Comparison-free polyregular functions

Lê Thành Dũng Tito Nguyễn, Camille Noûs, Pierre Pradic

Comparison-free polyregular functions

Lê Thành Dũng Nguyễn
Laboratoire d'informatique de Paris Nord, Villetaneuse, France
nltd@nguyentito.eu

Camille Noûs
Laboratoire Cogitamus
https://www.cogitamus.fr/camilleen.html

Pierre Pradic
Department of Computer Science, University of Oxford, United Kingdom
https://www.cs.ox.ac.uk/people/pierre.pradic/
pierre.pradic@cs.ox.ac.uk

Abstract

We study automata-theoretic classes of string-to-string functions whose output may grow faster than linearly in their input. Our central contribution is to introduce a new such class, with polynomial growth and three equivalent definitions: the smallest class containing regular functions and closed under a "composition by substitution" operation; a restricted variant of pebble transducers; 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. Other properties of our new function class that we show are incomparability with HDT0L transductions and closure under composition.

Finally, we look at 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 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, linear logic

Acknowledgements Thanks to Mikołaj Bojańczyk, Gaëtan Douéneau-Tabot and Amina Doumane for explaining some features of their work to the authors.

A table of contents is provided in the last page, after the appendix.

1 Introduction

The theory of transducers (as described in the surveys [12, 20]) 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* are thus named because they have (at most) polynomial growth and include regular functions (the most expressive of the traditional string-to-string transduction classes). They were defined in 2018 [3] 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 [19]. A subsequent work [5] gave a logical characterization base 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* 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 [11, 13, 8]. 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 have been characterized using two transducer models [8]. One of them, the *k-marble transducers* (where $k \in \mathbb{N}$ depends on the function to be computed), is a restricted variant of pebble transducers; it follows (although this is not explicitly stated in [8]) that a HDT0L transduction has polynomial growth if and only if it is polyregular. Moreover, as claimed in [8, 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 prove a few results on the above classes. For instance, we supply a proof for the aforementioned claim of [8, Section 6], and show that the polyregular functions are exactly those computable by compositions of k-marble transducers. 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:

- the smallest class containing regular functions and closed under a certain "composition by substitution" operation;
- a restriction on pebble transducers – we disallow comparing the positions of a transducer's multiple reading heads, hence the name *comparison-free polyregular functions*;
- the functions expressible in a certain way in a *λ-calculus with linear types*, obtained by tweaking a similar characterization of regular functions [21] (this notion of *linearity* in programming languages is analogous to "copylessness" or "single use restrictions" in various transducer models).

We believe that each of those definitions constitute self-evidently natural classes of string functions to investigate[1] We use the first as our official definition of comparison-free polyregular functions throughout the paper, and show it to be equivalent to the other two.

Interestingly, our starting point in this investigation was the λ-calculus based definition. Our initial goal was to extend our characterization of regular functions by terms of the *$\lambda\ell^{\oplus\&}$-calculus* in [21] to terms that are allowed to duplicate their input strings. Since the $\lambda\ell^{\oplus\&}$-calculus does not feature primitive constructs to operate on strings, inputs and outputs are presented via *Church encodings*. This means in particular that such characterizations do not "know" anything *a priori* about transducer models; they may well be seen as results in programming language theory, answering arguably natural questions about the expressiveness of the $\lambda\ell^{\oplus\&}$-calculus.

Properties By the first definition above, comparison-free polyregular functions are indeed polyregular, while the second item implies that regular functions are comparison-free. We

[1] While they do not, to the best of our knowledge, appear in the litterature, we were told that the definition based on comparison-free pebble transducers was already considered (Mikołaj Bojánczyk, personal communication).

rule out inclusions involving the other classes that we mentioned by proving some *separation results*: there exist

- comparison-free polyregular functions that are not HDT0L (we take one example from [8]),
- and a polynomially bounded HDT0L transduction which is not comparison-free – this relies on a technical property on the maximal factors in the output of comparison-free polyregular functions over unary alphabets, which is violated by the example.

