \Sigma^*$ (using the notations of Definition 2.3). For all $c \in \Gamma$, we have $\delta(-, c) \in \mathcal{M}_{R,\Sigma} \wr Q$, and the SST is copyless if and only if $\{\delta(-, c) \mid c \in \Gamma\} \subseteq \mathcal{M}^{\mathrm{cl}}_{R,\Sigma} \wr Q$. Furthermore, for all $w_1 \dots w_n \in \Gamma^*$,*

$$f(w_1 \dots w_n) \quad = \quad F(g(q_0))^\dagger(\alpha^\dagger(\vec{v})) \quad \text{where} \quad (g, \alpha) = \delta(-, w_1) \bullet \cdots \bullet \delta(-, w_n)$$

Finally, it will sometimes be useful to consider monoids of assignments over an *empty* output alphabet. This allows us to keep track of how the registers are shuffled around by transitions.

▶ **Proposition 2.21.** *Let R and Σ be disjoint finite sets. There is a monoid morphism $\mathcal{M}_{R,\Sigma} \to \mathcal{M}_{R,\varnothing}$, that sends the submonoid $\mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$ to $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$. For any Q, this extends to a morphism $\mathcal{M}_{R,\Sigma} \wr Q \to \mathcal{M}_{R,\varnothing} \wr Q$ that sends $\mathcal{M}^{\mathrm{cl}}_{R,\Sigma} \wr Q$ to $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q$. We shall use the name $\mathtt{erase}_\Sigma$ for both morphisms (R and Q being inferred from the context).*

▶ Remark 2.22. Consider an SST with a transition function δ. Let $\varphi_\delta \in \mathrm{Hom}(\Gamma^*, \mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)$ be defined by $\varphi_\delta(c) = \mathtt{erase}_\Sigma(\delta(-, c))$ for $c \in \Gamma$. The range $\varphi_\delta(\Gamma^*)$ is precisely the *substitution transition monoid (STM)* defined in [11, Section 3].

▶ **Proposition 2.23.** *For any finite R, the monoid $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ is* finite. *As a consequence, the substitution transition monoid of any* copyless *SST is finite.*

Proof idea. For all $\alpha \in \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ and $r \in R$, observe that $|\alpha(r)| \leq |R|$. ◀

3 Complements on HDT0L systems, SSTs and polyregular functions

Before embarking on the study of our new comparison-free polyregular functions, we state some minor results that consolidate our understanding of pre-existing classes.

Layered HDT0L systems Let us transpose the layering condition from SSTs to HDT0L systems. The hierarchy of models that we get corresponds *with an offset* to layered SSTs.

▶ **Definition 3.1.** *A HDT0L system $(\Gamma, \Sigma, \Delta, d, (h_c)_{c \in \Gamma}, h')$ is k-*layered *if its working alphabet can be partitioned as $\Delta = \Delta_0 \sqcup \cdots \sqcup \Delta_k$ such that, for all $c \in \Gamma$ and $i \in \{0, \dots, k\}$:*

- *for $r \in \Delta_i$, we have $h_c(r) \in (\Delta_0 \sqcup \cdots \sqcup \Delta_i)^*$;*
- *each letter in Δ_i appears at most once among all the $\alpha(r)$ for $r \in \Delta_i$ (but those in $\Delta_0 \sqcup \cdots \sqcup \Delta_{i-1}$ may appear an arbitrary number of times).*

▶ **Theorem 3.2.** *For $k \in \mathbb{N}$, a function can be computed by a k-layered SST if and only if it can be specified by a $(k+1)$-layered HDT0L system.*

In particular, regular functions correspond to 1-layered HDT0L systems.

The obvious translation from HDT0L systems to SSTs preserves 1-layeredness and produces a single-state machine, so one may sacrifice copylessness to eliminate states for SSTs.

▶ **Corollary 3.3.** *Every regular function can be computed by a single-state 1-layered SST.*

The converse to this corollary does not hold: the single-state 1-layered SST of Example 2.4 computes a function which is not regular (cf. Remark 2.9).

Polyregular functions vs layered SSTs By applying some results from [4], we can state a variant of Definition 2.11 which is a bit more convenient for us.

▶ **Proposition 3.4.** *Polyregular functions are the smallest class closed under composition that contains the regular functions and the squaring with underlining functions* $\texttt{squaring}_\Gamma$.

This allows us to show that composing HDT0L transductions with at most polynomial growth yields the polyregular functions. One direction of this equivalence is proved by encoding $\texttt{squaring}_\Gamma$ as a composition of two SSTs, one of which is Example 2.4. More precisely:

▶ **Theorem 3.5.** *Let $f : \Gamma^* \to \Sigma^*$. The following are equivalent:*

(i) *f is polyregular;*
(ii) *f can be obtained as a composition of layered SSTs;*
(iii) *f can be obtained as a composition of single-state 1-layered SSTs.*

But layered SSTs by themselves are *strictly less expressive* than polyregular functions, as we shall see later in Theorem 8.1. Therefore, as promised in the introduction:

▶ **Corollary 3.6** (claimed in [13, Section 6])**.** *Layered SSTs are* not *closed under composition.*

The importance of being stateful One interesting aspect of Theorem 3.2 is that 1-layered HDT0L systems can be seen, through Remark 2.6, as a kind of one-way transducer model for regular functions that does not use an explicit control state. This is in contrast with copyless SSTs, whose expressivity critically depends on the states (unlike copyful SSTs).

▶ **Proposition 3.7.** *The sequential (and therefore regular) function defined by the transducer of Figure 2 (Section 2.2)* cannot *be computed by a* single-state copyless SST.

In fact, the knowledgeable reader can verify that this counterexample belongs to the *first-order letter-to-letter sequential functions*, one of the weakest classical transduction classes.

Closure under map The pattern of Example 2.10 (iterated reverse) can be generalized:

▶ **Definition 3.8.** *Let $f : \Gamma^* \to \Sigma^*$ and suppose that $\# \notin \Gamma \cup \Sigma$. We define the function* $\mathbf{map}(f) : w_1\# \dots \# w_n \in (\Gamma \cup \{\#\})^* \mapsto f(w_1)\# \dots \# f(w_n) \in (\Sigma \cup \{\#\})^*$.

▶ **Proposition 3.9.** *If f is an HDT0L transduction, then so is $\mathbf{map}(f)$. For each $k \geq 1$, the functions that can be computed by k-layered HDT0L systems are also closed under* $\mathbf{map}$.

As an immediate corollary, closure under **map** holds for both regular and polyregular functions, but this was already known. In fact, $\mathtt{map}(f, [x_1, \ldots, x_n]) = [f(x_1), \ldots, f(x_n)]$ is an essential primitive in the *regular list functions* [6] and *polynomial list functions* [4, §4], two list-processing programming languages that characterize regular and polyregular functions respectively. We will come back to this point in Corollary 8.5 and the subsequent remark.

4 Composition by substitution

At last, we now introduce the class of *comparison-free polyregular functions*. The simplest way to define them is to start from the regular functions.

▶ **Definition 4.1.** *Let $f : \Gamma^* \to I^*$, and for each $i \in I$, let $g_i : \Gamma^* \to \Sigma^*$. The* composition by substitution *of f with the family $(g_i)_{i\in I}$ is the function*

$$\mathrm{CbS}(f, (g_i)_{i\in I}) \quad : \quad w \mapsto g_{i_1}(w) \ldots g_{i_k}(w) \quad \text{where } i_1 \ldots i_k = f(w)$$

That is, we first apply f to the input, then every letter i in the result of f is substituted by the image of the original input by g_i. Thus, $\mathrm{CbS}(f, (g_i)_{i\in I})$ is a function $\Gamma^* \to \Sigma^*$.

▶ **Definition 4.2.** *The smallest class of string-to-string functions closed under* CbS *and containing all regular functions is called the class of* comparison-free polyregular functions.

▶ **Example 4.3.** The following variant of "squaring with underlining" (cf. Definition 2.11) is comparison-free polyregular: $\mathtt{cfsquaring}_\Gamma : \mathtt{123} \in \Gamma^* \mapsto \underline{1}\mathtt{123}\underline{2}\mathtt{1}\underline{2}\mathtt{3}\underline{3}\mathtt{123} \in (\Gamma \cup \underline{\Gamma})^*$.

Indeed, it can be expressed as $\mathtt{cfsquaring}_\Gamma = \mathrm{CbS}(f, (g_i)_{i\in I})$ where $I = \Gamma \cup \{\#\}$, the function $f : w_1 \ldots w_n \mapsto w_1\# \ldots w_n\#$ is regular (more than that, a morphism between free monoids) and $g_\# = \mathrm{id}$, $g_c : w \mapsto \underline{c}$ for $c \in \Gamma$ are also regular. Its growth rate is quadratic, while regular functions have at most linear growth. Other examples that also require a single composition by substitution are given in Theorem 8.1.

We can already justify the latter half of the name of our new class. Using the "polynomial list functions" mentioned at the end of the previous section, we prove:

▶ **Theorem 4.4.** *Polyregular functions are closed under composition by substitution.*

▶ **Corollary 4.5.** *Every comparison-free polyregular function is, indeed, polyregular.*

Fundamentally, Definition 4.2 is inductive: it considers the functions generated from the base case of regular functions by applying compositions by substitution. The variant below with more restricted generators is sometimes convenient.

▶ **Definition 4.6.** *A string-to-string function is said to be:*

- *of* rank at most 0 *if it is regular;*
- *of* rank at most $k+1$ (for $k \in \mathbb{N}$) *if it can be written as* $\mathrm{CbS}(f, (g_i)_{i\in I})$ *where $f : \Gamma^* \to I^*$ is* regular *and each $g_i : \Gamma^* \to \Sigma^*$ is of rank at most k.*

▶ **Proposition 4.7.** *A function f is comparison-free polyregular if and only if there exists some $k \in \mathbb{N}$ such that f has rank at most k. In that case, we write $\mathrm{rk}(f)$ for the least such k and call it the* rank *of f. If $(g_i)_{i\in I}$ is a family of comparison-free polyregular functions,*

$$\mathrm{rk}(\mathrm{CbS}(f, (g_i)_{i\in I})) \leq 1 + \mathrm{rk}(f) + \max_{i\in I} \mathrm{rk}(g_i)$$

A straightforward consequence of this definition is that, just like regular functions, cfp functions are closed under *regular conditionals* and concatenation.

▶ **Proposition 4.8.** *Let $f, g : \Gamma^* \to \Sigma^*$ be comparison-free polyregular functions and $L \subseteq \Gamma^*$ be a regular language. The function that coincides with f on L and with g on $\Gamma^* \setminus L$ is cfp, and so is $w \in \Gamma^* \mapsto f(w) \cdot g(w)$; both have rank at most $\max(\mathrm{rk}(f), \mathrm{rk}(g))$.*

5 Comparison-free pebble transducers

We now characterize our function class by a machine model that will explain our choice of the adjective "comparison-free", as well as the operational meaning of the notion of rank we just defined. It is based on the *pebble transducers* first introduced for trees by Milo, Suciu and Vianu [28] and later investigated in the special case of strings by Engelfriet and Maneth [17, 14]. However, the definition using composition by substitution will remain our tool of choice to prove further properties, so the next sections do not depend on this one.

▶ **Definition 5.1.** *Let $k \in \mathbb{N}$ with $k \geq 1$. Let Γ, Σ be finite alphabets and $\triangleright, \triangleleft \notin \Gamma$.*

*A k-*pebble stack *on an input string $w \in \Gamma^*$ consists of an ordered list of p positions in the string $\triangleright w \triangleleft$ (i.e. of p integers between 1 and $|w| + 2$) for some $p \in \{1, \dots, k\}$. We therefore write $\mathrm{Stack}_k = \mathbb{N}^0 \cup \mathbb{N}^1 \cup \cdots \cup \mathbb{N}^k$, keeping in mind that given an input w, we will be interested in "legal" values bounded by $|w| + 2$.*

A comparison-free k-pebble transducer (k-CFPT) *consists of a finite set of states Q, an initial state $q_I \in Q$ and a family of transition functions*

$$Q \times (\Gamma \cup \{\triangleright, \triangleleft\})^p \to Q \times (\mathbb{N}^p \to \mathrm{Stack}_k) \times \Sigma^* \quad \textit{for } 1 \leq p \leq k$$

where the $\mathbb{N}^p$ on the left is considered as a subset of Stack_k. For a given state and given letters $(c_1, \dots, c_p) \in (\Gamma \cup \{\triangleright, \triangleleft\})^p$, the allowed values for the stack update function $\mathbb{N}^p \to \mathrm{Stack}_k$ returned by the transition function are:

(identity)	$(i_1, \dots, i_p)$	$\mapsto$	$(i_1, \dots, i_p)$	$\in$	$\mathbb{N}^p$
(move left, only allowed when $c_p \neq \triangleright$)	$(i_1, \dots, i_p)$	$\mapsto$	$(i_1, \dots, i_p - 1)$	$\in$	$\mathbb{N}^p$
(move right, only allowed when $c_p \neq \triangleleft$)	$(i_1, \dots, i_p)$	$\mapsto$	$(i_1, \dots, i_p + 1)$	$\in$	$\mathbb{N}^p$
(push, only allowed when $p \leq k - 1$)	$(i_1, \dots, i_p)$	$\mapsto$	$(i_1, \dots, i_p, 1)$	$\in$	$\mathbb{N}^{p+1}$
(pop, only allowed when $p \geq 1$)	$(i_1, \dots, i_p)$	$\mapsto$	$(i_1, \dots, i_{p-1})$	$\in$	$\mathbb{N}^{p-1}$

(Note that the codomains of all these functions are indeed subsets of Stack_k.)

The run of a CFPT over an input string $w \in \Gamma^*$ starts in the initial configuration comprising the initial state q_I, the initial k-pebble stack $(1) \in \mathbb{N}^1$, and the empty string as an initial output log. As long as the current stack is non-empty a new configuration is computed by applying the transition function to q and to $((\triangleright w \triangleleft)[i_1], \dots, (\triangleright w \triangleleft)[i_p])$ where $(i_1, \dots, i_p)$ is the current stack; the resulting stack update function is applied to $(i_1, \dots, i_p)$ to get the new stack, and the resulting output string in Σ^* is appended to the right of the current output log. If the CFPT ever terminates by producing an empty stack, the *output associated to w* is the final value of the output log.

This amounts to restricting in two ways[4] the definition of *pebble transducers* from [4, §2]:

- in a general pebble transducer, one can *compare positions*, i.e. given a stack $(i_1, \dots, i_p)$, the choice of transition can take into account whether[5] $i_j \leq i_{j'}$ (for any $1 \leq j, j' \leq p$);
- in a "push", new pebbles are initialized to the leftmost position ($\triangleright$) for a CFPT, instead of starting at the same position as the previous top of the stack (the latter would ensure the equality of two positions at some point; it is therefore an implicit comparison that we must relinquish to be truly "comparison-free").

[4] There is also an inessential difference: the definition given in [4] does not involve end markers and handles the edge case of an empty input string separately. This has no influence on the expressiveness of the transducer model. Our use of end markers follows [15, 25].

[5] One would get the same computational power, with the same stack size, by only testing whether $i_j = i_p$ for $j \leq p - 1$ as in [28] (this is also essentially what happens in the nested transducers of [25]).

This limitation is similar to (but goes a bit further than) the "invisibility" of pebbles in a transducer model introduced by Engelfriet et al. [16] (another difference, unrelated to position comparisons, is that their transducers use an unbounded number of invisible pebbles).

▶ Remark 5.2. Our definition guarantees that "out-of-bounds errors" cannot happen during the run of a comparison-free pebble transducer. The sequence of successive configurations is therefore always well-defined. But it may be infinite, that is, it may happen that the final state is never reached. Thus, a CFPT defines a *partial* function.

That said, the set of inputs for which a given pebble tree transducer does not terminate is always a regular language [28, Theorem 4.7]. This applies *a fortiori* to CFPTs. Using this, it is possible[6] to extend any partial function $f : \Gamma^* \rightharpoonup \Sigma^*$ computed by a k-CFPT into a total function $f' : \Gamma^* \to \Sigma^*$ computed by another k-CFPT for the same $k \in \mathbb{N}$, such that $f'(x) = f(x)$ for x in the domain of f and $f'(x) = \varepsilon$ otherwise. This allows us to *only consider CFPTs computing total functions* in the remainder of the paper.

A special case of particular interest is $k = 1$: the transducer has a single reading head, push and pop are always disallowed.

▶ **Theorem 5.3** ([1]). *Copyless SSTs and 1-CFPTs – which are more commonly called* two-way (deterministic) finite transducers (2DFTs) *– are equally expressive.*

Since we took copyless SSTs as our reference definition of regular functions, this means that 2DFTs characterize regular functions. But putting it this way is historically backwards: the equivalence between 2DFTs and MSO transductions came first [15] and made this class deserving of the name "regular functions" before the introduction of copyless SSTs.

▶ Remark 5.4. There are two different numbering conventions for pebble transducers. In [4, 25], 2DFTs are 1-pebble transducers, which is consistent with our choice. However, several other papers (e.g. [28, 17, 14, 16, 12]) consider that a 2DFT is a *0-pebble* transducer (likewise, in [13], 2DFTs are 0-marble transducers). This is because they think of a pebble automaton not as a restricted multi-head automaton, but as an enriched 2DFA that can drop stationary markers (called pebbles) on input positions, with a single moving head that is not a pebble.

Let us now show the equivalence with Definition 4.2. The reason for this is similar to the reason why k-pebble transducers are equivalent to the *k-nested transducers*[7] of [25], which is deemed "trivial" and left to the reader in [25, Remark 6]. But in our case, one direction (Theorem 5.6) involves an additional subtlety compared to in [25]; to take care of it, we use the fact that the languages recognized by pebble automata are regular (this is also part of [28, Theorem 4.7]) together with regular conditionals (Proposition 4.8).

▶ **Proposition 5.5.** *If f is computed by a k-CFPT, and the g_i are computed by l-CFPTs, then* $\mathrm{CbS}(f, (g_i)_{i\in I})$ *is computed by a $(k + l)$-CFPT.*

▶ **Theorem 5.6.** *If $f : \Gamma^* \to \Sigma^*$ is computed by a k-CFPT, for $k \geq 2$, then there exist a finite alphabet I, a regular function $h : \Gamma^* \to I^*$ and a family $(g_i)_{i\in I}$ computed by $(k - 1)$-CFPTs such that $f = \mathrm{CbS}(h, (g_i)_{i\in I})$.*

---

[6] Proof idea: do a first left-to-right pass to determine whether the input leads to non-termination of the original CFPT; if so, terminate immediately with an empty output; otherwise, move the first pebble back to the leftmost position and execute the original CFPT's behavior. This can be implemented by adding finitely many states, including those for a DFA recognizing non-terminating inputs.

[7] Remark: nested transducers should yield a machine-independent definition of polyregular functions as the closure of regular functions under a CbS-like operation that relies on *origin semantics* [29, §5].

▶ **Corollary 5.7.** *For all $k \in \mathbb{N}$, the functions computed by $(k+1)$-CFPTs are exactly the comparison-free polyregular functions of rank at most k.*

6 Composition of basic functions

Another possible definition of cfp functions consists in swapping out $\texttt{squaring}_\Gamma$ for some other function in Proposition 3.4:

▶ **Theorem 6.1.** *The class of comparison-free polyregular functions is the smallest class closed under usual function composition and containing both all regular functions and the functions* $\texttt{cfsquaring}_\Gamma$ *(cf. Example 4.3) for all finite alphabets* Γ.

The hard part is to show that cfp functions are closed under composition. We exploit the following combinatorial phenomenon, often applied to the study of copyless SSTs: a copyless register assignment, i.e. an element of $\mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ (cf. Section 2.4), can be specified by

- a "shape" described by an element of the *finite* monoid $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ (Proposition 2.23),
- plus finitely many "labels" in Σ^* (where Σ is the output alphabet) describing the constant factors that will be concatenated with the old register contents to give the new ones.

▶ **Proposition 6.2.** *There is a bijection*

$$\mathcal{M}^{\mathrm{cl}}_{R,\Delta} \quad \cong \quad \left\{ \left(\alpha, \vec{\ell}\right) \;\middle|\; \alpha \in \mathcal{M}^{\mathrm{cl}}_{R,\varnothing},\ \vec{\ell} \in \prod_{r \in R} (\Delta^*)^{|\alpha(r)|+1} \right\}$$

through which $\texttt{erase}_\Delta : \mathcal{M}^{\mathrm{cl}}_{R,\Delta} \to \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ *can be seen as simply removing the "labels"* $\vec{\ell}$.

Proof idea. Let $\beta \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$. For each $r \in R$, one can write $\beta(r) = w_0 r'_1 w_1 \dots r'_n w_n$ with $w_0, \dots, w_n \in \Delta^*$ and $r'_1, \dots, r'_n \in R$ such that $r'_1 \dots r'_n = \texttt{erase}_\Delta(\beta)(r) \in R^*$. ◀

This provides a clear way to represent a copyless register assignment inside the working memory of an SST: store the shape in the state and the labels in registers. Another important fact for us is that given two assignments $\beta, \beta' \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ the labels of $\beta \bullet \beta'$ can be obtained as a *copyless* recombination of the labels of β and β'.

(There is a subtlety worth mentioning here: while the set of stateful transitions $\mathcal{M}^{\mathrm{cl}}_{R,\Delta} \wr Q$ also admits a "shape + labels" representation, its monoid multiplication does *not* have this copylessness property. This prevents a naive proof of the closure under composition of copyless SSTs from working. Nevertheless, the composition of two regular functions *is* always regular, and we rely on this fact to prove Theorem 6.1.)

The rest of the proof of Theorem 6.1 is relegated to the technical appendix.

7 Rank vs asymptotic growth

Our next result is the comparison-free counterpart to recent work on polyregular functions by Lhote [25], whose proof techniques (in particular the use of Ramsey's theorem) we reuse. Compare item (ii) below to the main theorem of [25] and item (iii) – which provides yet another definition of cfp functions – to [25, Appendix A].

▶ **Theorem 7.1.** *Let $f : \Gamma^* \to \Sigma^*$ and $k \in \mathbb{N}$. The following are equivalent:*

(i) *f is comparison-free polyregular with rank at most k;*

(ii) *f is comparison-free polyregular and $|f(w)| = O(|w|^{k+1})$;*

(iii) *there exists a regular function* $g : (\{0, \dots, k\} \times \Gamma)^* \to \Sigma^*$ *such that* $f = g \circ \mathtt{cfpow}_\Gamma^{(k+1)}$, *with the following inductive definition:* $\mathtt{cfpow}_\Gamma^{(0)} : \ w \in \Gamma^* \mapsto \varepsilon \in (\varnothing \times \Gamma)^*$ *and*

$$\mathtt{cfpow}_\Gamma^{(n+1)} \ : \ \ w \mapsto (n, w_1) \cdot \mathtt{cfpow}_\Gamma^{(n)}(w) \cdot \ldots \cdot (n, w_{|w|}) \cdot \mathtt{cfpow}_\Gamma^{(n)}(w)$$

To make (ii) $\implies$ (i) more precise, if f is cfp with $\mathrm{rk}(f) \geq 1$, *then it admits a sequence of inputs* $w_0, w_1, \ldots \in \Gamma^*$ *such that* $|w_n| \to +\infty$ *and* $|f(w_n)| = \Omega(|w_n|^{\mathrm{rk}(f)+1})$.