From the third definition, it quickly follows that our new function class is *closed under composition*. But this point of view hides some difficulties: to show that this characterization based on the $\lambda\ell^{\oplus\&}$-calculus is equivalent to the other ones, we invoke the heavyweight machinery of categorical semantics of programming languages, whose relationship with transducers was investigated in [21]. We also give a (non-trivial) purely automata-theoretic proof for closure under composition, avoiding the detour through this machinery.

Plan of the paper We start by recalling in detail the definitions and main results from the above-mentioned previous work (§2), and follow this with some minor complements (§3). Comparison-free polyregular functions are then defined as an extension of the class of regular functions by an operator we dub "composition by substitutions" in Section 4, and their properties are presented in Sections 4 and 5. After this, we present the alternative characterizations (§6, §7) and close the paper with a concluding section (§8). Most of the proofs do not appear in the main text, and may be found in the appendix.

2 Preliminaries

Notations By convention, $0 \in \mathbb{N}$. We write $|w|$ for the length of a string $w \in \Sigma^*$ and either w_i or $w[i]$ for its i-th letter ($i \in \{1, \dots, |w|\}$). 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 [16] 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 [10], one can find (multi-valued) string functions defined by ET0L systems.

More recently, Ferté, Marin and Sénizergues [11] provided alternative characterizations of the string-to-string functions that *HDT0L systems* can express – what we call here *HDT0L transductions*. (Those characterizations, by catenative recurrent equations and higher-order pushdown transducers of level 2, had previously been announced in an invited paper by Sénizergues [25].) Later work by Filiot and Reynier [13] and then by Douéneau-Tabot, Filiot and Gastin [8] – that does not build on [25, 11] – proved the equivalence with, respectively, copyful SSTs (Definition 2.3) and unbounded marble transducers (not presented here).

▶ **Definition 2.1** (following [13]). *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 [25, 11] 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[3] *by u_r.*

A register assignment[4] *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** ([13])**.** *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 $\Sigma^* \to \Gamma^*$ 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 $(q_i)_{0\leq i\leq n}$ (sometimes called the run *of the transducer over the input word) is inductively defined, starting from the fixed initial state q_0, by $q_i = \delta_{\mathrm{st}}(q_{i-1}, w_i)$.*

Here is an example of a copyful SST which will be useful later.

▶ **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

$$(X \mapsto \mathtt{4}X,\ Y \mapsto \underline{\mathtt{4}}XY)^\dagger \circ \dots \circ (X \mapsto \mathtt{1}X,\ Y \mapsto \underline{\mathtt{1}}XY)^\dagger(\varepsilon, \varepsilon) = (\mathtt{4321}, \underline{\mathtt{4}}\mathtt{321}\underline{\mathtt{3}}\mathtt{21}\underline{\mathtt{2}}\mathtt{11}\underline{\mathtt{1}})$$

Since $F(q) = Y$, the function defined by $\mathcal{T}$ maps $\mathtt{1234}$ to $\underline{\mathtt{4}}\mathtt{321}\underline{\mathtt{3}}\mathtt{21}\underline{\mathtt{2}}\mathtt{11}\underline{\mathtt{1}} \in (\Gamma \cup \underline{\Gamma})^* = \Sigma^*$.

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

▶ **Theorem 2.5** ([13])**.** *A function $\Gamma^* \to \Sigma^*$ can be computed by a copyful SST if and only if it can be specified by a HDT0L system.*

---

[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] 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.

[4] Some papers e.g. [7, 8] call register assignments *substitutions*. We avoid this name since it differs from its meaning in the context of our "composition by substitution" operation or of the λ-calculus.

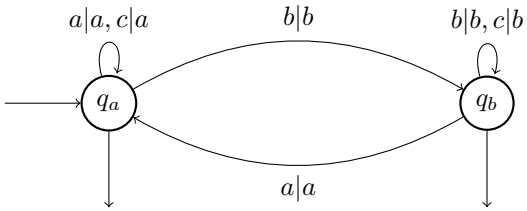

Figure 1 An example of sequential transducer.