Note that $\mathtt{cfpow}_\Gamma^{(2)}$ and $\mathtt{cfsquaring}_\Gamma$ are the same up to a bijection $\{0, 1\} \times \Gamma \cong \Gamma \cup \underline{\Gamma}$.

▶ Remark 7.2. The growth of an HDT0L transduction is also related, in a very similar way to item (ii) above, to the number of layers required in any SST that computes it [13, §5].

Some proof elements Let us present a few definitions and lemmas to give an idea of the ingredients that go into the proof. Those technical details take up the rest of this section.

Lhote's paper [25] makes a heavy use of *factorizations* of strings that depend on a morphism to a finite monoid. This is also the case for our proof, but we have found that a slightly different definition of the kind of factorization that we want works better for us.

▶ **Definition 7.3** (similar but not equivalent to [25, Definition 19]). *An* r-split *of a string* $s \in \Gamma^*$ *according to a morphism* $\varphi : \Gamma^* \to M$ *is a tuple* $(u, v_1, \ldots, v_r, w) \in (\Gamma^*)^{r+2}$ *such that:*

- $s = uv_1 \ldots v_n w$ *with* v_i *non-empty for all* $i \in \{1, \ldots, r\}$;
- $\varphi(u) = \varphi(uv_1) = \cdots = \varphi(uv_1v_r)$;
- $\varphi(w) = \varphi(v_r w) = \cdots = \varphi(v_1 \ldots v_r w)$.

▶ **Proposition 7.4** (immediate from the definition). $(u, v_1, \ldots, v_r, w)$ *is an* r*-split if and only if, for all* $i \in \{1, \ldots, r\}$, $(uv_1 \ldots v_{i-1}, v_i, v_{i+1} \ldots v_r w)$ *is a 1-split.*

The difference with the $(1, r)$-*factorizations* of [25, Definition 19] is that we have replaced the equality and idempotency requirements on $\varphi(v_1), \ldots, \varphi(v_n)$ by the "boundary conditions" involving $\varphi(u)$ and $\varphi(w)$ (actually, $(1, r + 2)$-factorizations induce r-splits). This change allows us to establish a subclaim used in the proof of Lemma 7.7 in an elementary way.

The point of r-splits is that given a split of an input string according to the morphism that sends it to the corresponding transition in a SST, we have some control over what happens to the output of the SST if we pump a middle factor in the split. Furthermore, it suffices to consider a quotient of the transition monoid which is finite when the SST is copyless (this is similar to Proposition 2.23). More precisely, we have the key lemma below, which is used pervasively throughout our proof of Theorem 7.1:

▶ **Lemma 7.5.** *Let* $f : \Gamma^* \to \Sigma^*$ *be a regular function. There exist a morphism to a* finite *monoid* $\nu_f : \Gamma^* \to \mathcal{N}(f)$ *and, for each* $c \in \Sigma$, *a set of* producing triples $P(f, c) \subseteq \mathcal{N}(f)^3$ *such that, for any* 1-split *according to* ν_f *composed of* $u, v, w \in \Gamma^*$ *– i.e.* $\nu_f(uv) = \nu_f(u)$ *and* $\nu_f(vw) = \nu_f(w)$ *– we have:*

- *if* $(\nu_f(u), \nu_f(v), \nu_f(w)) \in P(f, c)$, *then* $|f(uvw)|_c > |f(uw)|_c$;
- *otherwise (when the triple is not producing),* $|f(uvw)|_c = |f(uw)|_c$.

Furthermore, in the producing case, we get as a consequence that $\forall n \in \mathbb{N}$, $|f(uv^n w)|_c \geq n$.

▶ **Definition 7.6.** *We fix once and for all a choice of* $\mathcal{N}(f)$, ν_f *and* $P(f, c)$ *for each* $c \in \Sigma$ *and regular* $f : \Gamma^* \to \Sigma^*$. *We say that a 1-split* (u, v, w) *is* producing with respect to (f, c) *when* $(\nu_f(u), \nu_f(v), \nu_f(w)) \in P(f, c)$. *For* $\Pi \subseteq \Sigma$, *we also set* $P(f, \Pi) = \bigcup_{c \in \Pi} P(f, c)$.

Something like Lemma 7.5 (but not exactly) appears in the proof of [25, Lemma 18]. We first apply it to prove the following lemma, which is morally a counterpart to the "$k = 1$ case" of the central Dichotomy Lemma from [25], with r-splits instead of (k, r)-factorizations.

▶ **Lemma 7.7.** *Let $f : \Gamma^* \to \Sigma^*$ be regular and $\varphi : \Gamma^* \to M$ be a morphism with M finite. Suppose that $\pi \circ \varphi = \nu_f$ for some other morphism $\pi : M \to \mathcal{N}(f)$. Let $r \geq 1$ and $\Pi \subseteq \Sigma$.*

We define $L(f, \Pi, \varphi, r)$ to be the set of strings that admit an r-split $s = uv_1 \dots v_r w$ according to φ such that all the triples $(uv_1 \dots v_{i-1}, v_i, v_{i+1} \dots v_r w)$ are producing with respect to (f, Π) – let us call this a producing r-split with respect to (f, Π, φ).

Then $L(f, \Pi, \varphi, r)$ is a regular language, and $\sup\{|f(s)|_\Pi \mid s \in \Gamma^* \setminus L(f, \Pi, \varphi, r)\} < \infty$.

Our proof of the above lemma uses the proposition below, analogous to [25, Claim 20]. Its statement is a bit stronger than necessary for this purpose, but it will be reused in the proof of Theorem 8.3; as for its proof, this is where a standard Ramsey argument occurs.

▶ **Proposition 7.8.** *Let Γ be an alphabet, M be a finite monoid and $\varphi : \Gamma^* \to M$ be a morphism. There exists $N \in \mathbb{N}$ such that any string $s = uvw \in \Gamma^*$ such that $|v| \geq N$ admits an r-split $s = u'v'_1 \dots v'_r w'$ according to φ in which u is a prefix of u' and w is a suffix of w'.*

To leverage Lemma 7.7, we combine it with an elementary property of composition by substitution that does not depend on the previous technical development. (Compare the assumptions of the lemma below with the conclusion of Lemma 7.7.)

▶ **Lemma 7.9.** *Let $g : \Gamma^* \to I^*$ be a regular function and, for each $i \in I$, let $h_i : \Gamma^* \to \Sigma^*$ be comparison-free polyregular of rank at most k. Suppose that $\sup_{s \in \Gamma^*} |g(s)|_J < \infty$ where*

$$J = \begin{cases} \{i \in I \mid \mathrm{rk}(h_i) = k\} & \text{when } k \geq 1 \\ \{i \in I \mid |h_i(\Gamma^*)| = \infty\} & \text{when } k = 0 \end{cases}$$

(Morally, regular functions with finite range play the role of "comparison-free polyregular functions of rank -1".) Then $\mathrm{rk}(\mathrm{CbS}(g, (h_i)_{i \in I})) \leq k$.

The above lemma can be compared to [25, Claim 22], but it also seems to be related to the way the "nested transducer" R_{k+1} is defined in the proof of the Dichotomy Lemma in [25]: indeed, R_{k+1} can call either a k-nested subroutine or a $(k-1)$-nested one.

The remainder of the proof of Theorem 7.1 consists mainly of a rather technical induction on the rank, which we present in the appendix.

8 Separation results

Let us now demonstrate that the class of cfp functions is incomparable with the class of HDT0L transductions and is a strict subclass of polyregular functions.

▶ **Theorem 8.1.** *There exist comparison-free polyregular functions which are not HDT0L:*

- **(i)** *the function $a^n \in \{a\}^* \mapsto (a^n b)^{n+1} \in \{a, b\}^*$ for $a \neq b$;*
- **(ii)** *the function $w \in \Sigma^* \mapsto w^{|w|}$ for $|\Sigma| \geq 2$ (a simplification of Example 4.3);*
- **(iii)** (from [13, §6]) *the cfp functions that map $a^n \# w \in \Sigma^*$ to $(w\#)^n$ for $a, \# \in \Sigma$, $a \neq \#$.*

▶ Remark 8.2. The first example in [13, §5] shows that $a^n \mapsto a^{n \times n}$ is HDT0L (via the equivalent model of marble transducers), hence the necessity of $|\Sigma| \geq 2$ above. More generally, Douéneau-Tabot has shown very recently that *every polyregular function with unary output alphabet is HDT0L* [12]. So polyregular functions with unary output coincide with *polynomial growth $\mathbb{N}$-rational series* (cf. Remark 2.14), and the latter admit several algebraic characterizations in the literature (see [32] and [3, Chapter 9, Exercise 1.2]).

▶ **Theorem 8.3.** *Some HDT0L transductions are polyregular but not comparison-free:*

(i) $f : a^n \in \{a\}^* \mapsto ba^{n-1}b \dots baabab$ *(with* $f(\varepsilon) = \varepsilon$ *and* $f(a) = b$*);*
(ii) $\mathbf{map}(a^n \mapsto a^{n\times n}) : a^{n_1}\# \dots \# a^{n_k} \mapsto a^{n_1\times n_1}\# \dots \# a^{n_k\times n_k}$ *(cf. Definition 3.8).*

▶ Remark 8.4. The function $a^{n_1}\# \dots \# a^{n_k} \mapsto a^{n_1\times n_1+\dots+n_k\times n_k}$ obtained by erasing the #s in the output of $\mathbf{map}(a^n \mapsto a^{n\times n})$ is also *not* comparison-free. This result implies the second item of Theorem 8.3 by composition with the erasing morphism; we do not prove it here, but it appears in Douéneau-Tabot's aforementioned paper [12]. Therefore, according to [12], *not* every polyregular function with unary output is comparison-free.

To see why the first of the two functions in Theorem 8.3 is HDT0L, observe that it is Example 2.4 for $\Gamma = \{a\}$ (taking $b = \underline{a}$). As for the second one, combine Proposition 3.9 and the first observation in Remark 8.2.

The non-membership parts of Theorems 8.1 and 8.3 require more work. For the former, we use pumping arguments on HDT0L systems. Item (ii) of Theorem 8.3 is handled by first appealing to Theorem 7.1 to reduce to showing that $\mathbf{map}(a^n \mapsto a^{n\times n}) \neq \mathrm{CbS}(g, (h_i)_{i\in I})$ when g and all the h_i are *regular* functions; a combination of pumping and of a combinatorial argument then shows that inputs with $|I|$ occurrences of # suffice to discriminate the two sides of the inequality. This result also has the following consequence:

▶ **Corollary 8.5.** *Comparison-free polyregular functions are not closed under* **map***.*

▶ Remark 8.6. Contrast with Proposition 3.9. The discussion that follows that proposition lends some significance to the above corollary: the latter rules out the obvious conjectures for a characterization of cfp functions in the style of regular/polynomial list functions.

As for item (i) of Theorem 8.3, it concerns a function whose domain consists of words over a unary alphabet, i.e., up to isomorphism, a *sequence*. This motivates the study of such sequences, which is the subject of the next section.

9 Comparison-free polyregular sequences

From now on, we identify $\mathbb{N}$ with the set of words $\{a\}^*$ and freely speak, for instance, of cfp sequences $\mathbb{N} \to \Gamma^*$ instead of cfp functions $\{a\}^* \to \Gamma^*$. It turns out that cfp sequences admit a characterization as finite combinations of what we call *poly-pumping sequences*.

▶ **Definition 9.1.** *A* poly-pumping sequence *is a function of the form* $[\![e]\!] : \mathbb{N} \to \Sigma^*$ *where*
- e *is a* polynomial word expression *generated by* $e ::= w \mid e \cdot e' \mid e^*$ *where* $w \in \Sigma^*$*;*
- $[\![w]\!](n) = w$, $[\![e \cdot e']\!](n) = [\![e]\!](n)[\![e']\!](n)$ *and* $[\![e^*]\!](n) = ([\![e]\!](n))^n$*.*

The star-height *of a polynomial word expression is defined in the usual way.*

▶ **Theorem 9.2.** *Let* $s : \mathbb{N} \to \Sigma^*$ *and* $k \in \mathbb{N}$*. The sequence* s *is comparison-free polyregular with* $\mathrm{rk}(s) \leq k$ *if and only if there exists* $p > 0$ *such that, for any* $m < p$*, there is a polynomial word expression* e *of star-height at most* $k + 1$ *such that* $\forall n \in \mathbb{N},\ s((n + 1)p + m) = [\![e]\!](n)$*.*

In short, the cfp sequences are exactly the *ultimately periodic combinations* of poly-pumping sequences. Our proof strategy is an induction on k.

The base case $k = 0$ says that regular sequences are ultimately periodic combinations of *pumping sequences* $n \mapsto u_0(v_1)^n \dots (v_l)^n u_l$. An essentially equivalent result is stated with a proof sketch using 2DFTs in [10, p. 90]; we propose an alternative proof using copyless SSTs. (Non-deterministic two-way transducers (2NFTs) taking unary inputs have also been studied [23]; furthermore, the notion of "k-iterative language" that appears in a pumping lemma for general 2NFTs [35] is related to the shape of the above pumping sequences.)

To make the inductive step go through, it is enough to synchronize the periods of the different poly-pumping sequences involved and to observe that $\mathrm{CbS}([\![e]\!], ([\![e'_i]\!])_{i\in I})$ is realized by an expression obtained by substituting the e'_i for i in e.

Coming back to Theorem 8.3, we show that $a^n \mapsto ba^{n-1}b\dots bab$ is not comparison-free polyregular by proving that its subsequences are *not* poly-pumping: for every poly-pumping sequence $s : \mathbb{N} \to \{a, b\}^*$, there is a uniform bound on the number of distinct contiguous subwords of the shape $baa\dots ab$ occuring in each $s(n)$ for $n \in \mathbb{N}$. Another consequence of Theorem 9.2 that we establish by induction over expressions contrasts with Corollary 8.5:

▶ **Corollary 9.3.** *If $f : \Gamma^* \to \Sigma^*$ and $s : \mathbb{N} \to (\Gamma \cup \{\#\})^*$ are cfp, so is $\mathbf{map}(f) \circ s$.*

10 Further topics

Functional programming We mentioned in the introduction a forthcoming characterization of cfp functions using Church-encoded strings in a λ-calculus with linear types, in the vein of our previous results [31, 30]. Meanwhile, Corollary 8.5 could be understood as negative result in the search for another kind of functional programming characterization (cf. Remark 8.6).

It is also worth noting that the copying discipline of layered SSTs is very similar to what happens in the *parsimonious λ-calculus* [27]: a datum of type $!\tau$ cannot be duplicated into two copies of the same type $!\tau$, but it may yield an arbitrary number of copies of type τ without the modality '!'. Since the function classes defined following the methodology of [31, 30] are automatically closed under composition, Theorem 3.5 leads us to conjecture that polyregular functions can be characterized in a variant of the parsimonious λ-calculus.

First-order interpretations As we already said, regular and polyregular functions both admit logical characterizations using Monadic Second-Order Logic [15, 8]. The basic conceit behind these definitions is that a string w may be regarded as a finite model $\mathfrak{M}(w)$ over a signature containing the order relation $\leq$ on positions and predicates encoding their labeling.

The classes obtained by replacing MSO with first-order logic (FO) are to (poly)regular functions what star-free languages are to regular languages, see [11, 4]. We expect that in the same way, replacing regular functions (i.e. MSO transductions) by FO transductions in Definition 4.2 and Theorem 6.1 results in the same class in both cases, which would then be the natural FO counterpart of cfp functions. Furthermore, we believe it can be defined logically. Given a finite model $\mathfrak{U} = (U, R, \dots)$, we write $\mathfrak{U}^k$ for the kth power $(U^k, R_1, \dots, R_k, \dots)$ where $R_i(x_1, \dots, x_m)$ of arity m is defined as $R(\pi_i(x_1), \dots, \pi_i(x_m))$ for $1 \leq i \leq k$.

▶ **Conjecture 10.1.** *A function $f : \Gamma^* \to \Sigma^*$ is "FO comparison-free polyregular" if and only if there exists $k \in \mathbb{N}$ and a one-dimensional FO interpretation φ such that for every $w \in \Gamma^*$ with $|w| \geq 2$, there is an isomorphism of structures $\mathfrak{M}(f(w)) \simeq \varphi\left(\mathfrak{M}(w)^k\right)$.*

On an intuitive level, this seems to capture the inability to compare the positions of two heads of comparison-free pebble transducers. However, as mentioned to us by M. Bojańczyk, the naive transposition of this conjecture to MSO fails because the direct product, generalized to Henkin structures, does not preserve *standard* second-order models.

Integer sequences Recall from Remarks 8.2 and 8.4 that for unary outputs, polyregular and layered HDT0L transductions coincide, but comparison-free polyregular functions form a strictly smaller class (those results come from [12]). If we also restrict to unary inputs – in other words, if we consider sequences $\mathbb{N} \to \mathbb{N}$ – then we are fairly confident at this stage that the three classes collapse to a single one, and that this can be shown by routine methods:

▷ Claim 10.2. The classes of polyregular, comparison-free polyregular and layered HDT0L functions coincide on sequences of natural numbers.

Note that we already have a description of cfp integer sequences by specializing Theorem 9.2.

Membership and equivalence We presented comparison-free polyregular functions as a strict subclass of polyregular functions. This leads to a natural *membership problem*, for which partial results were recently obtained by Douéneau-Tabot [12]:

▷ Problem 10.3. Is there an algorithm taking as input a (code for a) pebble transducer which decides whether the corresponding function $\Sigma^* \to \Gamma^*$ is comparison-free or not?

There are many similar problems of interest on the frontier between comparison-free and general polyregular functions. We hope that investigating such issues may also lead to machine/syntax-free characterizations of the containment between the two classes.

Finally, a major open problem on polyregular functions is the *equivalence problem*:

▷ Problem 10.4. Is there an algorithm taking as input two pebble transducers which decides whether they compute the same function?

Interestingly, a positive answer is known for HDT0L transductions. There is an short proof using Hilbert's basis theorem [24], which is now understood to be an example of a general approach using polynomial grammars (see e.g. [2, 5]). One could hope that a restriction to comparison-free pebble transducers also puts the equivalence problem within reach of known tools. Unfortunately, the extended polynomial grammars that would serve as the natural target for a reduction from 2-CFPT equivalence already have an undecidable zeroness problem (this was shown recently by Schmude [34]). This does *not* extend, however, to an undecidability proof for the CFPT equivalence problem, so the latter is still open.

References

1. Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.1.
2. Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. Polynomial automata: Zeroness and applications. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, Reykjavik, Iceland, June 2017. IEEE. doi:10.1109/LICS.2017.8005101.
3. Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*, volume 137 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, October 2010.
4. Mikołaj Bojańczyk. Polyregular functions, 2018. arXiv:1810.08760.
5. Mikołaj Bojańczyk. The Hilbert method for transducer equivalence. *ACM SIGLOG News*, 6(1):5–17, 2019. doi:10.1145/3313909.3313911.
6. Mikołaj Bojańczyk, 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 '18*, pages 125–134, Oxford, United Kingdom, 2018. ACM Press. doi:10.1145/3209108.3209163.
7. Mikołaj Bojańczyk and Amina Doumane. First-order tree-to-tree functions. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany (online conference), July 8-11, 2020*, pages 252–265. ACM, 2020. doi:10.1145/3373718.3394785.

8 Mikołaj Bojańczyk, 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)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 106:1–106:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.106.

9 Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michał Pilipczuk, and Géraud Sénizergues. On polynomial recursive sequences. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 117:1–117:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.117.

10 Christian Choffrut. Sequences of words defined by two-way transducers. *Theoretical Computer Science*, 658:85–96, 2017. doi:10.1016/j.tcs.2016.05.004.

11 Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. Aperiodic String Transducers. *International Journal of Foundations of Computer Science*, 29(05):801–824, August 2018. doi:10.1142/S0129054118420054.

12 Gaëtan Douéneau-Tabot. Pebble transducers with unary output, 2021. arXiv:2104.14019.

13 Gaëtan Douéneau-Tabot, Emmanuel Filiot, and Paul Gastin. Register Transducers Are Marble Transducers. In Javier Esparza and Daniel Kráľ, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.29.

14 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.

15 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, April 2001. doi:10.1145/371316.371512.

16 Joost Engelfriet, Hendrik Jan Hoogeboom, and Bart Samwel. XML navigation and transformation by tree-walking automata and transducers with visible and invisible pebbles. *Theoretical Computer Science*, 850:40–97, January 2021. doi:10.1016/j.tcs.2020.10.030.

17 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.

18 Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences*, 20(2):150–202, 1980. doi:10.1016/0022-0000(80)90058-6.

19 Joost Engelfriet and Heiko Vogler. Macro tree transducers. *Journal of Computer and System Sciences*, 31(1):71–146, 1985. doi:10.1016/0022-0000(85)90066-2.

20 Julien Ferté, Nathalie Marin, and Géraud Sénizergues. Word-Mappings of Level 2. *Theory of Computing Systems*, 54(1):111–148, January 2014. doi:10.1007/s00224-013-9489-5.

21 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, Logic and Algebra for Functions of Finite Words. *ACM SIGLOG News*, 3(3):4–19, August 2016. doi:10.1145/2984450.2984453.

22 Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. *Fundamenta Informaticae*, 178(1-2):59–76, January 2021. doi:10.3233/FI-2021-1998.

23 Bruno Guillon. Input- or output-unary sweeping transducers are weaker than their 2-way counterparts. *RAIRO – Theoretical Informatics and Applications*, 50(4):275–294, 2016. doi:10.1051/ita/2016028.

24 Juha Honkala. A short solution for the HDT0L sequence equivalence problem. *Theoretical Computer Science*, 244(1-2):267–270, 2000. doi:10.1016/S0304-3975(00)00158-4.

25 Nathan Lhote. Pebble minimization of polyregular functions. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 703–712. ACM, 2020. doi:10.1145/3373718.3394804.

26 Aristid Lindenmayer. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, March 1968. doi:10.1016/0022-5193(68)90080-5.

27 Damiano Mazza. Simple Parsimonious Types and Logarithmic Space. In *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, pages 24–40, 2015. doi:10.4230/LIPIcs.CSL.2015.24.

28 Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *Journal of Computer and System Sciences*, 66(1):66–97, 2003. Journal version of a PODS 2000 paper. doi:10.1016/S0022-0000(02)00030-2.

29 Anca Muscholl and Gabriele Puppis. The Many Facets of String Transducers. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.2.

30 Lê Thành Dũng Nguyễn, Camille Noûs, and Pierre Pradic. Implicit automata in typed λ-calculi II: streaming transducers vs categorical semantics, 2020. arXiv:2008.01050.

31 Lê Thành Dũng Nguyễn and Pierre Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.135.

32 Christophe Reutenauer. Sur les séries associées à certains systèmes de Lindenmayer. *Theoretical Computer Science*, 9:363–375, 1979. doi:10.1016/0304-3975(79)90036-7.

33 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. Translated by Reuben Thomas. doi:10.1017/CBO9781139195218.

34 Janusz Schmude. On polynomial grammars extended with substitution, 2021. arXiv:2102.08705.

35 Tim Smith. A pumping lemma for two-way finite transducers. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 523–534. Springer, 2014. doi:10.1007/978-3-662-44522-8_44.

36 Géraud Sénizergues. Sequences of level 1, 2, 3, ..., k , .. In Volker Diekert, Mikhail V. Volkov, and Andrei Voronkov, editors, *Computer Science - Theory and Applications, Second International Symposium on Computer Science in Russia, CSR 2007, Ekaterinburg, Russia, September 3-7, 2007, Proceedings*, volume 4649 of *Lecture Notes in Computer Science*, pages 24–32. Springer, 2007. doi:10.1007/978-3-540-74510-5_6.

Contents

1 Introduction **1**

2 Preliminaries **3**
2.1 HDT0L transductions and streaming string transducers . . . 3
2.2 Regular functions . . . 5
2.3 Polynomial growth transductions . . . 6
2.4 Transition monoids for streaming string transducers . . . 7

3 Complements on HDT0L systems, SSTs and polyregular functions **8**

4 Composition by substitution **10**

5 Comparison-free pebble transducers **11**

6 Composition of basic functions **13**

7 Rank vs asymptotic growth **13**

8 Separation results **15**

9 Comparison-free polyregular sequences **16**

10 Further topics **17**

A Details for Remark 2.6 **22**

B Proofs for §3 (HDT0L systems, SSTs & polyregular functions) **22**
B.1 Proof of Theorem 3.2 . . . 22
B.2 Proof of Corollary 3.3 . . . 24
B.3 Proof of Proposition 3.7 . . . 24
B.4 Proof of Proposition 3.4 . . . 25
B.5 Proof of Theorem 3.5 . . . 25
B.6 Proof of Proposition 3.9 . . . 26

C Proofs for §4 (composition by substitution) **26**
C.1 Proof of Theorem 4.4 . . . 26
C.2 Proof of Proposition 4.7 . . . 27
C.3 Proof of Proposition 4.8 . . . 27

D Proofs for §5 (comparison-free pebble transducers) **28**
D.1 Proof of Proposition 5.5 . . . 28
D.2 Proof of Theorem 5.6 . . . 29
D.3 Proof of Corollary 5.7 . . . 29

E Closure under composition **29**

F A lower bound on growth from the rank **33**
F.1 Proofs for the lemmas in Section 7 . . . 34
F.2 Wrapping up the proof of Theorem F.1 . . . 37

G Proofs of Theorems 6.1 and 7.1 **41**

H Comparison-free polyregular sequences **43**
H.1 Proof of Theorem 9.2 43
H.2 Proof of Corollary 9.3 44

I Separation results **46**
I.1 Proof of Theorem 8.1 46
I.2 Proof of Theorem 8.3 47
I.2.1 Proof of Theorem 8.3 item (i) 47
I.3 Proof of Theorem 8.3 item (ii) 48

A Details for Remark 2.6

We recall the "natural" translation of HDT0L systems into single-state SSTs, which is relevant to some proofs in Section 3. Let $(\Gamma, \Sigma, \Delta, d, (h_c)_{c\in\Gamma}, h')$ be a HDT0L system. It is equivalent to the SST specified by the following data:

- a singleton set of states: $Q = \{q\}$;
- the working alphabet as the set of registers: $R = \Delta$ (minor technicality: if $\Delta \cap \Sigma \neq \varnothing$, one should take R to be a copy of Δ that is disjoint from Σ);
- $h_c \in \mathrm{Hom}(\Delta^*, \Delta^*) \cong (\Delta \to \Delta^*) \subseteq (\Delta \to (\Sigma \cup \Delta)^*)$ as the register assignment associated to an input letter $c \in \Gamma$ – in other words, the transition function is $\delta : (q, c) \mapsto (q, (h_c)_{\restriction\Delta})$;
- $(h'(r))_{r\in\Delta} \in (\Sigma^*)^R$ as the initial register values;
- $F : q \mapsto d$ as the final output function ($d \in \Delta^* \subseteq (\Sigma \cup \Delta)^*$).

The cases of the transition and output functions involve a codomain extension from Δ^* to $(\Sigma \cup \Delta)^*$. This reflects the intuition that a HDT0L system is the same thing as a single-state SST that "cannot access the output alphabet" (except in the initial register contents).

To prove the equivalence, the key observation is that h_c is turned into $\delta(-, c)$ by a *morphism* from $\mathrm{Hom}(\Delta^*, \Delta^*)$ to $\mathcal{M}_{\Delta,\varnothing}\wr\{q\} \subset \mathcal{M}_{\Delta,\Sigma}\wr\{q\}$, using the notations from Section 2.4. We leave the details to the reader.

B Proofs for §3 (HDT0L systems, SSTs & polyregular functions)

B.1 Proof of Theorem 3.2

▶ **Theorem 3.2.** *For $k \in \mathbb{N}$, a function can be computed by a k-layered SST if and only if it can be specified by a $(k+1)$-layered HDT0L system.*

In particular, regular functions correspond to 1-layered HDT0L systems.

Proof of $(\Rightarrow)$**.** The translation from SSTs to HDT0L systems given by [22, Lemma 3.5] turns out to work. It is also formulated in terms of "simple SSTs" (isomorphic to HDT0L systems, cf. Remark 2.6) in [13, §5.1], where the authors remark that "this construction does not preserve copylessness nor k-layeredness": indeed, what we show is that it increments the number of layers by one! For the sake of clarity, we give an alternative presentation that decomposes it into two steps.

Let Γ be the input alphabet and Σ be the output alphabet. Let $\mathcal{T}$ be a SST with a *k-layered* set of register variables $R = R_0 \sqcup \cdots \sqcup R_k$. First, we build a *$(k+1)$-layered* SST $\mathcal{T}'$ that computes the same function, with the set of registers

$$R' = \underline{\Sigma} \cup R = R'_0 \sqcup \cdots \sqcup R'_{k+1} \qquad R'_0 = \underline{\Sigma} \qquad \forall i \in \{1, \dots, k+1\},\ R'_i = R_{i-1}$$

assuming $\underline{\Sigma} \cap R = \varnothing$, and whose register assignments are *without fresh letters*: the range of every $\alpha' : R' \to (\Sigma \cup R')^*$ is included in R'^*, which allows us to write $\alpha' : R' \to R'^*$. This already brings us closer to the definition of HDT0L systems, since $(R \to R^*) \cong \mathrm{Hom}(R^*, R^*)$. Similarly, we will ensure that the range of the output function of $\mathcal{T}'$ is included in R'.

Let $\mathtt{underline}_\Sigma \in \mathrm{Hom}((\Sigma \cup R)^*, (\underline{\Sigma} \cup R)^*)$ be defined in the expected way, and note that its codomain is equal to R'^*. We specify $\mathcal{T}'$ as follows (and leave it to the reader to check that this works):

- the state space Q, initial state and state transitions are the same as those of $\mathcal{T}$;
- the initial value of $r' \in R'$ is the same as for $\mathcal{T}$ if $r' \in R$, or the single letter c if $r' = \underline{c} \in \underline{\Sigma}$;
- every assignment $\alpha : R \to (\Sigma \cup R)$ that appears in some transition of $\mathcal{T}$ becomes, in $\mathcal{T}'$,

$$\alpha' : R'^* \to R'^* \qquad \alpha' : \underline{c} \in \underline{\Sigma} \mapsto \underline{c} \qquad \alpha' : r \in R \mapsto \mathtt{underline}_\Sigma(\alpha(r))$$

- its output function is $F' = \mathtt{underline}_\Sigma \circ F$ where $F : Q \to (\Sigma \cup R)^*$ is the output function of $\mathcal{T}$.

Thus, the idea is to store a copy of $c \in \Sigma$ in the register $\underline{c}$. Since this register may feed in a copyful way all other registers (in a SST, there are no restrictions on the use of output alphabet letters), it must sit at the lowest layer, hence $R'_0 = \underline{\Sigma}$ and the resulting offset of one layer.

Next, we turn $\mathcal{T}'$ into an equivalent HDT0L system with $(k+1)$-layered working alphabet

$$\Delta = R' \times Q = \Delta_0 \sqcup \ldots \sqcup \Delta_{k+1} \qquad \forall i \in \{0, \ldots, k+1\},\ \Delta_i = R'_i \times Q$$

For $q \in Q$, let $\mathtt{pair}_q \in \mathrm{Hom}(R'^*, \Delta^*)$ be such that $\mathtt{pair}_q(r') = (r', q)$ for $r' \in R'$.

Let $Q = \{q^{(1)}, \ldots, q^{(n)}\}$ be the states of $\mathcal{T}'$ (which are also those of $\mathcal{T}$), with $q^{(1)}$ being its initial state[8]. Using the fact that $\mathcal{T}'$ is without fresh letters, let $F' : Q \to R'^*$ be its final output function. The initial word of our HDT0L system is then

$$d = \mathtt{pair}_{q^{(1)}}\left(F'\left(q^{(1)}\right)\right) \cdot \ldots \cdot \mathtt{pair}_{q^{(n)}}\left(F'\left(q^{(n)}\right)\right) \in \Delta^*$$

From the initial register values $(u_{I,r'})_{r' \in R'} \in (\Sigma^*)^{R'}$ of $\mathcal{T}'$, we define the final morphism:

$$h' \in \mathrm{Hom}(\Delta^*, \Sigma^*) \qquad \forall r' \in R', \quad \left[h'\left(r', q^{(1)}\right) = u_{I,r'} \quad \text{and} \quad \forall q \neq q^{(1)}, h'(r', q) = \varepsilon\right]$$

Finally, let $\delta'_{\mathrm{st}} : Q \to Q$ and $\delta'_{\mathrm{reg}} : Q \to (R' \to R'^*)$ be the components of the transition function of $\mathcal{T}'$. The morphisms $h_c \in \mathrm{Hom}(\Delta^*, \Delta^*)$ for $c \in \Gamma$ send $(r', q) \in \Delta$ to

$$h_c(r', q) = \mathtt{pair}_{q^{(i_1)}}(\delta'_{\mathrm{reg}}(q^{(i_1)}, c)(r')) \cdot \ldots \cdot \mathtt{pair}_{q^{(i_m)}}(\delta'_{\mathrm{reg}}(q^{(i_m)}, c)(r'))$$

where $i_1 < \ldots < i_m$ and $\{q^{(i_1)}, \ldots, q^{(i_m)}\} = \{q^{(?)} \in Q \mid \delta'_{\mathrm{st}}(q^{(?)}, c) = q\}$.

Checking that this HDT0L system computes the right function is a matter of mechanical verification, that has already been carried out in [22]. To wrap up the proof, we must justify that it is $(k+1)$-layered. To do so, let us fix a letter $c \in \Gamma$ and two layer indices $i, j \in \{0, \ldots, k+1\}$, and count the number $N_{r',q}$ of occurrences of $(r', q) \in \Delta_i$ among all the $h_c(\tilde{r}', \tilde{q})$ for $(\tilde{r}', \tilde{q}) \in \Delta_j$. The letter (r', q) can only appear in $h_c(\tilde{r}', \tilde{q})$ when $\tilde{q} = \delta(q, c)$, and in that case, its occurrences (if any) are in the substring $\mathtt{pair}_q(\delta'_{\mathrm{reg}}(q, c)(\tilde{r}'))$. So $N_{r',q}$ counts the occurrences of $r \in R'_i$ among the $\delta'_{\mathrm{reg}}(q, c)(\tilde{r}')$ for $\tilde{r}' \in R'_j$. Since $\mathcal{T}'$ is a $(k+1)$-layered SST, we are done. ◀

[8] Except for that, this enumeration of Q is arbitrary. We write $q^{(i)}$ instead of q_i to avoid confusion with the run of an automaton.

Proof of ($\Leftarrow$). The translation from HDT0L systems to single-state SSTs mentioned in Remark 2.6 (see Appendix A) is not enough: starting from a $(k+1)$-layered HDT0L system, it gives us a $(k+1)$-layered SST. But we can bring this down to k layers by adding states.

Let $(\Gamma, \Sigma, \Delta, d, (h_c)_{c\in\Gamma}, h')$ be a HDT0L system (with $d \in \Delta^*$, $h_c \in \mathrm{Hom}(\Delta^*, \Delta^*)$ for $c \in \Gamma$, and $h \in \mathrm{Hom}(\Delta^*, \Sigma^*)$). Suppose that it is $(k+1)$-layered with $\Delta = \Delta_0 \sqcup \cdots \sqcup \Delta_{k+1}$. This entails that $h_c(\Delta_0) \subseteq \Delta_0^*$, and furthermore that $(h_c)_{\restriction\Delta_0} : \Delta_0 \to \Delta_0^*$ satisfies a copylessness condition, that may succinctly be written as $(h_c)_{\restriction\Delta_0} \in \mathcal{M}^{\mathrm{cl}}_{\Delta_0,\varnothing}$ (cf. Definition 2.16).

We define a k-layered SST with:

- $\mathcal{M}^{\mathrm{cl}}_{\Delta_0,\varnothing}$ as the set of states (finite by Proposition 2.23), with the monoid identity as its initial state;
- the set of registers $R = \Delta \setminus \Delta_0 = \Delta_1 \sqcup \cdots \sqcup \Delta_{k+1}$, whose i-th layer is the $(i+1)$-th layer of the original HDT0L system ($0 \le i \le k$);
- the initial register contents $(h'(r))_{r\in R}$ – recall that h' is the final morphism;
- the transition function $(\alpha, c) \mapsto (\alpha \bullet (h_c)_{\restriction\Delta_0}, (h'_{\restriction\Delta_0^*} \circ \alpha)^{\odot} \circ (h_c)_{\restriction R})$ where $(-)^{\odot}$ extends functions $\Delta_0 \to \Sigma^*$ into morphisms in $\mathrm{Hom}((\Delta \cup \Sigma)^*, (R \cup \Sigma)^*)$ that map each letter in $R \cup \Sigma$ to itself (since $\Delta = \Delta_0 \sqcup R$, the domain of these morphisms is $(\Delta_0 \sqcup R \sqcup \Sigma)^*$);
- the final output function $\alpha \mapsto (h'_{\restriction\Delta_0^*} \circ \alpha)^{\odot}(d)$.

The layering condition for this SST is inherited is a direct consequence of the layering of the original HDT0L system, and one can check the functions computed by the two are the same. ◀

B.2 Proof of Corollary 3.3

▶ **Corollary 3.3.** *Every regular function can be computed by a single-state 1-layered SST.*

Any regular function is definable by some copyless SST, i.e. 0-layered SST. By Theorem 3.2, it can be turned into a 1-layered HDT0L system. The latter can be translated to a single-state SST by the construction of Appendix A. As can readily be seen from the definitions, this construction preserves the 1-layered property.

B.3 Proof of Proposition 3.7

▶ **Proposition 3.7.** *The sequential (and therefore regular) function defined by the transducer of Figure 2 (Section 2.2)* cannot *be computed by a* single-state copyless SST.

Consider any single-state copyless SST computing some $g : \{a,b,c\}^* \to \{a,b\}^*$ with a set of registers R. We wish to show g does not coincide with the function computed by the sequential transducer of Figure 2. Let $\omega \in (\{a,b\} \cup R)^*$ be the image of the single state by the output function, and for $x \in \{a,b,c\}$, let $\alpha_x : R \to (\{a,b\} \cup R)^*$ be the copyless assignment performed by the SST when it reads x (that is, using the notations of Definition 2.3, $\omega = F(q)$ and $\alpha_x = \delta_{\mathrm{reg}}(q, x)$ with $Q = \{q\}$). Let $\vec{u}$ be the initial register contents. Then

$$\forall x \in \{a,b,c\}^*, \forall n \in \mathbb{N},\ g(x \cdot c^n) = \omega^{\dagger} \circ (\alpha_c^{\dagger})^n \circ \alpha_x^{\dagger}(\vec{u})$$

Any register assignment $\beta : R \to (\{a,b,c\} \cup R)^*$ admits a unique extension into a monoid morphism $\beta^{\square} \in \mathrm{Hom}((\{a,b,c\} \cup R)^*, (\{a,b,c\} \cup R)^*)$ that maps every letter in $\{a,b,c\}$ to itself. Let $\omega_n = \left(\alpha_c^{\square}\right)^n(\omega)$ (so that $\omega_0 = \omega$). One can check that, for all $n \in \mathbb{N}$:

- $\omega_n^{\dagger} = \omega^{\dagger} \circ (\alpha_c^{\dagger})^n$;
- since α_c is copyless, $|\omega_n|_r \le |\omega|_r$ for all $r \in R$, writing $|w|_x$ for the number of occurrences of x in $w \in \Sigma^*$ for $x \in \Sigma$.

Let $(v_{x,r})_{r\in R} = \alpha_x^\dagger(\vec{u})$ for $x \in \{a,b,c\}$; that is, $v_{x,r}$ the value stored in the register $r \in R$ after the SST has read the single letter x. We can rewrite the above equation as

$$\forall x \in \{a,b,c\}^*,\ \forall n \in \mathbb{N},\ g(x \cdot c^n) = \omega_n^\dagger((v_{x,r})_{r\in R})$$

and derive a numerical (in)equality

$$\forall x \in \{a,b,c\}^*,\ \forall n \in \mathbb{N},\ |g(x \cdot c^n)|_a = |\omega_n|_a + \sum_{r\in R} |\omega_n|_r |v_{x,r}|_a \underset{n\to+\infty}{=} |\omega_n|_a + O(1)$$

using the fact that $|\omega_n|_r$, as a non-negative quantity lower than the constant $|\omega|_r$, is $O(1)$.

From this, it follows that as n increases, the difference between $|g(a \cdot c^n)|_a$ and $|g(b \cdot c^n)|_a$ stays bounded. This property distinguishes g from the $f : \{a,b,c\}^* \to \{a,b\}^*$ computed by the transducer given in Figure 2, since

$$\forall n \in \mathbb{N}, \quad |f(a \cdot c^n)|_a = \left|a^{n+1}\right|_a = n+1 \quad \text{and} \quad |f(b \cdot c^n)|_a = \left|b^{n+1}\right|_a = 0$$

B.4 Proof of Proposition 3.4

▶ **Proposition 3.4.** *Polyregular functions are the smallest class closed under composition that contains the regular functions and the squaring with underlining functions* $\texttt{squaring}_\Gamma$.

It is stated in the introduction to [4] that all regular functions are polyregular. One way to see this is discussed in Section 5: the characterization by pebble transducers given in [4] generalizes the classical definition of regular functions using two-way finite state transducers. This takes care of one direction of the equivalence; for the converse, observe that:

- sequential functions are regular, as already mentioned;
- since the SST of Example 2.10 is copyless, the iterated reverse function is regular.

B.5 Proof of Theorem 3.5

▶ **Theorem 3.5.** *Let $f : \Gamma^* \to \Sigma^*$. The following are equivalent:*

(i) *f is polyregular;*
(ii) *f can be obtained as a composition of layered SSTs;*
(iii) *f can be obtained as a composition of single-state 1-layered SSTs.*

Proof of (i) ⇒ (iii). Thanks to Proposition 3.4, we know that any polyregular functions can be written as a composition of a sequence of functions, each of which is either regular or equal to $\texttt{squaring}_\Gamma$ for some finite alphabet Γ. It suffices to show that each function in the sequence can in turn be expressed as a composition of single-state 1-layered SSTs.

We decompose $\texttt{squaring}_\Gamma$ as

$$\texttt{1234} \mapsto \underline{\texttt{4}}\texttt{321}\underline{\texttt{3}}\texttt{21}\underline{\texttt{2}}\texttt{11} \mapsto \underline{\texttt{1}}\texttt{2341}\underline{\texttt{2}}\texttt{3412}\underline{\texttt{3}}\texttt{4123}\underline{\texttt{4}}$$

The first step is performed by the SST of Example 2.4, which has a single state and, as mentioned in Section 2.3, is 1-layered. The second step can be implemented using a SST with a single state q (that we omit below for readability), two registers X (at layer 0) and Y (at layer 1) with empty initial values, an output function $F(q) = Y$, and

$$\forall c \in \Gamma, \quad \delta(c) = (X \mapsto X,\ Y \mapsto cY) \quad \text{and} \quad \delta(\underline{c}) = (X \mapsto cX,\ Y \mapsto \underline{c}XY)$$

As for regular functions, Corollary 3.3 takes care of them. ◀

Proof of (iii) ⇒ (ii). Immediate by definition. ◀

Proof of (ii) ⇒ (i). All functions computed by k-layered SSTs are polyregular; this applies in particular to single-state 1-layered SSTs. Therefore, their composition is also polyregular (according to Definition 2.11, polyregular functions are closed under composition). ◀

B.6 Proof of Proposition 3.9

▶ **Proposition 3.9.** *If f is an HDT0L transduction, then so is* $\mathbf{map}(f)$*. For each $k \geq 1$, the functions that can be computed by k-layered HDT0L systems are also closed under* $\mathbf{map}$*.*

Let $(\Gamma, \Sigma, \Delta, d, (h_c)_{c\in\Gamma}, h')$ be a HDT0L system computing $f : \Gamma^* \to \Sigma^*$. We define below a HDT0L system that computes $\mathbf{map}(f) : (\Gamma \cup \{\#\})^* \to (\Sigma \cup \{\#\})^*$.

- The intermediate alphabet is $\widehat{\Delta} = \Delta \cup \Sigma \cup \{\#, X\}$, assuming w.l.o.g. that $\# \notin \Delta \cup \Sigma$, where $X \notin \Delta \cup \Sigma \cup \{\#\}$ is an arbitrarily chosen fresh letter.
- The starting word is $Xd \in \widehat{\Delta}^*$.
- For $c \in \Gamma$, we extend h_c into $\widehat{h}_c \in \mathrm{Hom}(\Delta^*, \Delta^*)$ by setting $\widehat{h}_c(x) = x$ for $x \in \Sigma \cup \{\#, X\}$. Since the input alphabet is now $\Gamma \cup \{\#\}$, we also define the morphism $\widehat{h}_\#$ as the extension of h' (using $\Sigma \subset \widehat{\Delta}$) such that $\widehat{h}_\#(X) = Xd\#$ and $\widehat{h}_\#(x) = x$ for $x \in \Sigma \cup \{\#\}$.
- The final morphism $\widehat{h}'$ extends h' with $\widehat{h}'(X) = \varepsilon$ and $\widehat{h}'(x) = x$ for $x \in \Sigma \cup \{\#\}$

This shows that HDT0L transductions are closed under $\mathbf{map}$.