▶ Remark 2.6. As observed in [13], it is straightforward to translate a HDT0L system into a *single-state* SST. There is a "reversal": the initial register values correspond to the final morphisms, while the final output function corresponds to the initial word. Morally, SSTs are HDT0L systems endowed with a finite set of states and the additional ability to access the letters of the output alphabet; Theorem 2.5 is essentially a state elimination result. A direct translation from SSTs to single-state SSTs is given by Benedikt et al. in a paper on polynomial automata [2, Proposition 8].

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.

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

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 2.9** (Iterated reverse [3, 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 $\texttt{reverse}(u_1)\| \ldots \|\texttt{reverse}(u_n)$.

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 can produce output. For instance, the one in Figure 1 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). 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* [3].

▶ **Definition 2.10** ([3]). *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 1);*
- *the* iterated reverse *function of Example 2.9, 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. [8].

▶ **Theorem 2.11** ([8]). *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 2.12** ([8]). *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.

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. $|f(w)| = 2^{|w|}$.

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 $i \in \{0, \ldots, k\}$:*
- *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.

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

▶ Remark 3.4. The converse to this corollary does not hold: Example 2.4 provides an example of single-state 1-layered SST whose output size is quadratic in its input size, whereas regular functions have at most linear growth. This shows that there are meaningful differences between single-state SST and HDT0L systems when one looks at the layering condition due to the access to the output alphabet letters (see Remark 2.6).

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 3.5.** *The sequential (and therefore regular) function defined by the transducer of Figure 1 (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.

Polyregular functions vs layered SSTs By applying some results from [3], we can state a variant of Definition 2.10 which is a bit more convenient for us.

▶ **Proposition 3.6.** *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 layered SSTs – equivalently, HDT0L transductions with at most polynomial growth – yields exactly 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 3.7.** *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 5.3. Therefore, as promised in the introduction:

▶ **Corollary 3.8** (claimed in [8, Section 6])**.** *Layered SSTs are* not *closed under composition.*

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 substitutions *of f with the family* $(g_i)_{i\in I}$ *is the function*

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

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 squaring (without underlining) function $w \mapsto w^{|w|}$, which can be expressed as $\mathrm{CbS}(f : w \mapsto a^{|w|}, (g_a : w \mapsto w))$ with f and g_a regular, is comparison-free polyregular. Its growth rate is quadratic, while regular functions have at most linear growth. Other examples that also require a single CbS are given in Theorem 5.3.

This definition has the disadvantage of not explaining the name, which comes from the equivalent characterization by comparison-free pebble transducers (Section 6). In particular, one consequence of this equivalence is that:

▶ **Proposition 4.4.** *Every comparison-free polyregular function is, indeed, polyregular.*

But this inclusion can also be obtained as an immediate corollary of the following result, that we prove using *polynomial list functions* [3, Section 4], a formalism for polyregular functions based on the simply typed λ-calculus.

▶ **Theorem 4.5.** *Polyregular functions are closed under composition by substitutions.*

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

▶ **Proposition 4.6.** *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 $(g_i : \Gamma^* \to \Sigma^*)_{i\in I} \in \mathcal{C}^I$,*

is exactly the class of comparison-free polyregular functions.

For instance, by induction over the above characterization, we can assign to every comparison-free polyregular function an integer *rank*, whose operational meaning will be clarified by the connection with pebble transducers.

▶ **Definition 4.7.** *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* $\mathrm{CbS}(f, (g_i))$ *where f is regular and each g_i is of rank at most k.*

▶ **Proposition 4.8.** *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,*

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

5 Closure under composition and separation properties

Before we present alternative computational models for comparison-free polyregular functions, let us work with our current definition to establish some of their properties.

▶ **Theorem 5.1.** *Comparison-free polyregular functions are closed under composition.*

To prove this, we exploit the following combinatorial phenomenon: in a copyless SST, a transition acting on the state and registers can be specified by
- a "shape" described by an element of its *substitution transition monoid* [7, §3], this monoid being *finite* due to the copyless assignment restriction,

- 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.

This technique is often applied to the study of copyless SSTs (see also [4, p. 206–207]). Another property that we use is the closure of regular functions under regular conditionals:

▶ **Lemma 5.2** ([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.*

Next, we show that comparison-free polyregular functions are incomparable with HDT0L transductions, and strictly included in the class of polyregular functions.