We now prove that for any $k \in \mathbb{N}_{\geq 1}$, this closure property holds for k-layered HDT0L transductions (so, in particular, for regular functions by taking $k = 1$). Suppose that f is computed by a k-layered HDT0L system with intermediate alphabet Δ and initial word $d \in \Delta^*$. One can build a k-layered HDT0L system which computes the same function f and such that *the initial word contains at most one occurrence of each letter*; the idea is to replace Δ and $d = d_1 \dots d_n$ by $\Delta \times \{1, \dots, n\}$ and $(d_1, 1), \dots, (d_n, n)$ where $n = |d|$, and to adapt the morphisms accordingly. Applying the above construction then results in a k-layered HDT0L system that computes $\mathbf{map}(f)$; note that if we did not have this property for the inital word, we would get a $(k + 1)$-layering instead.

C Proofs for §4 (composition by substitution)

C.1 Proof of Theorem 4.4

▶ **Theorem 4.4.** *Polyregular functions are closed under composition by substitution.*

We start by briefly recalling the definition of *polynomial list functions* from [4, Section 4]. The explanation is geared towards a reader familiar with the simply typed λ-calculus, which this system extends. The λ-terms defining polynomial list functions are generated by the grammar of simply typed λ-terms enriched with constants, whose meaning can be specified by extending the β-rule. For instance, given a *finite* set S and $a \in S$, every element of S can be used as a constant, another allowed constant is $\mathtt{is}_a^S$ and we have

$$\mathtt{is}_a^S\ b =_\beta \mathtt{true} \quad \text{if } a = b \qquad \mathtt{is}_a\ b =_\beta \mathtt{false} \quad \text{if } b \in S \setminus \{a\}$$

The grammar of simple types and the typing rules are also extended accordingly. For instance, any finite set S induces a type also written S, such that every element $a \in S$ corresponds to a term $a : S$ of this type. There are also operations expressing the cartesian product $(\times)$

and disjoint union (+) of two types; and, for any type τ, there is a type τ^* of lists whose elements are in τ. So the constant $\mathtt{is}_a^S$ receives the type

$$\mathtt{is}_a^S : S \to \{\mathtt{true}\} + \{\mathtt{false}\} \qquad \text{for any finite set } S$$

See [4, Section 4] for the other primitive operations that are added to the simply typed λ-calculus; we make use of `is`, `case`, `map` and `concat` here. Bojańczyk's result is that if Γ and Σ are finite sets, then the polynomial list functions of type $\Gamma^* \to \Sigma^*$ correspond exactly the polyregular functions.

▶ **Lemma C.1.** *Let $I = \{i_1, \dots, i_{|I|}\}$. Then the function $\mathtt{match}^{I,\tau} : I \to \tau \to \dots \to \tau \to \tau$ which returns its $(k+1)$-th argument when its 1st argument is i_k is a polynomial list function.*

Proof idea. By induction on $|I|$, using $\mathtt{is}_i^I$ $(i \in I)$ and $\mathtt{case}^{\{\mathtt{true}\},\{\mathtt{false}\},\tau}$. ◀

Proof of closure by CbS. Let $f : \Gamma^* \to I^*$, and for $i \in I$, $g_i : \Gamma^* \to \Sigma^*$ be polyregular functions. Assuming that f and g_i $(i \in I)$ are defined by polynomial list functions of the same name, the λ-term

$$\lambda w.\ \mathtt{concat}^{\Sigma}\ (\mathtt{map}^{I,\Sigma^*}\ (\lambda i.\ \mathtt{match}^{I,\Sigma^*}\ i\ (g_{i_1}\ w)\ \dots\ (g_{i_{|I|}}\ w))\ (f\ w))$$

computes $\mathrm{CbS}(f, (g_i)_{i\in I})$. ◀

C.2 Proof of Proposition 4.7

▶ **Proposition 4.7.** *A function f is comparison-free polyregular if and only if there exists some $k \in \mathbb{N}$ such that f has rank at most k. In that case, we write $\mathrm{rk}(f)$ for the least such k and call it the* rank *of f. If $(g_i)_{i\in I}$ is a family of comparison-free polyregular functions,*

$$\mathrm{rk}(\mathrm{CbS}(f, (g_i)_{i\in I})) \le 1 + \mathrm{rk}(f) + \max_{i\in I} \mathrm{rk}(g_i)$$

This is equivalent to claiming that the smallest class $\mathcal{C}$ of functions such that

- every regular function is in $\mathcal{C}$,
- and $\mathrm{CbS}(f, (g_i)_{i\in I}) \in \mathcal{C}$ for any regular $f : \Gamma^* \to I^*$ and any $(g_i : \Gamma^* \to \Sigma^*)_{i\in I} \in \mathcal{C}^I$,

contains all comparison-free polyregular functions. It suffices to show that $\mathcal{C}$ is closed under composition by substitution, which can be done by induction using the equation

$$\mathrm{CbS}(\mathrm{CbS}(f, (g_i)_i), (h_j)_j) = \mathrm{CbS}(f, (\mathrm{CbS}(g_i, (h_j)_j))_i)$$

The same equation explains the inequality on the rank that we claim in the proposition.

C.3 Proof of Proposition 4.8

▶ **Proposition 4.8.** *Let $f, g : \Gamma^* \to \Sigma^*$ be comparison-free polyregular functions and $L \subseteq \Gamma^*$ be a regular language. The function that coincides with f on L and with g on $\Gamma^* \setminus L$ is cfp, and so is $w \in \Gamma^* \mapsto f(w) \cdot g(w)$; both have rank at most $\max(\mathrm{rk}(f), \mathrm{rk}(g))$.*

Closure under regular conditionals We first observe that the particular case where f and g are both regular ($\mathrm{rk}(f) = \mathrm{rk}(g) = 0$) already appears in the literature [1]. We shall use it in further appendices, so let us state it as a stand-alone lemma.

▶ **Lemma C.2** ([1, Proposition 2])**.** *Let $f, g : \Gamma^* \to \Sigma^*$ be regular functions and $L \subseteq \Gamma^*$ be a regular language. The function that coincides with f on L and with g on $\Gamma^* \setminus L$ is regular.*

With this in hand, let us turn to the general case where $f, g : \Gamma^* \to \Sigma^*$ are cfp. It means that we have regular functions f' and g', as well as families of functions $(f''_i)_{i\in I}$ and $(g''_j)_{j\in J}$ such that

$$f \quad = \quad \mathrm{CbS}(f', (f''_i)_{i\in I}) \qquad \text{and} \qquad g \quad = \quad \mathrm{CbS}(g', (g''_j)_{j\in J})$$

(if f (or g) is of rank 0, we can introduce a spurious CbS by taking f' (resp. g') to be a constant function outputting a single letter over the singleton alphabet). Assume without loss of generality that $I \cap J = \varnothing$. Using Lemma C.2 applied to $f', g' : \Gamma^* \to (I \cup J)^*$, there is a function $h' : \Gamma^* \to (I \cup J)^*$ coinciding with f' over L and g' over $L \setminus \Gamma^*$. Setting $(h''_k)_{k\in I\cup J}$ to be the family of functions such that $h''_i = f''_i$ for $i \in I$ and $h''_j = g''_j$ for $j \in J$, we obtain a cfp function $h = \mathrm{CbS}(h', (h''_k)_{k\in I\cup J})$ corresponding to the desired conditional.

Closure under concatenation Similarly, first observe that the result holds for regular functions. Using SSTs, this can be shown using a product construction.

Then, taking regular functions f' and g', as well as families of functions $(f''_i)_{i\in I}$ and $(g''_j)_{j\in J}$ so that $f = \mathrm{CbS}(f', (f''_i)_{i\in I})$ and $g = \mathrm{CbS}(g', (g''_j)_{j\in J})$ with I and J disjoint as above, one check that the pointwise concatenation $f \cdot g$ is equal to $\mathrm{CbS}(f' \cdot g', (h''_k)_{k\in I\cup J})$, which is cfp since $f' \cdot g'$ is regular.

D Proofs for §5 (comparison-free pebble transducers)

D.1 Proof of Proposition 5.5

▶ **Proposition 5.5.** *If f is computed by a k-CFPT, and the g_i are computed by l-CFPTs, then* $\mathrm{CbS}(f, (g_i)_{i\in I})$ *is computed by a $(k + l)$-CFPT.*

First note that any k-CFPT can be transformed into an equivalent k-CFPT whose transition functions $\delta : Q \times (\Gamma \cup \{\triangleright, \triangleleft\})^p \to Q \times (\mathbb{N}^p \to \mathrm{Stack}_k) \times \Sigma^*$ are such that, for every input $(q, \vec{b})$, we have either $\pi_3(\delta(q, \vec{b})) = \varepsilon$ (in which case we call $\delta(q, \vec{b})$ a *silent transition*) or $\pi_3(\delta(q, \vec{b})) \in \Sigma$ and $\pi_2(\delta(q, \vec{b}))$ is the identity. So, without loss of generality, suppose that we have a k-CFPT $\mathcal{T}_f$ implementing f is of this shape, with state space Q_f and transition function δ_f Similarly, we may assume without loss of generality that the current height of the stack is tracked by the state of CFPTs if we allow multiple final states; assume that we have such height-tracking l-CFPT and that we have l-CFPTs $\mathcal{T}_i$ implementing g_i with state spaces Q_i and transition functions δ_i.

We combine these CFPTs into a single $k + l$ CFPT $\mathcal{T}'$ with state space

$$Q' \quad = \quad Q_f \quad \sqcup \quad Q_f \times \bigsqcup_{i\in I} Q_i$$

The initial and final states are those of $\mathcal{T}_f$. The high-level idea is that $\mathcal{T}'$ behaves as $\mathcal{T}_f$ until it produces an output $i \in I$; in such a case it "performs a call" to $\mathcal{T}_i$ that might spawn additional heads to perform its computations. At the end of the execution of $\mathcal{T}_i$, we return the control to $\mathcal{T}_f$. Formally speaking, the transition function δ' of $\mathcal{T}'$ behaves as follows:

- $\delta'(q, \vec{b}) = \delta_f(q, \vec{b})$ if $q \in Q_f$ and $\delta_f(q, \vec{b})$ is *silent*.
- otherwise we take, we have $\pi_3(\delta_f(q, \vec{b})) = i$ for some $i \in I$. Calling r_i the initial state of $\mathcal{T}_i$, we set $\pi_1(\delta'(q, \vec{b})) = (q, r_i)$ and $\pi_2(\delta'(q, \vec{b}))$ corresponds to push a new pebble onto the stack. We make $\delta'(q, \vec{b})$ silent in such a case.
- $\delta'((q, r), \vec{b}\vec{b'})$ then corresponds to $\delta_i(r, \vec{b'})$ if we are not in the situation where the stack height is 1 and the stack update function is pop.

- otherwise we take $\pi_1(\delta'((q,r),\vec{b}b')) = \pi_1(\delta_f(q,\vec{b}))$, $\pi_2(\delta'(q,n+1,\vec{b}b'))$ to be a pop action and $\pi_3(\delta'((q,r),\vec{b}b')) = \pi_3(\delta_i((q,r),\vec{b}b'))$.

D.2 Proof of Theorem 5.6

▶ **Theorem 5.6.** *If $f : \Gamma^* \to \Sigma^*$ is computed by a k-CFPT, for $k \geq 2$, then there exist a finite alphabet I, a regular function $h : \Gamma^* \to I^*$ and a family $(g_i)_{i\in I}$ computed by $(k-1)$-CFPTs such that $f = \mathrm{CbS}(h,(g_i)_{i\in I})$.*

Assume we have $f : \Gamma^* \to \Sigma^*$ computed by a k-CFPT $\mathcal{T}$ with state space Q and transition function δ that we assume to be disjoint from Σ. For each $q \in Q$, we describe a $k-1$ CFPT $\mathcal{T}_q$ with the same state space, initial state q and transition function δ_q such that, for every $b' \in \mathbb{N}$, $\vec{b} \in \mathrm{Stack}_l$ for $l \leq k-1$ and $q' \in Q$, $\delta(q', b'\vec{b})$ and $\delta_q(q', \vec{b})$ coincide on the first and last component; on the second component, we require they also coincide up to the difference in stack size. If we fix $r \in Q$, by [28, Theorem 4.7], the language consisting of those $w \in \Gamma^*$ such that $\mathcal{T}_q$ halts on r is regular. Since regular languages are closed under intersection, for any map $\gamma \in Q^Q$, the language $L_\gamma \subseteq \Gamma^*$ of those words w such that $\mathcal{T}_q$ halts on $\gamma(q)$ is regular.

Now fix $\gamma \in Q^Q$ and let us describe a 1-CFPT transducer $\mathcal{T}_\gamma$ intended to implement the restriction of a function $h : \Gamma^* \to (\Sigma \cup Q)^*$ to L_γ. $\mathcal{T}_\gamma$ has the same state space and initial state as $\mathcal{T}$, but has a transition function δ_γ defined by

$$\delta_\gamma(q,b) = \begin{cases} \delta(q,b) & \text{if } \pi_2(\delta(q,b)) \text{ is not a push} \\ (\gamma(r),(p \mapsto p),r) & \text{otherwise, for } r = \pi_1(\delta(q,b)) \end{cases}$$

Since $\Gamma^* = \bigcup_{\gamma\in Q^Q} L_\gamma$, by applying repeatedly Lemma C.2, this determines the regular function $h : \Gamma^* \to (\Sigma \cup Q)^*$. We can then check that $f = \mathrm{CbS}(h,(g_i)_{i\in\Sigma\cup Q})$ where g_a is the constant function outputting the one-letter word a for $a \in \Sigma$ (which can certainly be implemented by a 1-CFPT) and g_q is the function $\Gamma^* \to \Sigma^*$ implemented by the $(k-1)$-CFPT $\mathcal{T}_q$.

D.3 Proof of Corollary 5.7

▶ **Corollary 5.7.** *For all $k \in \mathbb{N}$, the functions computed by $(k+1)$-CFPTs are exactly the comparison-free polyregular functions of rank at most k.*

The proof goes by induction over $k \in \mathbb{N}$. By Theorem 5.3, the result holds for $k = 0$ since 2DFTs characterize regular functions; let us detail each direction of the inductive case $k > 0$:

- for the left-to-right inclusion, assume we are given a $(k+1)$-CFPT computing f and apply Theorem 5.6 to obtain h and g_is such that $f = \mathrm{CbS}(h,(g_i)_{i\in I})$ with h regular and the g_is computable by k-CFPTs. The induction hypothesis implies that the g_is have rank $< k$, and thus f has rank $\leq k$.
- conversely, if f has rank k, it can be written as $\mathrm{CbS}(h,(g_i)_{i\in I})$ with h regular and the g_is with rank $< k$; the induction hypothesis implies that the g_is can be computed by k-CFPTs. By Theorem 5.3, h is computable by a 1-CFPT, so by Proposition 5.5, f is computed by a $(k+1)$-CFPT

E Closure under composition

This section is dedicated to establishing the following theorem, which constitutes most of the work that goes into proving Theorem 6.1.

▶ **Theorem E.1.** *Comparison-free polyregular functions are closed under composition.*

First, by induction on the rank of the left-hand side of the composition, we can reduce to the case where that side is a mere regular function, using the straightforward identity

$$\mathrm{CbS}(f, (g_i)_{i\in I}) \circ h = \mathrm{CbS}(f \circ h, (g_i \circ h)_{i\in I})$$

We then treat this case by another induction, this time on the rank of the right-hand side. The base case is handled by invoking the closure under composition of regular functions. Therefore, what remains is the following inductive case.

▶ **Lemma E.2.** *Let $f : \Gamma^* \to I^*$ be a regular function and let $(g_i)_{i\in I}$ be a family of comparison-free polyregular functions $\Gamma^* \to \Sigma^*$. Suppose that for all regular $h : \Sigma^* \to \Delta^*$ and all $i \in I$, the composite $h \circ g_i$ is comparison-free polyregular.*

Then, for all regular $h : \Sigma^* \to \Delta^*$, $h \circ \mathrm{CbS}(f, (g_i)_{i\in I})$ is comparison-free polyregular.

Our proof of the above lemma and related subclaims rely on the properties of transition monoids introduced in Section 2.4 and on the combinatorics of register transitions discussed in the paragraphs following Proposition 6.2.

▶ **Lemma E.3.** *Let δ be the transition function of some copyless SST $\Sigma^* \to \Delta^*$ whose sets of states and registers are Q and R respectively, so that $\delta(-, c) \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta} \wr Q$ for $c \in \Sigma$. Let*

$$\psi_\delta \in \mathrm{Hom}(\Sigma^*, \mathcal{M}^{\mathrm{cl}}_{R,\Delta} \wr Q) \quad \textit{such that} \quad \forall c \in \Sigma,\ \psi_\delta(c) = \delta(-, c)$$

and $\varphi_\delta = \mathtt{erase}_\Delta \circ \psi_\delta$ as in Remark 2.22, $q \in Q$, $r \in R$, $\alpha \in \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ and $j \in \{0, \dots, |\alpha(r)|\}$. Then the following function $\Sigma^* \to \Delta^*$, defined thanks to Proposition 6.2, is regular:

$$s \quad \mapsto \quad \begin{cases} w_j & \textit{where} \quad \pi_2(\psi_\delta(s)(q))(r) = w_0 r'_1 w_1 \dots r'_n w'_n \quad \textit{if } \pi_2(\varphi_\delta(s)(q)) = \alpha \\ \varepsilon & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{otherwise} \end{cases}$$

(recall that $\pi_2 : Q \times M \to M$ is the second projection and $M \wr Q = Q \to Q \times M$).

Proof. We consider during this proof that the names q, r, α and j introduced in the above statement are *not in scope*, so that we can use those variable names for generic elements of Q, R, $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ and $\mathbb{N}$ instead. Those data will be given other names when we need them.

We build a copyless SST whose set of states is $Q \times \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$. This is made possible by the finiteness of $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ (Proposition 2.23). As for the set of registers, we would like it to *vary depending on the current state* for the sake of conceptual clarity, i.e. to have a family of finite sets indexed by $Q \times \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$; when the SST moves from state (q, α) to (q', α'), it would perform a register assignment from $R_{q,\alpha}$ to $R_{q',\alpha'}$ (described by a map $R_{q',\alpha'} \to (\Delta \cup R_{q,\alpha})^*$). Such devices have been called *state-dependent memory copyless SSTs* in [30], and they are clearly equivalent in expressive power to usual copyless SSTs.

The idea is that we want the configuration (current state plus register contents) of our new SST, after reading $s = s_1 \dots s_n$, to faithfully represent

$$\psi_\delta(s)(q_0) = (\delta(-, s_1) \bullet \cdots \bullet \delta(-, s_n))(q_0) \in Q \times \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$$

where δ and ψ_δ are given in the lemma statement, and q_0 is the given state that was called q in that statement. Following Proposition 6.2, since we already have the "shape" stored in the second component $\mathcal{M}^{\mathrm{cl}}_{\Delta,\varnothing}$ of the set $Q \times \mathcal{M}^{\mathrm{cl}}_{\Delta,\varnothing}$ of new states, it makes sense to use the register to store the "labels", hence $R_{q,\alpha} = R_\alpha$ with

$$R_\alpha = \{(r, j) \mid r \in R,\ j \in \{0, \dots, |\alpha(r)|\}\} \quad \text{so that} \quad (\Delta^*)^{R_\alpha} \cong \prod_{r\in R} (\Delta^*)^{|\alpha(r)|+1}$$

The configurations of our SST are thus in bijection with $Q \times \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ via Proposition 6.2, and we would like the transition performed when reading $c \in \Sigma$ to correspond through this bijection to (using the notations of Definition 2.3)

$$(q, \beta) \in Q \times \mathcal{M}^{\mathrm{cl}}_{R,\Delta} \quad \mapsto \quad (\delta_{\mathrm{st}}(q),\, \beta \bullet \delta_{\mathrm{reg}}(q))$$

For a fixed $\beta' \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$, let us consider the right multiplication $\beta \mapsto \beta \bullet \beta'$ in $\mathcal{M}^{\mathrm{cl}}_{R,\Delta}$. Since $\mathtt{erase}_\Delta : \mathcal{M}^{\mathrm{cl}}_{R,\Delta} \to \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ is a morphism, the "shape" of $\beta \bullet \beta'$ can be obtained from the "shape" of β by multiplying by $\alpha' = \mathtt{erase}_\Delta(\beta')$. The important point is to show that we can obtain the new labels from the old ones by a *copyless assignment* – formally speaking, that for any $\alpha \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ there exists a copyless

$$\gamma_{\alpha,\beta'} : R_{\alpha \bullet \alpha'} \to (\Delta \cup R_\alpha)^*$$

such that for any $\beta \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ such that $\mathtt{erase}_\Delta(\beta) = \alpha$, which therefore corresponds to

$$\left(\alpha, \vec{\ell}\right) \quad \text{for some} \quad \vec{\ell} \in (\Delta^*)^{R_\alpha} \cong \prod_{r \in R} (\Delta^*)^{|\alpha(r)|+1}$$

the shape-label pair that corresponds to $\beta \bullet \beta'$ is $(\alpha \bullet \alpha',\, \gamma^\dagger_{\alpha,\beta'}(\vec{\ell}))$ (cf. Definition 2.2).

Our next task is to analyze the composite assignment $\beta \bullet \beta'$ in order to derive a $\gamma_{\alpha,\beta'}$ that works. Let $r'' \in R$. First, if $\alpha'(r'') = r'_1 \dots r'_n \in R^*$, then

$$\beta'(r'') = w'_0 r'_1 w'_1 \dots r'_n w'_n \quad \text{for some} \quad w'_0, \dots, w'_n \in \Delta^*$$

and by applying the unique morphism $\beta^\odot \in \mathrm{Hom}((\Delta \cup R)^*, (\Delta \cup R)^*)$ that extends β and sends letters of Δ to themselves, we have

$$(\beta \bullet \beta')(r'') = \beta^\odot(\beta'(r)) = w'_0 \cdot \beta(r'_1) \cdot w'_1 \cdot \ldots \cdot \beta(r'_n) \cdot w'_n$$

Let us decompose further, for $i \in \{1, \dots, n\}$:

$$\beta(r'_i) = w_{i,0} r_{i,1} w_{i,1} \dots w_{i,n_i} r_{n_i} \quad \text{for some} \quad w_{i,0}, \dots, w_{i,n_i} \in \Delta^*$$

By plugging this into the previous equation, we have $(\beta \bullet \beta')(r'') = w_0 r_1 w_1 \dots r_m w_m$ where

$$\{r_1, \dots, r_m\} = \bigcup_{i=1}^{n} \{r_{i,1}, \dots, r_{i,n_i}\}$$

Furthermore, *each w_k for $k \in \{0, \dots, m\}$ is a concatenation of some w'_i and some $w_{i,j}$*, and from the formal expression of w_k depending on these w'_i and $w_{i,j}$ – which only depends on the shape α and α' – we can derive a definition of $\gamma_{\alpha,\beta'}(r'', k)$. For instance,

$$w_{42} = w_{3,2} w'_3 w_{4,0} \qquad \rightsquigarrow \qquad \gamma(r'', 42) = (r'_3, 2) \cdot w'_3 \cdot (r'_4, 0) \in (\Delta \cup R_{\alpha \bullet \alpha'})^*$$

Observe that this does not refer to the $w_{i,j}$; therefore, $\gamma_{\alpha,\beta'}$ does not depend on β, as required. One can check that defined this way, $\gamma_{\alpha,\beta'}$ is indeed a copyless assignment and that the desired property of $\gamma^\dagger_{\alpha,\beta'}$ holds.

What we have just seen is the heart of the proof. We leave it to the reader to finish the construction of the copyless SST. ◀

With this done, we can move on to proving Lemma E.2, which suffices to finish the proof of Theorem E.1.

Proof of Lemma E.2. Let $w \in \Gamma^*$ be an input string. In the composition, we feed to a copyless SST $\mathcal{T}_h$ that computes h the word $\mathrm{CbS}(f, (g_i)_{i\in I})(w) = g_{i_1}(w) \dots g_{i_k}(w)$ where $f(w) = i_1 \dots i_k$. A first idea is therefore to tweak $\mathcal{T}_h$ into a new copyless SST that takes I^* as input and which executes, when it reads $i \in I$, the transition of $\mathcal{T}_h$ induced by $g_i(w)$. If we call h'_w the regular function computed by this new SST, we would then have $h'_w(f(w)) = h \circ \mathrm{CbS}(f, (g_i)_{i\in I})(w)$. The issue is of course that h'_w depends on the input w.

More precisely, the data that h'_w depends on is the family of transitions

$$(\psi_\delta \circ g_i(w))_{i\in I} \in (\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I \qquad \text{(see Lemma E.3 for } \psi_\delta)$$

where Q, R and δ are respectively the set of states, the set of registers and the transition function of $\mathcal{T}_h$. We will be able to disentangle this dependency by working with

$$(\varphi_\delta \circ g_i(w))_{i\in I} = (\mathtt{erase}_\Delta \circ \psi_\delta \circ g_i(w))_{i\in I} \in (\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$$

Concretely:

▷ Claim E.4. For each $\vec{\mu} \in (\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$, there exist:

- a finite alphabet $\Lambda_{\vec{\mu}}$ equipped with a function $\iota_{\vec{\mu}} : \Lambda_{\vec{\mu}} \to I$;
- a regular function $h''_{\vec{\mu}} : I^* \to (\Delta \cup \Lambda_{\vec{\mu}})^*$;
- and regular functions $l_\lambda : \Sigma^* \to \Delta^*$ for $\lambda \in \Lambda$;

such that for $i_1 \dots i_n \in I^*$ and $w \in \Gamma^*$, *if* $(\varphi_\delta \circ g_i(w))_{i\in I} = \vec{\mu}$, *then*

$$h(g_{i_1}(w) \cdot \dots \cdot g_{i_n}(w)) \quad = \quad \text{replace each } \lambda \in \Lambda_{\vec{\mu}} \text{ in } h''_{\vec{\mu}}(i_1 \dots i_n) \text{ by } l_\lambda \circ g_{\iota(\lambda)}(w)$$

Proof. Proposition 6.2 says that every $\beta = \psi_\delta(g_i(w)) \in \mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ can be decomposed into a shape $\alpha = \mathtt{erase}_\Delta(\beta) \in \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ and a finite family $\vec{\ell}$ of strings in Δ^*. Each $\beta(r)$ for $r \in R$ can then be reconstituted as an interleaving of letters in $\alpha(r)$ with labels in $\vec{\ell}$, a process that can be decomposed into two steps:

- first, interleave the letters of $\alpha(r)$ with placeholder letters, taken from an alphabet disjoint from both Δ and R;
- then *substitute* the labels for those letters.

Roughly speaking, this will allow us to manipulate an assignment with placeholders without knowing the labels, and then add the labels afterwards.

Let $\vec{\mu} \in (\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$. We define a copyless SST $\mathcal{T}_{\vec{\mu}}$ with the same sets of states and registers as $\mathcal{T}_h$, namely Q and R. Its initial register values and final output function are also the same. It computes a function $I^* \to (\Delta \cup \Lambda_{\vec{\mu}})^*$, and its transition function is

$$\delta_{\vec{\mu}} : (q, i) \mapsto \Big(\pi_1 \circ \mu_i(q),\ \Big(r \mapsto \mathtt{interleave}\Big(\lambda_0^{q,i,r} \dots \lambda^{q,i,r}_{|\pi_2(\mu_i(q))(r)|},\ \pi_2(\mu_i(q))(r)\Big)\Big)\Big)$$

where $\mathtt{interleave}(u_0 \dots u_n, v_1 \dots v_n) = u_0 v_1 u_1 \dots v_n u_n$ for letters $u_0, \dots, u_n, v_1, \dots, v_n$ over some alphabet (recall also that $\mu_i : Q \to Q \times \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ for $i \in I$). Thus, we take

$$\Lambda_{\vec{\mu}} = \Big\{\lambda_j^{q,i,r} \ \Big|\ q \in Q,\ i \in I,\ r \in R,\ j \in \{0, \dots, |\pi_2(\mu_i(q))(r)|\}\Big\} \qquad \iota(\lambda_j^{q,i,r}) = i$$

and $h''_{\vec{\mu}}$ to be the function computed by $\mathcal{T}_{\vec{\mu}}$. (Note that although $\delta_{\vec{\mu}}$ does not involve letters from Δ, the final output function and the initial register contents do.) Finally, given $\lambda = \lambda_j^{q,i,r} \in \Lambda_{\vec{\mu}}$, we define l_λ to be the regular function provided by Lemma E.3 for the transition function δ of $\mathcal{T}_h$, the state q_0 (which is the initial state of both $\mathcal{T}_h$ and $\mathcal{T}_{\vec{\mu}}$), the register r, the assignment shape $\alpha = \pi_2(\mu_i(q))$ and the position $j \in \{0, \dots, |\alpha(r)|\}$.

Let $w \in \Gamma^*$ be such that $(\varphi_\delta \circ g_i(w))_{i\in I} = \vec{\mu}$. Consider $\chi_w \in \mathrm{Hom}((\Delta \cup \Lambda_{\vec{\mu}})^*, \Delta^*)$ which maps each letter of Δ to itself and each $\lambda \in \Lambda_{\vec{\mu}}$ to $l_\lambda \circ g_{\iota(\lambda)}(w)$. It lifts to a morphism $\widehat{\chi_w} \in \mathrm{Hom}(\mathcal{M}^{\mathrm{cl}}_{R,\Delta\cup\Lambda_{\vec{\mu}}}, \mathcal{M}^{\mathrm{cl}}_{R,\Delta})$, and we have $\widehat{\chi_w}(\delta_{\vec{\mu}}(-,i)) = \psi_\delta \circ g_i(w)$. This leads to the following invariant: the configuration of $\mathcal{T}_h$ after reading $g_{i_1}(w) \cdot \ldots \cdot g_{i_n}(w)$ is, in a suitable sense, the "image by χ_w" of the configuration of $\mathcal{T}_{\vec{\mu}}$ after reading $i_1 \ldots i_n$. (In other words, the "image of the SST $\mathcal{T}_{\vec{\mu}}$ by χ_w" is the copyless SST computing h'_w that we sketched at the very beginning of this proof of Lemma E.2.) This directly implies the property relating h, $h''_{\vec{\mu}}$ and $(l_\lambda)_{\lambda\in\Lambda_{\vec{\mu}}}$ that we wanted. ◁

Let us finish proving Lemma E.2 using the fact we just proved. First of all, since the letters of $\Lambda_{\vec{\mu}}$ only serve as placeholders to be eventually substituted, they can be renamed at our convenience. That means that we can take the $\Lambda_{\vec{\mu}}$ to be *disjoint* for $\vec{\mu} \in (\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$, and define Λ to be their disjoint union. We also take $\iota : \Lambda \to I$ to be the unique common extension of the $\iota_{\vec{\mu}}$. In the same spirit, we glue together the functions $h''_{\vec{\mu}} \circ f$ into

$$H : w \in \Gamma^* \mapsto h''_{(\varphi_\delta \circ g_i(w)|i\in I)}(f(w)) \in (\Delta \cup \Lambda)^*$$

From the above equation on $h''_{\vec{\mu}}$, one can then deduce for *all* $w \in \Gamma^*$ *without condition* that

$$h(\mathrm{CbS}(f, (g_i)_{i\in I})(w)) = \mathrm{CbS}(H, (l_\lambda \circ g_{\iota(\lambda)})_{\lambda\in\Lambda})(w)$$

(strictly speaking, one should have a family indexed by $\Delta \cup \Lambda$ on the right-hand side – to comply with that, just extend the family with constant functions equal to x for each $x \in \Delta$).

Using the above equation, we can rephrase our goal: we want to prove that the function $\mathrm{CbS}(H, (l_\lambda \circ g_{\iota(\lambda)})_{\lambda\in\Lambda})$ is comparison-free polyregular. This class of functions is – by definition – closed under composition by substitution, so we can reduce this to the following subgoals:

- H is comparison-free polyregular: in fact, it is regular, because regular functions are closed under composition and regular conditionals (Lemma C.2). This argument relies on the finiteness of the indexing set $(\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$ – a consequence of Proposition 2.23 – and on the regularity of the language $\{w \in \Gamma^* \mid (\varphi_\delta \circ g_i(w))_{i\in I} = \vec{\mu}\}$ for any $\vec{\mu}$. The reasons for the latter are as follows:
  - φ_δ is a morphism whose codomain $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q$ is finite, so $\varphi_\delta^{-1}(\{\mu_i\})$ is regular for $i \in I$;
  - the functions g_i for $i \in I$ are assumed to be comparison-free polyregular, so they preserve regular languages by inverse image, as all polyregular functions do [4];
  - regular languages are closed under finite intersections, and I is finite.
- $l_\lambda \circ g_{\iota(\lambda)}$ is comparison-free polyregular for all $\lambda \in \Lambda$: because our main existence claim states that l_λ is regular for all $\lambda \in \Lambda$, and one of our assumptions is that any g_i (for $i \in I$) postcomposed with any regular function gives us a comparison-free polyregular function.

◀

F A lower bound on growth from the rank

In this section, we prove a statement that directly implies the last claim of Theorem 7.1:

▶ **Theorem F.1.** *Let $f : \Gamma^* \to \Sigma^*$ be comparison-free polyregular of rank at least 1. Then there exists a sequence of inputs $(s_n)_{n\in\mathbb{N}} \in (\Gamma^*)^{\mathbb{N}}$ such that $|s_n| = O(n)$ and $|f(s_n)| \geq n^{\mathrm{rk}(f)+1}$.*

Let us start by proving the lemmas stated in the main text which this theorem depends on.

F.1 Proofs for the lemmas in Section 7

Recall that the notion of r-split has been defined in Definition 7.3.

▶ **Lemma 7.5.** *Let $f : \Gamma^* \to \Sigma^*$ be a regular function. There exist a morphism to a* finite *monoid $\nu_f : \Gamma^* \to \mathcal{N}(f)$ and, for each $c \in \Sigma$, a set of* producing triples *$P(f, c) \subseteq \mathcal{N}(f)^3$ such that, for any* 1-split *according to ν_f composed of $u, v, w \in \Gamma^*$ – i.e. $\nu_f(uv) = \nu_f(u)$ and $\nu_f(vw) = \nu_f(w)$ – we have:*

- *if $(\nu_f(u), \nu_f(v), \nu_f(w)) \in P(f, c)$, then $|f(uvw)|_c > |f(uw)|_c$;*
- *otherwise (when the triple is not producing), $|f(uvw)|_c = |f(uw)|_c$.*

Furthermore, in the producing case, we get as a consequence that $\forall n \in \mathbb{N},\ |f(uv^n w)|_c \geq n$.

Proof idea. We reuse an idea from [25], but instead of using transition monoids of two-way transducers, we rely on monoids of copyless register assignments. We shall use the notations introduced in Section 2.4 for these monoids and the operations they support.

Let R and Σ be finite alphabets. First, we factor $\mathtt{erase}_\Sigma : \mathcal{M}^{\mathrm{cl}}_{R,\Sigma} \to \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ into two surjective morphisms $\mathcal{M}^{\mathrm{cl}}_{R,\Sigma} \to \mathcal{M}^{\mathrm{cl01}}_{R,\Sigma} \to \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$, going through a new monoid which keeps some information about the letters of Σ but is still *finite*. To do so, we define an equivalence relation on register assignments as follows: for $\alpha, \beta \in \mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$, we say that $\alpha \sim \beta$ when

- $\mathtt{erase}_\Sigma(\alpha) = \mathtt{erase}_\Sigma(\beta)$;
- for each $r \in R$, the sets of letters *from* Σ that appear in $\alpha(r)$ and $\beta(r)$ are equal.

One can show that $\sim$ is a congruence, so we may form the quotient monoid $\mathcal{M}^{\mathrm{cl01}}_{R,\Sigma} = \mathcal{M}^{\mathrm{cl}}_{R,\Sigma}/\sim$. Thanks to the first clause in the definition of $\sim$, the morphism $\mathtt{erase}_\Sigma$ factors through the canonical projection. The quotient is finite since each equivalence class has a representative α such that $|\alpha(r)| \leq |R| + |\Sigma|$ for all $r \in R$: essentially, $\sim$ only takes into account the presence or absence of each letter in Σ, not their multiplicity (hence the notation "01").

Next, let $f : \Gamma^* \to \Sigma^*$ be computed by some copyless SST $(Q, q_0, R, \delta, \vec{u}_I, F)$. We take $\mathcal{N}(f) = \mathcal{M}^{\mathrm{cl01}}_{R,\Sigma} \wr Q$ and define ν_f as a composition $\Gamma^* \to \mathcal{M}^{\mathrm{cl}}_{R,\Sigma} \wr Q \to \mathcal{N}(f)$ where the first morphism – which we may call ψ_δ, as in Lemma E.3 – maps $c \in \Gamma$ to $\delta(-, c)$ and the second morphism is the canonical projection.

What we need to show now is that, given a *1-split* $(u, v, w) \in (\Gamma^*)^3$ with respect to ν_f, the comparison between $|f(uvw)|$ and $|f(uw)|$ depends only on $\nu_f(x)$ for $x \in \{u, v, w\}$.

Let q' and $\vec{u}'_I$ be the state and register values of the SST after reading u; to be more formal, $\psi_\delta(u)(q_0) = (q', \alpha)$ and $\alpha^\dagger(\vec{u}_I) = \vec{u}'_I$. Note that q' is also the first component of the pair $\nu_f(u)(q_0)$; since $\nu_f(uv) = \nu_f(u)$ (by definition of 1-split), the SST reaches the state q' after reading uv as well: $\psi_\delta(q', v) = (q', \beta)$ for some $\beta \in \mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$.

Let $\psi_\delta(w)(q') = (q'', \gamma)$. Then

$$f(uvw) = F(q'')^\dagger \circ (\beta \bullet \gamma)^\dagger(\vec{u}'_I) \qquad f(uw) = F(q'')^\dagger \circ \gamma^\dagger(\vec{u}'_I)$$

Since $\nu_f(vw) = \nu_f(w)$, we have $\mathtt{erase}_\Sigma(\beta \bullet \gamma) = \mathtt{erase}_\Sigma(\gamma)$. Therefore, $\omega = (\beta \bullet \gamma)^\odot(F(q''))$ and $\omega' = \gamma^\odot(F(q''))$ have the same letters from R with the same multiplicities (and appearing in the same order, although this does not matter for us here): $|\omega|_r = |\omega'|_r$ for all $r \in R$. This is why the two sums over R cancel out in the following computation (writing $\vec{u}'_I = (u'_r)_{r \in R}$):

$$\begin{aligned}
\forall c \in \Sigma, \quad |f(uvw)|_c - |f(uw)|_c &= |\omega^\dagger(\vec{u}'_I)|_c - |(\omega')^\dagger(\vec{u}'_I)|_c \\
&= |\omega|_c + \sum_{r \in R} |\omega|_r \cdot |u'_r|_c - |\omega'|_c - \sum_{r \in R} |\omega'|_r \cdot |u'_r|_c \\
&= |\omega|_c - |\omega'|_c
\end{aligned}$$

From now on, let $c \in \Sigma$. From the definition of $\beta^{\odot}$, we have

$$|\omega|_c = |\beta^{\odot}(\omega')|_c = |\omega'|_c + \sum_{r \in R} |\omega'|_r \cdot |\beta(r)|_c$$

So we get the dichotomy of the lemma statement:

- if there exists some $r \in R$ such that $|\omega'|_r > 0$ and $|\beta(r)|_c > 0$, then $|f(uvw)| > |f(uw)|$;
- otherwise, $|f(uvw)| = |f(uw)|$.

For each $r \in R$, the condition $|\omega'|_r > 0$ can be checked from q' and $\nu_f(w)$; in turn, q' depends only on $\nu_f(u)$. As for $|\beta(r)|_c > 0$, since it is a condition on the presence or not of a certain letter from Σ in $\beta(r)$, without considering its precise multiplicity, it depends only on $\nu_f(v)$: this is the information that ν_f was designed to encode. This gives us the definition of the set of producing triples $P(f, c)$.

There remains a final claim to prove in the lemma statement, concerning $|f(uv^n w)|_c$ when $(u, v, w) \in P(f, c)$ and $n \in \mathbb{N}$. Using $\mu(uv) = \mu(u)$, one can show that for all $m \in \mathbb{N}$, the triple (uv^m, v, w) is also a producing 1-split. So we have

$$|f(uv^n w)|_c > |f(uv^{n-1} w)|_c > \cdots > |f(uw)|_c$$

and since all elements of this sequence are natural numbers, $|f(uv^n w)|_c \geq n$. ◀

Next, we prove Proposition 7.8 before Lemma 7.7, following the order of logical dependency.

▶ **Proposition 7.8.** *Let Γ be an alphabet, M be a finite monoid and $\varphi : \Gamma^* \to M$ be a morphism. There exists $N \in \mathbb{N}$ such that any string $s = uvw \in \Gamma^*$ such that $|v| \geq N$ admits an r-split $s = u'v'_1 \dots v'_r w'$ according to φ in which u is a prefix of u' and w is a suffix of w'.*

Proof. By the finite Ramsey theorem for pairs, there exists $R \in \mathbb{N}$ such that every complete undirected graph with at least R vertices whose edges are colored using $|M|$ colors contains a monochromatic clique with $r + 3$ vertices. We take $N = R - 1$.

Let $s = uvw \in \Gamma^*$ with $|v| \geq N$. Let us write $s[i \dots j]$ for the substring of s between two positions $i, j \in \{0, \dots, |s|\}$. Those indices are considered as positions *in-between* letters, so, for instance, $s = s[0 \dots |s|]$, while $s[(i-1) \dots i]$ is the i-th letter of s; note also that $s[i \dots j] \cdot s[j \dots k] = s[i \dots k]$. In particular, we have $v = s[|u| \dots |uv|]$.

Consider the following coloring of the complete graph over $V = \{|u|, \dots, |uv|\}$: the edge $(i, j) \in V^2$ with $i < j$ is given the color $\varphi(s[i \dots j])$. Since $|V| \geq N + 1 = R$, there exists a monochromatic clique $\{i_0, \dots, i_{r+2}\} \subseteq V$ with $i_0 < \cdots < i_{r+2}$.

We now define $u' = s[0 \dots i_1]$ and $w' = s[i_{r+1} \dots |s|]$, which ensures that u is a prefix of u' and w is a suffix of w' since i_1 and i_{r+1} are positions in v. For $m \in \{1, \dots, r\}$, we also take $v'_m = s[i_m \dots i_{m+1}] \neq \varepsilon$ (because $|v'_m| = i_{m+1} - i_m \geq 1$). Then $s = u'v'_1 \dots v'_r w'$, and

$$\begin{aligned}
\varphi(u'v'_1 \dots v'_m) &= \varphi(s[0 \dots i_{m+1}]) = \varphi(s[0 \dots i_0])\varphi(s[i_0 \dots i_{m+1}]) \\
&= \varphi(s[0 \dots i_0])\varphi(s[i_0 \dots i_1]) \quad \text{by monochromaticity} \\
&= \varphi(s[0 \dots i_1]) = \varphi(u')
\end{aligned}$$

and similarly, $\varphi(v'_m \dots v'_1 w') = \varphi(w')$. Thus, by definition, we have an r-split of s. ◀

▶ **Lemma 7.7.** *Let $f : \Gamma^* \to \Sigma^*$ be regular and $\varphi : \Gamma^* \to M$ be a morphism with M finite. Suppose that $\pi \circ \varphi = \nu_f$ for some other morphism $\pi : M \to \mathcal{N}(f)$. Let $r \geq 1$ and $\Pi \subseteq \Sigma$.*

We define $L(f, \Pi, \varphi, r)$ to be the set of strings that admit an r-split $s = uv_1 \dots v_r w$ according to φ such that all the triples $(uv_1 \dots v_{i-1}, v_i, v_{i+1} \dots v_r w)$ are producing with respect to (f, Π) – let us call this a producing r-split with respect to (f, Π, φ).

Then $L(f, \Pi, \varphi, r)$ is a regular language, and $\sup\{|f(s)|_\Pi \mid s \in \Gamma^* \setminus L(f, \Pi, \varphi, r)\} < \infty$.

Proof. $L(f, \Pi, \varphi, r)$ can be recognized by a non-deterministic automaton that guesses an adequate r-split and computes $\varphi(u), \varphi(v_1), \ldots, \varphi(v_r), \varphi(w)$. The hard part is showing that $|f(-)|_\Pi$ is bounded on the complement of this language.

By the previous proposition, there exists some $N \in \mathbb{N}$ such that any string $s \in \Gamma^*$ of length at least N admits an r-split according to φ. Thanks to the existence of π, it is also an r-split according to ν_f. So if this long string is in $\Gamma^* \setminus L(f, \Pi, \varphi, r)$, then it is of the form $s = uv_1 \ldots v_r w$ where, for some $i \in \{1, \ldots, r\}$, $(uv_1 \ldots v_{i-1}, v_i, v_{i+1} \ldots v_r w)$ is *not* producing. Therefore, $|f(uv_1 \ldots v_{i-1} v_{i+1} \ldots v_r w)|_\Pi = |f(uv_1 \ldots v_r w)|_\Pi$. The important part is that the argument in the left-hand side is strictly shorter (the definition of r-split contains $v_i \neq \varepsilon$). Furthermore, we claim that $s' = uv_1 \ldots v_{i-1} v_{i+1} \ldots v_r w \in \Gamma^* \setminus L(f, \Pi, \varphi, r)$. Once this is established, a strong induction on the length suffices to show that $|f(-)|_\Pi$ restricted to $\Gamma^* \setminus L(f, \Pi, \varphi, r)$ reaches its maximum at some string of length smaller than N, and thus to conclude the proof.