▶ **Theorem 5.3.** *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 squaring function $w \in \Sigma^* \mapsto w^{|w|}$ for $|\Sigma| \geq 2$ (cf. Example 4.3);*
- **(iii)** (from [8, §6]) *any function that maps $a^n \| w \in \Sigma^*$ to $(w\|)^n$ for $a, \| \in \Sigma$ with $a \neq \|$.*

▶ **Theorem 5.4.** *There exists a HDT0L transduction which is also a polyregular function, but is not comparison-free: $f : a^n \in \{a\}^* \mapsto ba^{n-1}b \dots baabab$ (with $f(\varepsilon) = \varepsilon$ and $f(a) = b$).*

Theorem 5.3 follows from a pumping argument. Concerning the first part of Theorem 5.4, observe that f is precisely the function computed by the (single-state) 1-layered SST of Example 2.4 for $\Gamma = \{a\}$, taking $b = \underline{a}$. The proof that f is not comparison-free requires more work and is the subject of the remainder of this section. First, we need a pumping lemma for regular functions with unary input alphabet:

▶ **Lemma 5.5.** *For any regular function $f : \{a\}^* \to \Sigma^*$, there exist $m, p_0, k \in \mathbb{N}$ with $m \neq 0$ such that, for every $p \geq p_0$, one can find $u_0, \dots, u_k, v_1, \dots, v_k \in \Sigma^*$ with the property that*

$$\forall n \in \mathbb{N}, \quad f(a^{mn+p}) = u_0 (v_1)^n u_1 (v_2)^n \dots (v_k)^n u_k$$

▶ Remark 5.6. Let $L \subseteq \Sigma^*$ be the range of some two-way nondeterministic finite transducer – this includes, by Theorem 6.3, the case $L = f(\Gamma^*)$ where $f : \Gamma^* \to \Sigma^*$ is regular. A result by Smith [24] states that L is *k-iterative* for some $k \in \mathbb{N}$: every $w \in L$ that is long enough can be decomposed into $w = u_0 v_1 u_1 v_2 \dots v_k u_k$ with $v_1 \dots v_k \neq \varepsilon$ such that for any $n \in \mathbb{N}$, the pumped word $u_0 (v_1)^n u_1 (v_2)^n \dots (v_k)^n u_k$ is in L. The special case for a regular function $\Gamma^* \to \Sigma^*$ with $|\Gamma| = 1$ can be recovered as a consequence of Lemma 5.5. (But our proof is rather different from [24] since we work with SSTs instead of two-way transducers.)

With our pumping lemma, we can prove a combinatorial result on the maximal number of consecutive occurrences of a given letter in a word.

▶ **Definition 5.7.** *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 function $f : \{a\}^* \to \Sigma^*$ is poly-uniform *if for every $c \in \Sigma$ there exists a* finite *set of polynomials $A_{f,c} \subseteq \mathbb{Q}[X]$ such that, for every $n \in \mathbb{N}$, $\beta_c(f(a^n)) \subseteq \{P(n) \mid P \in A_{f,c}\}$.*

▶ **Lemma 5.8.** *Every comparison-free polyregular function $f : \{a\}^* \to \Sigma^*$ is poly-uniform. Furthermore, the polynomials in $A_{f,c}$ can be chosen to have degrees at most $\mathrm{rk}(f) + 1$.*

Proof of Theorem 5.4. The function f given in the theorem statement has the property that $\beta_a(f(a^n)) = \{0, \dots, n-1\}$ for $n \in \mathbb{N} \setminus \{0\}$. Since this sequence of sets has unbounded cardinality, f is not poly-uniform. Therefore, it is not comparison-free polyregular. ◀

6 Comparison-free pebble transducers

As promised, we now justify the name of our function class by an alternative characterization.

▶ **Definition 6.1.** *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 strings $\triangleright w \triangleleft$ (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*

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

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

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[5] $((\triangleright w \triangleleft)[i_1], \dots, (\triangleright w \triangleleft)[i_p])$ 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[6] the definition of *pebble transducers* from [3, §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[7] $i_j \leq i_{j'}$ (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").

▶ Remark 6.2. 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.