It remains to show that $s' \notin L(f, \Pi, \varphi, r)$. If this were false, then by definition we would have a producing r-split $s' = u'v'_1 \ldots v'_r w'$. Assuming this, we will lift this split to a producing r-split of s in order to contradict $s \notin L(f, \Pi, \varphi, r)$. We give notations to the components of our non-producing triple: $\hat{u} = uv_1 \ldots v_{i-1}$, $\hat{v} = v_i$, $\hat{w} = v_{i+1} \ldots v_r w$.

Suppose that for some $j \in \{1, \ldots, r\}$ and $x \in \Gamma^*$, we have $\hat{u} = u'v'_1 \ldots v'_{j-1} x$ and $|x| \leq |v'_j|$. Then there must exist a unique $y \in \Gamma^*$ such that $v'_j = xy$ and $\hat{w} = yv'_{j+1} \ldots v'_r w'$. What we want to show now is that $(u', v'_1, \ldots, v'_{j-1}, x\hat{v}y, v'_{j+1}, \ldots, v'_r, w')$ is a producing r-split of s.

- First, the concatenation of this sequence of length $r + 2$ is indeed equal to $\hat{u}\hat{v}\hat{w} = s$.
- Next, we have $\varphi(u'v'_1 \ldots v_{j-1}(x\hat{v}y)) = \varphi(\hat{u}\hat{v}y) = \varphi(\hat{u}\hat{v})\varphi(y) = \varphi(\hat{u})\varphi(y)$ since $(\hat{u}, \hat{v}, \hat{w})$ is a 1-split of s, and $\varphi(\hat{u})\varphi(y) = \varphi(\hat{u}y) = \varphi(u'v'_1 \ldots v'_j)$. For $k \geq j$, by multiplying by $\varphi(v'_{j+1} \ldots v'_k)$ on the right, we get $\varphi(u'v'_1 \ldots v'_{j-1}(x\hat{v}y)v'_{j+1} \ldots v'_k) = \varphi(u'v'_1 \ldots v'_k)$. Similarly, for $k \leq j$, we have $\varphi(v'_k \ldots v'_{j-1}(x\hat{v}y)v'_{j+1} \ldots v'_r w') = \varphi(v'_k \ldots v'_r w')$.
- Combining the above with the fact that $(u', v'_1, \ldots, v'_r, w')$ is an r-split of s' gives us directly from the definitions that our new $(r + 2)$-tuple with $x\hat{v}y$ is an r-split of s.
- Finally, we must check that it is producing.
  - Let $k \leq j - 1$. We must show that $(u'v'_1 \ldots v'_{k-1}, v_k, v'_{k+1} \ldots v'_{j-1}(x\hat{v}y)v_{j+1} \ldots v'_r w')$ is producing with respect to (f, Π). We have seen previously that the componentwise image by φ of this triple is the same as the one for $(u'v'_1 \ldots v'_{k-1}, v'_k, v'_{k+1} \ldots v'_r w')$. The latter is producing (since it comes from an r-split chosen to be producing), and therefore so is the former, because thanks to $\nu_f = \pi \circ \varphi$, the image by φ suffices to determine whether a triple is producing.
  - The case $k \geq j + 1$ is symmetrical.
  - The remaining case is $(u'v'_1 \ldots v'_{j-1}, x\hat{v}y, v'_{j+1} \ldots v'_r w')$. It would be convenient if $x\hat{v}y$ and v'_j had the same image by φ, but this is not guaranteed. Instead, we come back to the dichotomy concerning what happens when we remove the substring $x\hat{v}y$ in s. This can be done in two steps: first remove $\hat{v}$ in s, which gives us s', then remove $xy = v'_j$ from s', resulting in $s'' = u'v'_1 \ldots v'_{j-1} v'_{j+1} \ldots v'_r w'$. Using the fact that the r-split of s' is producing while $(\hat{u}, \hat{v}, \hat{w})$ is not, we have $|f(s)|_\Pi = |f(s')|_\Pi > |f(s'')|_\Pi$. This means that the 1-split containing $x\hat{v}y$ must be producing.

If the (j, x) chosen previously does not exist, then either u' is a prefix of $\hat{u}$ or w' is a suffix of $\hat{w}$. In those cases, there is an analogous lifting procedure, and its proof of correctness is simpler; we leave this to the reader. ◀

▶ Remark F.2. It is not clear whether the above reasoning can be made to work if we require idempotency in the definition of r-split. An analogous argument is made in the first paragraph of the proof of the original Dichotomy Lemma in [25], but we were unable to

check that $s' \notin L(f, \Pi, \varphi, r)$ when forcing the central elements of producing triples to be idempotent. Thankfully it does not seem to be required to carry out further arguments leading to a proof of Theorem F.1.

▶ **Lemma 7.9.** *Let $g : \Gamma^* \to I^*$ be a regular function and, for each $i \in I$, let $h_i : \Gamma^* \to \Sigma^*$ be comparison-free polyregular of rank at most k. Suppose that $\sup_{s \in \Gamma^*} |g(s)|_J < \infty$ where*

$$J = \begin{cases} \{i \in I \mid \mathrm{rk}(h_i) = k\} & \text{when } k \geq 1 \\ \{i \in I \mid |h_i(\Gamma^*)| = \infty\} & \text{when } k = 0 \end{cases}$$

(Morally, regular functions with finite range play the role of "comparison-free polyregular functions of rank -1".) Then $\mathrm{rk}(\mathrm{CbS}(g, (h_i)_{i \in I})) \leq k$.

Proof. We write $f = \mathrm{CbS}(g, (h_i)_{i \in I})$.

First, let us consider the case $k = 0$. For convenience, we assume w.l.o.g. that $I \cap \Sigma = \varnothing$. Let $N = \sup\{|g(s)|_J \mid s \in \Gamma^*\}$ – in the degenerate case $J = \varnothing$, this leads to $N = 0$ – and $\iota_n(s)$ be the n-th letter of J in $g(s)$ if it exists, or else ε. Then we use the equation $g(s) = \rho_0(s)\iota_1(s)\rho_1(s)\dots\iota_N(s)\rho_N(s)$ to define uniquely $\rho_0, \dots, \rho_N : \Gamma^* \to (I \setminus J)^*$. One can build for each $n \in \{0, \dots, N\}$ a sequential transducer whose composition with g yields ρ_n; therefore, since g is regular, so is ρ_n. We define $\psi_n(s)$ next as $h_{\iota_n(s)}(s)$ when $\iota_n(s) \in J$, and ε otherwise. For any $n \in \{1, \dots, N\}$, since the languages $g^{-1}(((I \setminus J)^* J)^{n-1}(I \setminus J)^* i I^*)$ are regular for all $i \in J$, this defines ψ_n as a combination of $\{s \mapsto \varepsilon\} \cup \{h_i \mid i \in J\}$ by regular conditionals, so ψ_n is regular. Finally, we set $f'(s) = \rho_0(s)\psi_1(s)\rho_1(s)\dots\psi_N(s)\rho_N(s) \in (\Sigma \cup I \setminus J)^*$; the function f' thus defined is regular by closure under concatenation (use a product construction on copyless SSTs). Observe that $f'(s)$ is obtained by substituting each occurrence of a letter $i \in J$ in $g(s)$ by $h_i(s)$ (thus, it is equal to $g(s)$ when $J = \varnothing$, and to $f(s)$ when $J = I$).

What remains to do is to substitute the letters of $I \setminus J$ to get f. To do so, let us define $L_{\vec{w}} = \{s \in \Gamma^* \mid \forall i \in I \setminus J,\ h_i(s) = w_i\}$ for $\vec{w} = (w_i)_{i \in I \setminus J} \in \prod_{i \in I \setminus J} h_i(\Gamma^*)$. The function f coincides on $L_{\vec{w}}$ with f' postcomposed with the morphism that replaces each $i \in I \setminus J$ by w_i; this is regular by closure under composition. Furthermore, the factors of $\prod_{i \in I \setminus J} h_i(\Gamma^*)$ are finite by definition of J, and $I \setminus J$ itself is a subset of the finite alphabet I. So there are finitely many $L_{\vec{w}}$, and they partition Σ^*; they are also all regular, as finite intersections of preimages of singletons by regular functions. Therefore, f is obtained by combining regular functions by a regular conditional, so it is regular, i.e. $\mathrm{rk}(f) = 0$ as we wanted.

This being done, let us move on to the case $k \geq 1$. For $i \in J$, let $h_i = \mathrm{CbS}(g'_i, (h'_{i,x})_{x \in X_i})$ where all the g'_i are regular and the $h'_{i,x}$ are of rank at most $k - 1$, choosing the X_i to be pairwise disjoint as well as disjoint from I. Let $f'(s)$ be obtained from $g(s)$ by substituting each occurrence of a letter $i \in J$ by $g'_i(s)$. For the same reasons as those exposed in the first paragraph of the case $k = 0$, this defines a regular function f'. By taking its composition by substitution with the disjoint union of the families $(h_i)_{i \in I \setminus J}$ and $(h'_{i,x})_{x \in X_i}$ for $i \in J$, we recover f. Since the functions involved in this union family are all of rank at most $k - 1$ (by definition of J), this means that $\mathrm{rk}(f) \leq k$. ◀

F.2 Wrapping up the proof of Theorem F.1

▶ **Lemma F.3.** *Let $f : \Gamma^* \to \Sigma^*$ be comparison-free polyregular. There exists a morphism to a* finite *monoid $\nu'_f : \Gamma^* \to \mathcal{N}'(f)$ such that, for any* 1-split *according to ν'_f composed of $u, v, w \in \Gamma^*$ and any $c \in \Sigma^*$, the sequence $(|f(uv^n w)|_c)_{n \in \mathbb{N}}$ is non-decreasing.*

Proof. By straightforward induction on $\mathrm{rk}(f)$, using Lemma 7.5: for $f = \mathrm{CbS}(g, (h_i)_{i\in I})$ where g is regular and the h_i are comparison-free, we take $\mathcal{N}'(f) = \mathcal{N}(g) \times \prod_{i\in I} \mathcal{N}'(h_i)$. ◀

▶ **Lemma F.4.** *Let $f : \Gamma^* \to \Sigma^*$ be a comparison-free polyregular function. Let $\varphi : \Gamma^* \to M$ be a morphism to a finite monoid and let $r \in \mathbb{N}_{\geq 1}$. Then there exists a regular language $\widehat{L}(f, \varphi, r) \subseteq \Gamma^*$ such that:*

- *the function which maps $\widehat{L}(f, \varphi, r)$ to ε and coincides with f on $\Gamma^* \setminus \widehat{L}(f, \varphi, r)$*
  - *is regular and takes finitely many values if $\mathrm{rk}(f) = 0$ i.e. f is regular;*
  - *is comparison-free polyregular with rank strictly lower than $\mathrm{rk}(f)$ otherwise;*
- *for any $s \in \widehat{L}(f, \varphi, r)$, there exist $k = \mathrm{rk}(f) + 1$ r-splits according to φ – let us write them as $s = u^{(m)} v_1^{(m)} \dots v_r^{(m)} w^{(m)}$ for $m \in \{1, \dots, k\}$ – such that, for any factorization $s = \alpha_0 \beta_1 \alpha_1 \dots \beta_k \alpha_k$ where, for some* permutation *σ of $\{1, \dots, k\}$, each β_m coincides with some $v_l^{(\sigma(m))}$ (in the sense that their positions as substrings of s are equal), we have*

$$\forall n \in \mathbb{N},\ |f(\alpha_0 \beta_1^n \alpha_1 \dots \beta_k^n \alpha_k)| \geq n^k$$

(note that in general, such factorizations $s = \alpha_0 \beta_1 \alpha_1 \dots \beta_k \alpha_k$ might not exist, for instance when $r = 1$ and all the substrings $v_1^{(m)}$ overlap)

Proof. We proceed by induction on $\mathrm{rk}(f)$.

Base case: $\mathrm{rk}(f) = 0$. In this case, f is regular. Let $\psi : \Gamma^* \to M \times \mathcal{N}(f)$ be the monoid morphism obtained by pairing φ (given in the lemma statement) with ν_f (given in Lemma 7.7). Then, using Lemma 7.7, one can see that taking $\widehat{L}(f, \varphi, r) = L(f, \Sigma, \psi, r)$ works.

Inductive case: $\mathrm{rk}(f) \geq 1$. In this case, $f = \mathrm{CbS}(g, (h_i)_{i\in I})$ for some regular $g : \Gamma^* \to I^*$ and some comparison-free polyregular $h_i : \Gamma^* \to \Sigma^*$ with $\mathrm{rk}(h_i) \leq \mathrm{rk}(f) - 1$ for all $i \in I$. Let φ and r be as given in the lemma statement. Let J be defined as in Lemma 7.9:

$$J = \begin{cases} \{i \in I \mid \mathrm{rk}(h_i) = \mathrm{rk}(f) - 1\} & \text{when } \mathrm{rk}(f) \geq 2 \\ \{i \in I \mid |h_i(\Gamma^*)| = \infty\} & \text{when } \mathrm{rk}(f) = 1 \end{cases}$$

For $i \in J$, let $\psi_i : \Gamma^* \to M \times \mathcal{N}(g) \times \mathcal{N}'(h_i)$ be obtained by combining φ with the morphisms given by Lemmas 7.5 and F.3. We shall consider the regular languages $\widehat{L}(h_i, \psi_i, r)$ provided by the inductive hypothesis.

Let us take a copy $\underline{J} = \{\underline{i} \mid i \in J\}$ of J such that $\underline{J} \cap I = \varnothing$. We define the regular function $g' : \Gamma^* \to (I \cup \underline{J})^*$ as follows: for any input $s \in \Gamma^*$, to build the output $g'(s)$, we start from $g(s)$ and then, for each $i \in J$ *such that* $s \notin \widehat{L}(h_i, \psi_i, r)$, we replace all the occurrences of i by $\underline{i}$. For $i \in J$, we also define $h_{\underline{i}}$ to be the function which maps $\widehat{L}(h_i, \psi_i, r)$ to ε and coincides with h_i on $\Gamma^* \setminus \widehat{L}(h_i, \psi_i, r)$. By construction, $f = \mathrm{CbS}(g', (h_i)_{i\in I\cup\underline{J}})$.

Note that g' is regular: it is indeed generated from g using regular conditionals and postcomposition by letter-to-letter morphisms. We can therefore build a morphism

$$\chi : \Gamma^* \to M \times \mathcal{N}(g') \times \prod_{i\in J} \mathcal{N}'(h_i)$$

in the expected way, and define the language provided by the lemma statement as

$$\widehat{L}(f, \varphi, r) = L(g', J, \chi, r)$$

According to Lemma 7.7, it is indeed a regular language. Concerning the first item of the lemma statement, the function that it considers can be expressed as

$$\mathrm{CbS}(g'', (h_i)_{i\in I\cup \underline{J}}) \qquad \text{where} \qquad g'' : s \mapsto \begin{cases} \varepsilon & \text{when } s \in L(g', J, \chi, r) \\ g'(s) & \text{otherwise} \end{cases}$$

We want to show that $\mathrm{rk}(\mathrm{CbS}(g'', (h_i)_{i\in I\cup \underline{J}})) \leq \mathrm{rk}(f) - 1$. The shape of this statement fits with the conclusion of Lemma 7.9, so we just have to check the corresponding assumptions.

- g'' is regular, by closure of regular functions under regular conditionals.
- for $i \in I \cup \underline{J}$, the function h_i is comparison-free polyregular of rank at most $\mathrm{rk}(f) - 1$:
  - for $i \in I$, this was required in our choice of expression for $f = \mathrm{CbS}(g, (h_i)_{i\in I})$ (and such a choice was possible by definition of rank);
  - for $i = \underline{j} \in \underline{J}$, we get this by applying the first item of the inductive hypothesis to h_j (indeed, the function introduced by this item is none other than $h_{\underline{j}} = h_i$).
- We also get that, *with the same J as before,*

$$J = \begin{cases} \{i \in I \cup \underline{J} \mid \mathrm{rk}(h_i) = \mathrm{rk}(f) - 1\} & \text{when } \mathrm{rk}(f) \geq 2 \text{ i.e. } \forall i \in J,\ \mathrm{rk}(h_i) \geq 1 \\ \{i \in I \cup \underline{J} \mid |h_i(\Gamma^*)| = \infty\} & \text{when } \mathrm{rk}(f) = 1 \text{ i.e. } \forall i \in J,\ \mathrm{rk}(h_i) = 0 \end{cases}$$

  using again the first item of the inductive hypothesis to handle the case of indices in $\underline{J}$.
- Finally, by definition of g'' and by Lemma 7.7, using the convention $\sup \varnothing = 0$,

$$\sup_{s\in\Gamma^*} |g''(s)|_J = \sup\{|g'(s)|_J \mid s \in \Gamma^* \setminus L(g', J, \chi, r)\} < \infty$$

Let us now check the second item concerning splits and factorizations. Let $s \in \widehat{L}(f, \varphi, r)$. By definition, there exists $i \in J$ such that $s \in L(g', i, \chi, r)$. In particular, $|g'(s)|_i \geq 1$, which entails that $s \in \widehat{L}(h_i, \psi_i, r)$ by definition of g'. The inductive hypothesis gives us a family of r-splits $s = u^{(m)}v_1^{(m)} \dots v_r^{(m)}w^{(m)}$ according to ψ for $m \in \{1, \dots, k-1\}$ – recall that $\mathrm{rk}(h_i) + 1 = \mathrm{rk}(f) = k - 1$. We complete it by taking $(u^{(k)}, v_1^{(k)}, \dots, v_r^{(k)}, w^{(k)})$ to be a producing r-split of s with respect to (g', i, χ), whose existence is guaranteed by definition of $L(g', i, \chi, r)$. Since φ factors through both ψ_i and χ by construction, this indeed gives us a family of k r-splits according to φ.

Now, let $s = \alpha_0\beta_1\alpha_1 \dots \beta_k\alpha_k$ be a factorization and σ be a permutation of $\{1, \dots, k\}$ such each β_m coincides with some $v_l^{(\sigma(m))}$ for some l. Note that from the original expression of f as a composition by substitution, we have

$$\forall s' \in \Gamma^*, \quad |f(s')| \geq |g(s')|_i \cdot |h_i(s')|$$

Therefore, our desired inequality will follow once we prove the ones below:

$$\forall n \in \mathbb{N}, \quad |g(\alpha_0\beta_1^n\alpha_1 \dots \beta_k^n\alpha_k)|_i \geq n \quad \text{and} \quad |h_i(\alpha_0\beta_1^n\alpha_1 \dots \beta_k^n\alpha_k)| \geq n^{k-1}$$

To illustrate the idea, we assume $\sigma(k) = k$, so that $\beta_k = v_l^{(k)}$ for some l, and we invite the reader to convince themself that this is merely a matter of notational convenience for the rest of the proof.

Let us start with h_i. Since ν'_{h_i} factors through χ, the triple

$$(\alpha_0\beta_1 \dots \beta_{k-1}\alpha_{k-1}, \beta_k, \alpha_k) = (u^{(k)}v_1^{(k)} \dots v_{l-1}^{(k)}, v_l^{(k)}, v_{l+1}^{(k)} \dots v_r^{(k)}w^{(k)})$$

is a 1-split according to ν'_{h_i}. Using the fact that ν'_{h_i} factors through ψ_i, one can show that $(\alpha_0\beta_1^n \dots \beta_{k-1}^n\alpha_{k-1}, \beta_k, \alpha_k)$ is still a 1-split according to ν'_{h_i}. Therefore, for $n \in \mathbb{N}$,

$$|h_i(\alpha_0\beta_1^n \dots \beta_{k-1}^n\alpha_{k-1}\beta_k^n\alpha_k)| \geq |h_i(\alpha_0\beta_1^n \dots \beta_{k-1}^n\alpha_{k-1}\beta_k\alpha_k)| \geq n^{k-1}$$

where β_k is *not* raised to the n-th power in the middle; the left inequality comes from Lemma F.3, while the right inequality is part of the induction hypothesis applied to h_i.

The case of g requires an additional step. We know that $(\alpha_0\beta_1 \dots \beta_{k-1}\alpha_{k-1}, \beta_k, \alpha_k)$ is a producing triple with respect to (g', i, χ); therefore, by Lemma 7.5,

$$\forall n \in \mathbb{N},\ |g'(\alpha_0\beta_1 \dots \beta_{k-1}\alpha_{k-1}\beta_k^n\alpha_k)|_i \geq n$$

To replace g' by g in the above inequality, recall that by definition of g', since $i \in J$,

$$\forall s' \in \Gamma^*,\ (|g'(s')|_i \neq 0 \implies |g'(s')|_i = |g(s')|_i)$$

One can then conclude by Proposition F.5 below, taking $l = k - 1$. There is a subtlety here: our definitions ensure that ν_g factors through ψ_i, but this might not be the case for $\nu_{g'}$ (because ψ_i had to be defined before g'). So for this final step, we must work with the function g, whereas to leverage the producing triple, we had to use g'. ◀

The following proposition, which we used at the end of the above proof, will also be useful to prove Theorem 8.3.

▶ **Proposition F.5.** *Let $g : \Gamma^* \to \Sigma^*$ be a regular function and $s = \alpha_0\beta_1\alpha_1 \dots \beta_l\alpha_l \in \Gamma^*$ such that every triple $(\alpha_0\beta_1 \dots \alpha_m, \beta_{m+1}, \alpha_{m+1}\beta_{m+2} \dots \alpha_l)$ is a 1-split according to ν_g. Then for every $c \in \Sigma$, the function*

$$(n_1, \dots, n_l) \mapsto |g(\alpha_0\beta_1^{n_1}\alpha_1 \dots \beta_l^{n_l}\alpha_l)|_c$$

is monotone according to the product partial order on $\mathbb{N}^l$.

Proof idea. In order to apply Lemma 7.7, the key observation is that the triple

$$(\alpha_0\beta_1^{n_1} \dots \alpha_m\beta_{m+1}^{n_{m+1}}, \beta_{m+1}, \alpha_{m+1}\beta_{m+2}^{n_{m+2}} \dots \alpha_l)$$

is also a 1-split. This is because we have, by definition of 1-split, $\nu_g(\alpha_0\beta_1^{n_1}) = \nu_g(\alpha_0\beta_1)$, then $\nu_g(\alpha_0\beta_1\alpha_1\beta_2^{n_2}) = \nu_g(\alpha_0\beta_1\alpha_1\beta_2)$, etc., and similarly on the right side. ◀

After having established Lemma F.4, we can use it to finally wrap up this section.

Proof of Theorem F.1. We apply Lemma F.4 to get a language $\widehat{L}(f, \varphi, \mathrm{rk}(f) + 1)$ where φ does not matter (take for instance the morphism from Γ^* to the trivial monoid). It must be non-empty (or else we would have the contradiction $\mathrm{rk}(f) < \mathrm{rk}(f)$), so we can choose an arbitrary element $s \in \widehat{L}(f, \varphi, \mathrm{rk}(f) + 1)$.

Let $k = \mathrm{rk}(f) + 1$. Lemma F.4 gives us k factorizations $s = u^{(m)}v_1^{(m)} \dots v_k^{(m)}w^{(m)}$ satisfying certain properties. Note that k plays two roles here that were distinct in the lemma. We claim that thanks to this, there exists a factorization $s = \alpha_0\beta_1\alpha_1 \dots \beta_k\alpha_k$ as described in Lemma F.4. This entails that setting $s_n = \alpha_0\beta_1^n\alpha_1 \dots \beta_k^n\alpha_k$ proves the theorem.

Our task is therefore to select one element in each of the k sets $\{v_l^{(m)} \mid l \in \{1, \dots, k\}\}$ of substrings of s for $m \in \{1, \dots, k\}$, such that the selected substrings are pairwise non-overlapping. There is a strategy for this which is similar to the classical greedy algorithm for computing a maximum independent set in an interval graph. We take β_1 to be the substring of s among the $v_1^{(m)}$ whose right endpoint is leftmost. One can check that β_1 cannot overlap with any $v_l^{(m)}$ for $l \geq 2$. Thus, by discarding the set to which β_1 belongs, as well as each $v_1^{(m)}$ in the other sets, we reduce the remainder of the task to our original goal with k being decremented by 1. At this stage, an induction suffices to conclude the proof. ◀

G Proofs of Theorems 6.1 and 7.1

Now that we have shown that cfp functions are closed under composition and that their asymptotic growth are tightly linked to their ranks, we have the essential ingredients to prove Theorems 6.1 and 7.1. There are a couple of preliminary lemmas helpful for both that we first prove here.

▶ **Lemma G.1.** *For any comparison-free polyregular function $f : \Gamma^* \to \Sigma^*$ and $k \geq \mathrm{rk}(f)$, there exists a regular function $f' : (\{0, \dots, \mathrm{rk}(f)\} \times \Gamma)^* \to \Sigma^*$ such that $f = f' \circ \mathtt{cfpow}_\Gamma^{(k+1)}$.*

Proof. By induction on $\mathrm{rk}(f)$ (with an inductive hypothesis that quantifies over k).

Base case (f regular). Consider the unique $\varphi \in \mathrm{Hom}((\{0, \dots, \mathrm{rk}(f)\} \times \Gamma)^*, \Gamma^*)$ such that for every $c \in \Gamma$, $\varphi(k, c) = c$ and $\varphi(m, c) = \varepsilon$ when $m < k$. Since regular functions are closed under composition, $f' = f \circ \varphi$ is regular, and the desired equation follows from the fact that $\varphi \circ \mathtt{cfpow}_\Gamma^{(k+1)} = \mathrm{id}_{\Gamma^*}$.

Inductive case. Let $f = \mathrm{CbS}(g, (h_i)_{i \in I})$ with $g : \Gamma^* \to I^*$ regular and $h_i : \Gamma^* \to \Sigma^*$ cfp such that $\mathrm{rk}(h_i) \leq \mathrm{rk}(f) - 1$ for all $i \in I$. Using the inductive hypothesis, we know that $h_i = h'_i \circ \mathtt{cfpow}_\Gamma^{(k)}$ for some family of regular functions $(h'_i)_{i \in I}$. Thus, let us assume we are given 2DFTs $\mathcal{T}$ and $(\mathcal{T}'_i)_{i \in I}$ corresponding to g and the family $(h'_i)_{i \in I}$. Without loss of generality, let us assume further that $\mathcal{T}$ always output at most one letter at each transition, never outputs a letter upon reading $\triangleleft$, and that the $\mathcal{T}_i$ always terminate on the marker $\triangleright$ by a transition that does not move the reading head. With these assumptions, let us describe informally a 2DFT $\mathcal{T}''$ corresponding to the function f' such that $\mathrm{CbS}(g, (h'_i \circ \mathtt{cfpow}_\Gamma^{(k)})_{i \in I}) = f' \circ \mathrm{CbS}(g, (h'_i)_{i \in I}) \circ \mathtt{cfpow}_\Gamma^{(k+1)}$.

Assuming that the state space of $\mathcal{T}$ is Q and the state space of $\mathcal{T}'_i$ is Q'_i, with Q and the Q'_is all pairwise disjoint, we take the state space of $\mathcal{T}''$ to be

$$Q'' \quad = \quad Q \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\} \times \left(\{\bullet\} \sqcup \bigcup_{i \in I} Q'_i \times \{\mathsf{L}, \mathsf{R}, \mathsf{S}\} \right)$$

with initial state $(q_0, \mathsf{R}, \bullet)$, if q_0 is the initial state of $\mathcal{T}$ and final states the triples $(q_f, \mathsf{M}, \bullet)$ such that q_f is a final state of $\mathcal{T}$. To guide intuitions, the elements L, R and S should be respectively read as "left", "right" and "stay". With this in mind, the high-level description of computations carried out by $\mathcal{T}''$ over words $\mathtt{pow}_\Gamma^{(k)}$ is as follows.

- When in a state $(q_0, \mathsf{M}, \bullet)$, $\mathcal{T}''$ essentially acts as $\mathcal{T}$ on letters of the shape (k, a) or end-markers and ignores letters (l, a) for $l < k$; the central component M then determines whether to seek the next relevant position to the left or to the right when reading such an irrelevant letter. This continues up until upon reading a letter (k, a) in state $(q, \mathsf{M}, \bullet)$ such that $\mathcal{T}$ would ouput i when reading a in q, $\mathcal{T}''$ moves into the state $(r, \mathsf{M}', (q'_{0,i}, \mathsf{R}))$ where $q'_{0,i}$ is the initial state of $\mathcal{T}'_i$ and (r, M') is determined by the transition in $\mathcal{T}$.
- When in a state $(q, \mathsf{M}, (q'_i, \mathsf{M}'))$ for $q'_i \in Q'_i$, $\mathcal{T}$ behaves exactly as $\mathcal{T}_i$ as long as the current transition does not reach the final state, treating letters outside of its input alphabets as end markers; this is possible because of the component M' of the state, that we use to keep track of the last move of the reading head. Meanwhile the components $q \in Q$ and M are untouched. When a final transition is taken, by our assumption we return control to $\mathcal{T}$ by going to state $(q, \mathsf{M}, \bullet)$ and moving in the direction prescribed by M (recall that, by assumption, we are moving away from (or staying in) the position at which $\mathcal{T}_i$ started running).

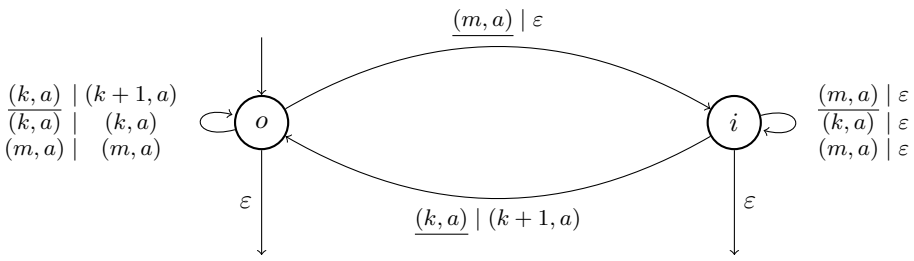
We leave formalizing this definition and checking that $\mathcal{T}''$ has a desirable behaviour to the reader. ◀

▶ **Lemma G.2.** *For every $k \in \mathbb{N}$, $\mathtt{cfpow}_\Gamma^{(k)}$ is equal to a a composition of sequential functions and squaring functions $\mathtt{cfsquaring}_\Delta$.*

Proof. We proceed by induction over k. The cases of $k = 0, 1, 2$ are immediate as $\mathtt{cfpow}_\Gamma^{(k)}$ then corresponds, up to isomorphism of output alphabet, to a constant function, the identity and $\mathtt{cfsquaring}_\Gamma$ respectively, so we focus on the inductive step. To achieve the desired result, it suffices to show that there exists a sequential function

$$f : ((\{0, \dots, k\} \times \Gamma) \cup \underline{(\{0, \dots, k\} \times \Gamma)})^* \to (\{0, \dots, k+1\} \times \Gamma)^*$$

such that $\mathtt{cfpow}_\Gamma^{(k+1)} = f \circ \mathtt{cfsquaring}_{\{0,\dots,k\}\times\Gamma} \circ \mathtt{cfpow}_\Gamma^{(k+1)}$. In fact, the sequential transducer pictured below computes such an f:



where m designates any element of $\{0, \dots, k-1\}$. ◀

Now we turn to the proofs of our main theorems.

▶ **Theorem 6.1.** *The class of comparison-free polyregular functions is the smallest class closed under usual function composition and containing both all regular functions and the functions $\mathtt{cfsquaring}_\Gamma$ (cf. Example 4.3) for all finite alphabets Γ.*

Proof of Theorem 6.1. The direct implication is obtained by combining the two lemmas above: every cfp function can be written as a composition $f \circ \mathtt{cfpow}_\Gamma^{(k)}$ for some $k \in \mathbb{N}$ and f regular by Lemma G.1, and Lemma G.2 guarantees that in turn, $\mathtt{cfpow}_\Gamma^{(k)}$ is a composition of sequential (and a fortiori regular) functions and squarings. Conversely, that cfp functions are closed under composition is proven in Appendix E, which is enough to conclude as regular functions and $\mathtt{cfsquaring}_\Gamma$ are cfp. ◀

▶ **Theorem 7.1.** *Let $f : \Gamma^* \to \Sigma^*$ and $k \in \mathbb{N}$. The following are equivalent:*

(i) *f is comparison-free polyregular with rank at most k;*

(ii) *f is comparison-free polyregular and $|f(w)| = O(|w|^{k+1})$;*

(iii) *there exists a regular function $g : (\{0, \dots, k\} \times \Gamma)^* \to \Sigma^*$ such that $f = g \circ \mathtt{cfpow}_\Gamma^{(k+1)}$, with the following inductive definition: $\mathtt{cfpow}_\Gamma^{(0)} : w \in \Gamma^* \mapsto \varepsilon \in (\varnothing \times \Gamma)^*$ and*

$$\mathtt{cfpow}_\Gamma^{(n+1)} \ : \ w \mapsto (n, w_1) \cdot \mathtt{cfpow}_\Gamma^{(n)}(w) \cdot \dots \cdot (n, w_{|w|}) \cdot \mathtt{cfpow}_\Gamma^{(n)}(w)$$

To make (ii) $\implies$ (i) more precise, if f is cfp with $\mathrm{rk}(f) \geq 1$, then it admits a sequence of inputs $w_0, w_1, \dots \in \Gamma^*$ such that $|w_n| \to +\infty$ and $|f(w_n)| = \Omega(|w_n|^{\mathrm{rk}(f)+1})$.

Proof of Theorem 7.1. We prove the circle of implications (i) $\Rightarrow$ (iii) $\Rightarrow$ (ii) $\Rightarrow$ (i). (The claim after this equivalence has already been established in Theorem F.1.)

The first implication (i) $\Rightarrow$ (iii) corresponds exactly to Lemma G.1 we just proved.

The implication (iii) $\Rightarrow$ (ii) is also relatively easy: $\mathtt{cfpow}_\Gamma^{(k)}$ is cfp (this is a consequence of Lemma G.2 and Theorem 6.1, although $\mathtt{cfpow}_\Gamma^{(k)}$ can also be shown to fit Definition 4.2 in a more elementary way) and so is $f \circ \mathtt{cfpow}_\Gamma^{(k+1)}$ by Theorem 6.1 for f regular. Furthermore, $|\mathtt{cfpow}_\Gamma^{(k+1)}(w)| = O(|w|^{k+1})$ and, since f is regular, $|f(u)| = O(|u|)$, so we have, as expected, $|(f \circ \mathtt{cfpow}_\Gamma^{(k+1)})(w)| = O(|w|^{k+1})$.

The final implication (ii) $\Rightarrow$ (i) is technically the hardest as it relies on Theorem F.1. Let $f : \Gamma^* \to \Sigma^*$ be cfp and $k \in \mathbb{N}$ such that $|f(w)| = O(|w|^{k+1})$. If f is regular, then $\mathrm{rk}(f) = 0 \leq k$. Otherwise, by Theorem F.1, there exists a sequence $(w_n)_{n\in\mathbb{N}}$ of inputs such that $|w_n| = O(n)$ and $|f(w_n)| \geq n^{\mathrm{rk}(f)+1}$. So $n^{\mathrm{rk}(f)+1} = O(n^{k+1})$, hence $\mathrm{rk}(f) \leq k$. ◀

H Comparison-free polyregular sequences

H.1 Proof of Theorem 9.2

▶ **Theorem 9.2.** *Let $s : \mathbb{N} \to \Sigma^*$ and $k \in \mathbb{N}$. The sequence s is comparison-free polyregular with $\mathrm{rk}(s) \leq k$ if and only if there exists $p > 0$ such that, for any $m < p$, there is a polynomial word expression e of star-height at most $k+1$ such that $\forall n \in \mathbb{N},\ s((n+1)p+m) = [\![e]\!](n)$.*

As announced, we prove Theorem 9.2 inductively on the rank of the sequence under consideration. The bulk of the reasoning is concentrated in the base case, stating that regular sequences are exactly the ultimately periodic combinations of pumping sequences.

▶ **Lemma H.1.** *A sequence of words $s : \mathbb{N} \to \Sigma^*$ is regular if and only if there is $m > 0$ such that for every $k < m$, there are words $u_0, \ldots, v_l, v_1, \ldots, v_l$ such that for every $n \in \mathbb{N}$, we have*

$$\forall n \in \mathbb{N},\ s((n+1)m+k) = u_0(v_1)^n \ldots (v_l)^n u_l$$

Proof. The "if" direction is straightforward, so we only prove the "only if" part of the statement. To keep notations harmonized, let us work with $f : \{a\}^* \to \Sigma^*$ such that $s(n) = f(a^n)$ for every $n \in \mathbb{N}$ and fix a copyless SST computing $f : \{a\}^* \to \Sigma^*$ whose set of states, set of registers and transition function we call Q, R and δ respectively. We use the monoid $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q$ introduced in Section 2.4, which contains $\mu = \mathtt{erase}_\Sigma(\delta(-,a))$. Since $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q$ is finite (Proposition 2.23), there is an exponent $m \in \mathbb{N} \setminus \{0\}$ such that $\mu^{\bullet m} = \mu \bullet \ldots (m \text{ times}) \ldots \bullet \mu$ is idempotent, i.e. $\mu^{\bullet m} = \mu^{\bullet 2m}$. This m is the one put forth in the lemma statement.

Let us fix $k < m$. Let $(q, \alpha) = \mu^{\bullet m}(q_0)$ where q_0 is the initial state of the SST. We have $\mu^{\bullet(m+k)} \bullet \mu^{\bullet m} = \mu^{\bullet(2m+k)} = \mu^{\bullet k} \bullet \mu^{\bullet 2m} = \mu^{\bullet k} \bullet \mu^{\bullet m} = \mu^{\bullet(m+k)}$ as usual. Therefore, $\mu^{\bullet m}(q) = (q, \beta)$ with $\alpha \bullet \beta = \alpha$ and $\beta \bullet \beta = \beta$ (the latter is because of $\mu^{\bullet 2m} = \mu^{\bullet m}$). Thus, q is the state reached by the SST after reading $a^{m(n+1)+k}$ for any $n \in \mathbb{N}$. We also have $(\delta(-,a))^{\bullet m}(q) = (q, \gamma)$ with $\gamma \in \mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$ and $\mathtt{erase}_\Sigma(\gamma) = \beta$.

Given $r \in R$, we distinguish two cases.

- First, suppose that $\beta(r) = \varepsilon$ or equivalently that $\gamma(r) \in \Sigma^*$ (in general, the codomain of γ is $(\Sigma \cup R)^*$). When the SST is in state q and reads a^m, it executes the assignment γ; when $\beta(r) = \varepsilon$, the new value of the register r is this $\gamma(r) \in \Sigma^*$ which does not depend on the old value of any register. Therefore, for all $n \in \mathbb{N}$, the content of the register r after having read $a^{m(n+1)+k}$ (starting from the initial configuration) is the constant $\gamma(r)$.

- We now treat the case where $\beta(r)$ is non-empty. By definition, $\beta \bullet \beta = \beta^* \circ \beta$ where $\beta^* \in \mathrm{Hom}(R^*, R^*)$ extends $\beta : R \to R^*$. Since we know, as a consequence of the idempotency of $\mu^{\bullet m}$, that $\beta \bullet \beta = \beta$, we have $\beta^*(\beta(r)) = \beta(r) \neq \varepsilon$.
  Let us study in general the situation $\beta^*(\rho) = \beta(r) \neq \varepsilon$ for $\rho \in R^*$. A first observation is that the letters in $\beta(r)$ cannot be found in any other $\beta(r')$ for $r' \in R \setminus \{r\}$ because β is copyless, so $\rho \notin (R \setminus \{r\})^*$. We therefore have $n \geq 1$ occurrences of r in ρ, so $\rho = \rho_0 r \dots r \rho_n$ with $\rho_0, \dots, \rho_n \notin (R \setminus \{r\})^*$. By coming back to $\beta^*(\rho) = \beta(r)$, into which we plug this expression for ρ, and using the fact that $\beta(r)$ has non-zero length, we can see that $n = 1$ and $\beta^*(\rho_0) = \beta^*(\rho_1) = \varepsilon$.
  Let us apply this to $\rho = \beta(r) = \mathtt{erase}_\Sigma(\gamma)(r)$ and lift the result to $\gamma(r)$:

  $$\gamma(r) = u_r r v_r \quad \text{for some} \quad u_r, v_r \in (\Sigma \cup \beta^{-1}(\{\varepsilon\}))^*$$

  In the previous case ($\beta(r') = \varepsilon$ for $r' \in R$), we saw that $\gamma(\beta^{-1}(\{\varepsilon\})) \subseteq \Sigma^*$. Therefore $\gamma^\odot(u_r), \gamma^\odot(v_r) \in \Sigma^*$, where $\gamma^\odot \in \mathrm{Hom}((\Sigma \cup R)^*, (\Sigma \cup R)^*)$ extends $\gamma : R \to (\Sigma \cup R)^*$ by being the identity on Σ. Since Σ^* is fixed by $\gamma^\odot$, when we iterate, we obtain

  $$\gamma^{\bullet(n+1)}(r) = (\gamma^\odot)^n \circ \gamma(r) = (\gamma^\odot(u_r))^n \cdot u_r r v_r \cdot (\gamma^\odot(v_r))^n$$

Now, let F be the final output function of the SST that computes f, and $\vec{w}_{m+k}$ be the register values after it has read a prefix a^{m+k}. Then after reading $a^{m(n+1)+k}$, the new register values are $(\gamma^{\bullet(n+1)})^\dagger(\vec{w}_{m+k})$. More precisely, the register r contains:

- $\gamma(r) \in \Sigma^*$ if $\beta(r) = \varepsilon$;
- $(\gamma^\odot(u_r))^n \cdot ((u_r r v_r)^\dagger(\vec{w}_{m+k})) \cdot (\gamma^\odot(v_r))^n$ otherwise.

These values are combined by $F(q)^\dagger$ – where q is the recurrent state we have been working with all along, and F is the final output function – to produce the output $f(a^{m(n+1)+k})$. This yields the desired shape: an interleaved concatenation of finitely many factors that are either constant, $(\gamma^\odot(u_r))^n$ or $(\gamma^\odot(v_r))^n$ for some $r \in R$. ◀

Proof of Theorem 9.2. We proceed by induction on the rank of the sequence $s : \mathbb{N} \to \Sigma^*$ under consideration. If the rank of s is 0, it is regular and we apply Lemma H.1 and the desired polynomial word expression is of the shape $u_0 \cdot (v_1)^* \dots (v_l)^* \cdot u_l$.

If the rank of s is $k+1$, thanks to the induction hypothesis and the base case above, it can be written as $\mathrm{CbS}(\llbracket e \rrbracket, (\llbracket e'_i \rrbracket)_{i \in I})$ where e is an expression over the alphabet I with star-height at most one and the e'_is expressions over Σ with star-height at most k. Without loss of generality, we may assume that that terminal nodes of polynomial word expressions are words of length at most one. For such an expression over alphabet I, one may define inductively the following *substitution operation* to obtain an expression of Σ^*:

$$\begin{aligned} j[(e'_i)_{i \in I}] &= e'_j & \qquad \varepsilon[(e'_i)_{i \in I}] &= \varepsilon \\ (f \cdot f')[(e'_i)_{i \in I}] &= f[(e'_i)_{i \in I}] \cdot f'[(e'_i)_{i \in I}] & \qquad f^*[(e'_i)_{i \in I}] &= (f[(e'_i)_{i \in I}])^* \end{aligned}$$

One can then check by induction on the structure of e that $\llbracket e[(e'_i)_{i \in I}] \rrbracket = \mathrm{CbS}(\llbracket e \rrbracket, (\llbracket e'_i \rrbracket)_{i \in I})$ and that $e[(e'_i)_{i \in I}]$ has star-height bounded by $k+1$. ◀

H.2 Proof of Corollary 9.3

We finally show that cfp sequences are closed by post-composing with functions $\mathbf{map}(f)$ for f cfp.

▶ **Corollary 9.3.** *If $f : \Gamma^* \to \Sigma^*$ and $s : \mathbb{N} \to (\Gamma \cup \{\#\})^*$ are cfp, so is $\mathbf{map}(f) \circ s$.*

We first prove the result for poly-pumping sequences.

▶ **Lemma H.2.** *If $[\![e]\!] : \mathbb{N} \to (\Gamma \cup \{\#\})^*$ is a poly-pumping sequence and $f : \Gamma^* \to \Sigma^*$ is comparison-free polyregular, then $\mathbf{map}(f) \circ [\![e]\!]$ is a cfp sequence.*

For the rest of this subsection, we write S for the successor function $n \mapsto n+1$ over $\mathbb{N}$. We will use the fact that s is a cfp sequence iff $s \circ \mathsf{S}$ also is.

Proof. We first note that if the separator $\#$ does not occur at any leaf of e, then the result is immediate as we would have $\mathbf{map}(f) \circ [\![e]\!] = f \circ [\![e]\!]$. We thus focus on the cases when it does occur, and proceed inductively over e.

- If $e = w \in (\Gamma \cup \{\#\})^*$, then $\mathbf{map}(f) \circ [\![e]\!]$ is a constant sequence, which is obviously cfp.
- If $e = (e')^*$, with $\#$ occuring in e', let $h_{\mathrm{l}}, h_{\mathrm{r}} : \mathbb{N} \to \Gamma^*$ and $h_{\mathrm{c}} : \mathbb{N} \to (\Gamma \sqcup \{\#\})^*$ be the sequences such that

  $$[\![e']\!] \circ \mathsf{S} \circ \mathsf{S} = h_{\mathrm{l}} \cdot \# \cdot h_{\mathrm{c}} \cdot \# \cdot h_{\mathrm{r}}$$

  with $h_{\mathrm{l}}(n)$ being the largest $\#$-free prefix of $[\![e']\!](n+2)$ and $h_{\mathrm{r}}(n)$ the largest $\#$-free suffix of $[\![e']\!](n+2)$. There is a regular function

  $$\begin{array}{rccl} f' : & (\Gamma \sqcup \{\#\})^* & \to & (\Gamma \sqcup \{\#\})^* \\ & w_0\#w_1\#\dots w_{n-1}\#w_n & \mapsto & w_1\#\dots w_{n-1} \quad (w_0,\dots,w_n \in (\Gamma \sqcup \underline{\Gamma})^*) \end{array}$$

  stripping away the first and last component of its input, so that it satisfies

  $$f' \circ \mathbf{map}(f) \circ [\![e']\!] \circ \mathsf{S} \circ \mathsf{S} \quad = \quad \mathbf{map}(f) \circ h_{\mathrm{c}}$$

  By the inductive hypothesis, we know that $\mathbf{map}(f) \circ [\![e']\!]$ is comparison-free polyregular. We may therefore conclude by composition (cf. Theorem 6.1) that $\mathbf{map}(f) \circ h_{\mathrm{c}}$ is cfp. One can check analogously that h_{l} and h_{r} are also cfp. Then observe that

  $$\begin{array}{rcl} ([\![e]\!] \circ \mathsf{S} \circ \mathsf{S})(n) & = & (h_{\mathrm{l}} \cdot \# \cdot h_{\mathrm{c}} \cdot \# \cdot h_{\mathrm{r}})(n)^{n+2} \\ & = & (h_{\mathrm{l}} \cdot (\# \cdot h_{\mathrm{c}} \cdot \# \cdot h_{\mathrm{r}} \cdot h_{\mathrm{l}})^{n+1} \cdot \# \cdot h_{\mathrm{c}} \cdot \# \cdot h_{\mathrm{r}})(n) \end{array}$$

  which means that we have

  $$\mathbf{map}(f) \circ [\![e]\!] \circ \mathsf{S} \circ \mathsf{S} = \begin{cases} (f \circ h_{\mathrm{l}}) \\ \cdot \\ (\# \cdot (\mathbf{map}(f) \circ h_{\mathrm{c}}) \cdot \# \cdot (f \circ (h_{\mathrm{r}} \cdot h_{\mathrm{l}})))^* \\ \cdot \\ \# \cdot (\mathbf{map}(f) \circ h_{\mathrm{c}}) \cdot \# \cdot (f \circ (h_{\mathrm{r}} \cdot h_{\mathrm{l}})) \\ \cdot \\ \# \cdot (\mathbf{map}(f) \circ h_{\mathrm{c}}) \cdot \# \cdot (f \circ h_{\mathrm{r}}) \end{cases}$$

  Thanks again to the closure under composition, each component of this expression is cfp, so $\mathbf{map}(f) \circ [\![e]\!] \circ \mathsf{S} \circ \mathsf{S}$ is also cfp. Hence, so is $\mathbf{map}(f) \circ [\![e]\!]$.
- The last case where $e = e' \cdot e''$ is handled similarly after a case analysis determining whether $\#$ occurs only in e', e'' or in both; we leave it to the reader. ◀

Proof of Corollary 9.3. Suppose we are given $f : \Gamma^* \to \Sigma^*$ and $s : \mathbb{N} \to (\Gamma \cup \{\#\})^*$ cfp. By Theorem 9.2, s is an ultimately periodic combination of poly-pumping sequences, so that there are $m > 0$ and some expressions $e_0, \dots, e_{m-1}$ such that $s(m(n+1)+k) = [\![e_k]\!](n)$ for every $k < m$. By Lemma H.2, every $\mathbf{map}(f) \circ [\![e_k]\!]$ is cfp. The set $L_k = \{m(n+1)+k \mid n \in \mathbb{N}\}$ is semi-linear, i.e., corresponds to a regular language, and there are regular sequences $r_k : \mathbb{N} \to \mathbb{N}$ such that $r_k(m(n+1)+k) = n$. Further, $\mathbb{N} = \{n \mid n < m\} \cup \bigcup_{k<m} L_k$, so we may use the regular conditional provided by Proposition 4.8 to show that the combination of the $\mathbf{map}(f) \circ [\![e_k]\!] \circ r_k$ and the first m values of $\mathbf{map}(f) \circ s$, which corresponds exactly to $\mathbf{map}(f) \circ s$, is indeed cfp. ◀

I Separation results

I.1 Proof of Theorem 8.1

▶ **Theorem 8.1.** *There exist comparison-free polyregular functions which are not HDT0L:*
- **(i)** *the function* $a^n \in \{a\}^* \mapsto (a^n b)^{n+1} \in \{a,b\}^*$ *for* $a \neq b$*;*
- **(ii)** *the function* $w \in \Sigma^* \mapsto w^{|w|}$ *for* $|\Sigma| \geq 2$ *(a simplification of Example 4.3);*
- **(iii)** (from [13, §6]) *the cfp functions that map* $a^n \# w \in \Sigma^*$ *to* $(w\#)^n$ *for* $a, \# \in \Sigma$*,* $a \neq \#$*.*

These examples are comparison-free. We have seen in Example 4.3 that $w \mapsto w^{|w|}$ is a comparison-free polyregular function. For the other examples:
- $(a^n \mapsto (a^n b)^{n+1}) = \mathrm{CbS}((a^n \mapsto a^{n+1}), (a^n \mapsto a^n b)_{i \in \{a\}})$ is obtained as a composition by substitution of *sequential functions*, i.e. functions computed by sequential transducers (cf. Section 2.2), which are in particular regular;
- for an alphabet Σ with $a, \# \in \Sigma$, there exist sequential functions $f : \Sigma^* \to \{a\}^*$ and $g : \Sigma^* \to \Sigma^*$ such that $f(a^n \# w) = a^n$ and $g(a^n \# w) = w\#$ for $n \in \mathbb{N}$ and $w \in \Sigma^*$, so that $\mathrm{CbS}(f, (g)_{i \in \{a\}})(a^n \# w) = (w\#)^n$.

(i) is not HDT0L. Let us fix a HDT0L system $(\{a\}, \{a,b\}, \Delta, d, (h)_{i \in \{a\}}, h')$ and show that it does not compute $a^n \mapsto (a^n b)^{n+1}$. Let $\texttt{letters}(w)$ be the set of letters occurring in the string w at least once. By the infinite pigeonhole principle, there exists an infinite $X \subseteq \mathbb{N}$ such that $\texttt{letters}(h^n(d))$ has the same value Δ' for all $n \in X$. Let us do a case analysis:
- Suppose first that for some $r \in \Delta'$ and some $m \in \mathbb{N}$, the letter b appears twice in $h' \circ h^m(r)$; in other words, that the latter contains a factor $ba^k b$ for some $k \in \mathbb{N}$. Then for all $n \in X$, $h' \circ h^{m+n}(d) \in \Sigma^* ba^k b \Sigma^*$. Since X is infinite, this holds for some n such that $m + n > k$, so that this word – i.e. the output of the HDT0L system for a^{m+n} – is different from $(a^{m+n} b)^{m+n+1} \notin \Sigma^* ba^k b \Sigma^*$.
- Otherwise, for all $r \in \Delta'$ (that includes the degenerate case $\Delta' = \varnothing$) and all $m \in \mathbb{N}$, there is at most one occurrence of b in $h' \circ h^m(r)$. Then for all $m \in \mathbb{N}$, the length of $h^{\min(X)}(d)$ bounds the number of occurrences of b in $h' \circ h^{m+\min(X)}(d)$, and this bound is independent of m. On the contrary, in the sequence $((a^n b)^{n+1})_{n \geq m+\min(X)}$, the number of occurrences of b is unbounded.

(ii) is not HDT0L. The second counterexample, namely $w \mapsto w^{|w|}$, reduces to the first one: indeed, $(a^n b)^{n+1} = (a^n b)^{|a^n b|}$ for all $n \in \mathbb{N}$, which can also be expressed as

$$(w \mapsto w^{|w|}) \circ (u \in \{a\}^* \mapsto ub) = (a^n \mapsto (a^n b)^{n+1})$$

Suppose for the sake of contradiction that there is a HDT0L system $(\Sigma, \Sigma, \Delta, d, (h_c)_{c \in \Sigma}, h')$ that computes $w \mapsto w^{|w|}$ with $|\Sigma| \geq 2$; we may assume without loss of generality that $a, b \in \Sigma$. Then $(\{a\}, \{a,b\}, \Delta, h_b(d), (h_a)_{c \in \{a\}}, h')$ computes $a^n \mapsto (a^n b)^{n+1}$.

(iii) is not HDT0L. (This is claimed without proof in [13, Section 6].)

Let $\Sigma \supseteq \{a, \#\}$ be an alphabet and let $(\Sigma, \Sigma, \Delta, d, (h_c)_{c \in \Sigma}, h')$ be a HDT0L system. We reuse a similar argument to our treatment of the counterexample **(i)**. Let the sets $\Delta' \subseteq \Delta$ and $X \subseteq \mathbb{N}$ with X infinite be such that $\texttt{letters}(h_a^n(d)) = \Delta'$ for all $n \in X$.
- Suppose first that for some $r \in \Delta'$ and some $m \in \mathbb{N}$, the string $h' \circ h_a^m \circ h_\#(r)$; contains a factor $\# \cdot a^k \cdot \#$ for some $k \in \mathbb{N}$. Then for all $n \in X$, the given HDT0L system maps $a^m \# a^n$ to a string in $\Sigma^* \cdot \# \cdot a^k \cdot \# \cdot \Sigma^*$. For $n > k$, this language does not contain $(a^n \#)^m$; such a $n \in X$ exists because X is infinite.

- Otherwise, for any $m \in \mathbb{N}$, since # occurs at most once in $h' \circ h_a^m \circ h_\#(r)$ for $r \in \Delta'$, the output of the HDT0L system has at most $|h^{\min(X)}(d)|$ occurrences of # on input $a^m \# a^{\min(X)}$. Therefore, for large enough m, this output is different from $(a^{\min(X)}\#)^m$.

I.2 Proof of Theorem 8.3

Let us recall the theorem.

▶ **Theorem 8.3.** *Some HDT0L transductions are polyregular but not comparison-free:*
(i) $f : a^n \in \{a\}^* \mapsto ba^{n-1}b \dots baabab$ *(with $f(\varepsilon) = \varepsilon$ and $f(a) = b$);*
(ii) $\mathbf{map}(a^n \mapsto a^{n\times n}) : a^{n_1}\# \dots \# a^{n_k} \mapsto a^{n_1\times n_1}\# \dots \# a^{n_k\times n_k}$ *(cf. Definition 3.8).*
(i) and (ii) are proven separately.

I.2.1 Proof of Theorem 8.3 item (i)

As mentioned in the body of the paper, this is proven by showing that the lengths of blocks of $baa \dots aab$, or equivalently, maximal blocks of a in the output of a given regular sequence is determined by a finite number of polynomial expressions. Let us formalize this notion.

▶ **Definition I.1.** *Let Σ be a finite alphabet and $c \in \Sigma$. Call $\beta_c : \Sigma^* \to \mathcal{P}(\mathbb{N})$ the function assigning to a word w the set of lengths of its maximal factors lying in $\{c\}^*$ (including ε):*

$$\beta_c(w) \quad = \quad \{k \in \mathbb{N} \mid w \in (\Sigma^* \setminus (\Sigma^* \cdot c)) \cdot c^k \cdot (\Sigma^* \setminus (c \cdot \Sigma^*))\}$$

We say that a sequence $s : \mathbb{N} \to \Sigma^*$ is poly-uniform *if for every $c \in \Sigma$ there exists a* finite *set of polynomials $A_{s,c} \subseteq \mathbb{Q}[X]$ such that, for every $n \in \mathbb{N}$,*

$$\beta_c(s(n)) \ \subseteq \ A_{s,c}(n) = \{P(n) \mid P \in A_{s,c}\}$$

▶ **Lemma I.2.** *Every comparison-free polyregular sequence $f : \mathbb{N} \to \Sigma^*$ is poly-uniform.*

Proof. First, observe that any ultimately periodic combination of poly-uniform sequence is poly-uniform. Indeed, assume that we have such a sequence s and $m > 0$ so that $n \mapsto s(m(n+1)+k)$ is poly-uniform for every k, and finite sets $A_{k,c} \subseteq \mathbb{Q}[X]$ so that $\beta_c(s(m(n+1)+k)) \subseteq A_{k,c}(n)$. Then we have

$$A_{s,c} \quad = \quad \bigcup_{l<m} \left\{ P\left(\frac{X-l}{m}\right) \mid P \in A_{k,c} \right\} \cup \beta_c(s(l))$$

witnessing that s is poly-uniform.

Hence, by Theorem 9.2, it suffices to show that poly-pumping sequences are all poly-uniform. We proceed by induction over polynomial word expressions e, defining suitable finite sets of polynomials $A_{e,c}$ for $c \in \Sigma$ such that $\beta_c([\![e]\!](n)) \subseteq A_{e,c}(n)$ and $0 \in A_{e,c}$:

$$\begin{array}{lcl} A_{e\cdot e',c} & = & \{P+Q \mid (P,Q) \in A_{e,c} \times A_{e',c}\} \\ A_{e^*,c} & = & A_{e,c} \cup \{XP \mid P \in A_{e,c}\} \end{array} \qquad \begin{array}{lcl} A_{w,c} & = & \beta_c(w) \cup \{0\} \\ & & \end{array}$$

◀

We can now conclude the proof of the first item of Theorem 8.3 by observing that the function $f : a^n \mapsto ba^{n-1}b \dots bab$ does *not* correspond to a poly-uniform sequence: $\beta_c(f(a^n)) = \{0, \dots, n-1\}$ is unbounded, and thus cannot be covered by a finite set of functions, let alone polynomials in $\mathbb{Q}[X]$.

I.3 Proof of Theorem 8.3 item (ii)

Suppose for the sake of contradiction that $f = \mathbf{map}(a^n \mapsto a^{n\times n})$ is comparison-free. Using Theorem 7.1, it must then have rank 1 since $|f(w)| = O(|w|^2)$. Thus, we may write $f = \mathrm{CbS}(g, (h_i)_{i\in I})$ where $g : \{a, \#\}^* \to I^*$ and all the $h_i : \{a, \#\}^* \to \{a, \#\}^*$ are regular.

For each $J \subseteq I$ and $k \in \{0, \dots, |I|\}$ (though the definition would make sense for $k \in \mathbb{N}$), let $\rho_{J,k} : \{a^*\} \to (I \setminus J)^*$ be uniquely defined by the condition

$$\forall w \in \{a, \#\}^*,\ \rho_{J,k}(w) = \begin{cases} s & \text{when} \quad g(w) \in ((I \setminus J)^* J)^k \cdot s \cdot (\{\varepsilon\} \cup JI^*) \\ \varepsilon & \text{when} \quad |g(w)|_J < k \end{cases}$$

(recall from Appendix F the notation $|\cdot|_J$). To put it plainly, $\rho_{J,k}(w)$ is the k-th block of letters from $I \setminus J$ that appears in $g(w)$ (the block may be the empty string if there are consecutive letters from J), or the empty string if this k-th block does not exist. The function $\rho_{J,k}$ is regular because it is the composition of a sequential function with g.

We reuse some tools from Appendix F, especially the notion of producing 1-split from Lemma 7.7. There is a unique sensible way to combine the morphisms $\nu_{f'} : \{a, \#\}^* \to \mathcal{N}(f')$ given by this lemma into a morphism

$$\varphi : \{a, \#\}^* \to \prod_{f' \in \mathcal{F}} \mathcal{N}(f') \quad \text{for} \quad \mathcal{F} = \{g\} \cup \{h_i \mid i \in I\} \cup \{\rho_{J,k} \mid J \subseteq I,\ k \in \{0, \dots, |I|\}\}$$

Note that the codomain above is a finite monoid: this allows us to apply Proposition 7.8 to this morphism φ and $r = 1$, which gives us some $N \in \mathbb{N}$. Let $s = a^N(\#a^N)^{|I|}$. For each $m \in \{0, \dots, |I|\}$, we apply the proposition to the factorization $s = u_k v_k w_k$ with $u_k = (a^N\#)^k$, $v_k = a^N$ and $w_k = (\#a^N)^{|I|-k}$ to get a 1-split $s = u'_k v'_k w'_k$ according to φ where u_k is a prefix of u'_k and w_k is a suffix of w'_k. Let $p_k = |v_k| \neq 0$ and $q_k = N - |v_k|$.

For $f' \in \mathcal{F}$ (the finite set of functions introduced above), we then define

$$\widetilde{f'} : (n_0, \dots, n_{|I|}) \in \mathbb{N}^{|I|+1} \mapsto f'\left(a^{n_0 p_0 + q_0} \# \dots \# a^{n_{|I|} p_{|I|} + q_{|I|}}\right)$$

Thanks Proposition F.5 and to the 1-split conditions that we made sure to get previously, we see that for each letter c in the codomain of f' (either $\{a, \#\}$ or I), $|\widetilde{f'}|_c : \mathbb{N}^{|I|+1} \to \mathbb{N}$ is monotone for the product partial order. Since $f = \mathbf{map}(a \mapsto a^{n\times n}) = \mathrm{CbS}(g, (h_i)_{i\in I})$,

$$\forall x \in \mathbb{N}^{|I|+1},\ \sum_{i\in I} |\widetilde{g}(x)|_i \cdot \left|\widetilde{h_i}(x)\right|_\# = \left|\widetilde{f}(x)\right|_\# = |I|$$

$$\text{where} \qquad \widetilde{f} : (n_0, \dots, n_{|I|}) \in \mathbb{N}^{|I|+1} \mapsto f\left(a^{n_0 p_0 + q_0} \# \dots \# a^{n_{|I|} p_{|I|} + q_{|I|}}\right) = a^{(n_0 p_0 + q_0)^2} \# \dots \# a^{(n_{|I|} p_{|I|} + q_{|I|})^2}$$

Since $|\widetilde{g}|_i$ and $|\widetilde{h_i}|_\#$ are monotone for all $i \in I$, and $\mathbb{N}^{|I|+1}$ admits a minimum $(0, \dots, 0)$, the fact that the above sum is constant means that, for each $i \in I$,

- either one of $|\widetilde{g}|_i$ and $|\widetilde{h_i}|_\#$ is constant equal to 0,
- or both are non-zero constant.

Let $J^\# \subseteq I$ be the set of indices that fit the second case. We claim that for $i \in J^\#$, the constant value taken by $|\widetilde{h_i}|_\#$ must be 1. If this were not the case, then for all $n \in \mathbb{N}$, there would be a substring of the form $\#a \dots a\#$ in $|\widetilde{h_i}(n, \dots, n)|_\#$, and since $f = \mathrm{CbS}(g, (h_i)_{i\in I})$ and $|\widetilde{g}(n, \dots, n)|_i \neq 0$, it would also be a substring of $\widetilde{f}(n, \dots, n)$ with length at most $|\widetilde{h_i}(n, \dots, n)|_\# = O(n)$ (since h_i is regular). This is impossible: for $k \in \{0, \dots, |I|\}$, the k-th substring of this form in $\widetilde{f}(n, \dots, n)$ has length $(np_k + q_k)^2 + 2 = \Theta(n^2)$.

Combining this with the above equation for $|\widetilde{f}|_{\#}$, we see that $|\widetilde{g}|_{J^{\#}}$ is the constant function equal to $|I|$. Let us abbreviate $\rho_k = \rho_{J^{\#},k} \in \mathcal{F}$ (recall that we defined it at the beginning of this proof) for $k \in \{0, \dots, |I|\}$; then

$$\forall x \in \mathbb{N}^{|I|+1},\ \exists! \iota_1(x), \dots, \iota_{|I|}(x) \in J^{\#} : \widetilde{g}(x) = \widetilde{\rho}_0(x)\iota_1(x)\widetilde{\rho}_1(x) \dots \iota_{|I|}(x)\widetilde{\rho}_{|I|}(x)$$

Using this, we define $h'_k(x) = \widetilde{h}_{\iota_k(x)}(x)$ for $x \in \mathbb{N}^{|I|+1}$ and $k \in \{1, \dots, |I|\}$, plus two edge cases $h'_0 : x \mapsto \varepsilon$ and $h'_{|I|+1} : x \mapsto \varepsilon$.

Let $k \in \{0, \dots, |I|\}$. Write $\vec{e}_k = (0, \dots, 0, 1, 0, \dots, 0) \in \mathbb{N}^{|I|+1}$ for the k-th vector of the canonical basis of $\mathbb{Q}^{|I|+1}$. By looking again at the k-th substring of the form $\#a \dots a\#$ in $\widetilde{f}(x)$, with $x = n\vec{e}_k$ here, we get

$$\forall n \in \mathbb{N},\ (np_k + q_k)^2 + 2 \leq |h'_k(n\vec{e}_k)| + \sum_{i \in I} |\widetilde{\rho}_k(n\vec{e}_k)|_i \cdot \left|\widetilde{h}_i(n\vec{e}_k)\right| + |h'_{k+1}(n\vec{e}_k)|$$

Note that all the lengths involved in the right-hand side above are linearly bounded in n because of the regularity of the functions involved. So there must exist $i_k \in I$ such that both $|\widetilde{\rho}_k(n\vec{e}_k)|_{i_k}$ and $|\widetilde{h}_{i_k}(n\vec{e}_k)|$ are unbounded: otherwise, the whole RHS would be $O(n)$, contradicting the $\Omega(n^2)$ lower bound induced by the above inequality.

We thus get a finite sequence of indices $i_0, \dots, i_{|I|} \in I$. By the pigeonhole principle, there must exist $k, l \in \{0, \dots, |I|\}$ such that $k \neq l$ and $i_k = i_l$; we call i this common value. Let $m \in \mathbb{N}$ be such that $|\widetilde{\rho}_k(m\vec{e}_k)|_i \geq 1$. By monotonicity (since $\rho_k, h_i \in \mathcal{F}$):

- $|\widetilde{\rho}_k(m\vec{e}_k + n\vec{e}_l)|_i \geq 1$ for all $n \in \mathbb{N}$;
- $\left|\widetilde{h}_i(m\vec{e}_k + n\vec{e}_l)\right|$ is unbounded when $n \to +\infty$.

The product of those two quantities is a lower bound for the length of the k-th substring of the form $\#a \dots a\#$ in $\widetilde{f}(m\vec{e}_k + n\vec{e}_l)$, which contradicts the fact that this length does not depend on n (it is equal to $(mp_k + q_k)^2 + 2$).