---

[5] Recall that we write $u[i]$ for the i-th letter of the word u.

[6] There is also an inessential difference: the definition given in [3] 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 [9, 15].

[7] 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 [19] (this is also essentially what happens in the nested transducers of [15]).

That said, the set of inputs for which a given pebble tree transducer does not terminate is always a regular language [19, Theorem 4.7]. This applies *a fortiori* to CFPTs. Using this, it is possible 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 6.3** ([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 [9] and made this class deserving of the name "regular functions" before the introduction of copyless SSTs.

Let us now show the equivalence with the definition based on composition by substitutions. The reason for this is similar to the reason why k-pebble transducers are equivalent to the *k-nested transducers* of [15], which is deemed "trivial" and left to the reader in [15, Remark 6]. But in our case, one direction (Theorem 6.5) involves an additional subtlety compared to in [15]; to take care of it, we use the fact that the languages recognized by pebble automata are regular (this is [19, Theorem 4.7] again) together with regular conditionals (Lemma 5.2).

▶ **Proposition 6.4.** *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 6.5.** *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 $f = \mathrm{CbS}(h, (g_i)_{i\in I})$.*

▶ **Corollary 6.6.** *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.*

7 Definability in the $\lambda\ell^{\oplus\&}$-calculus

We now conclude by discussing a final description of comparison-free polyregular functions as functions definable in a simply-typed λ-calculus, building on our previous work characterizing regular functions [21]. We give a high-level overview of the formalism; we refer the reader to [21, Section 2.4] for details.

The calculus, which they call $\lambda\ell^{\oplus\&}$, corresponds to proofs in intuitionistic additive multiplicative linear logic. Its grammar of types is accordingly inductively defined to contain an anonymous base type $\mathbb{o}$, tagged unions $\tau \oplus \sigma$, cartesian products $\tau \,\&\, \sigma$ and two distinct kinds of function type constructors $\tau \to \sigma$ and $\tau \multimap \sigma$. Each type constructor comes with the expected term constructors (e.g., λ-abstraction $\lambda x.t$ for functions types $\tau \multimap \sigma$ and pairings $\langle t, u\rangle$ for product types) and eliminators (e.g., function application $t\ u$ and projections $\pi_i(t)$ for cartesian products). We only consider well-typed terms, and write $t : \tau$ to mean "t is a term of type τ". Given types τ and σ, we write $\tau[\sigma]$ for the type τ where every occurrence of the base type $\mathbb{o}$ is replaced with σ. It is easy to check that if $t : \tau$, then $t : \tau[\sigma]$.

The two distinct arrow types serve to distinguish between unrestricted functions $\tau \to \sigma$ and *linear* functions $\tau \multimap \sigma$ which are only allowed to use their argument exactly once. This

linearity discipline is crucial to capture a counterpart of the copylessness restriction for SST. We say that a type τ is *purely linear* if it is built without using the non-linear arrow $\to$.

While *there is no primitive constructs for strings*, strings can be represented as anonymous functions through so-called *Church encodings*[8]: typically, the string *abaa* will be represented by $a \mapsto b \mapsto e \mapsto a(b(a(a(e))))$. The encoding of a string over alphabet Γ can be typed as the function type $\mathtt{Str}_\Gamma = (\mathbb{o} \multimap \mathbb{o}) \to \ldots \to \mathbb{o} \to \mathbb{o}$ with $|\Gamma|$ arguments of type $\mathbb{o} \multimap \mathbb{o}$. With this definition, regular functions can be thus characterized:

▶ **Theorem 7.1** ([21, Theorem 1.1])**.** *A function* $\Gamma^* \to \Sigma^*$ *is* regular *if and only if it is definable by a* $\lambda\ell^{\oplus\&}$*-term*[9] *of type* $\mathtt{Str}_\Gamma[\tau] \multimap \mathtt{Str}_\Sigma$ *for some* purely linear *type* τ*.*

The result of this section is that changing a single character in the above type gives us a characterization of comparison-free polyregular functions instead. A possible intuition is that while the pure linearity of τ corresponds to copylessness in SSTs, the use of a linear function arrow corresponds to the fact that an SST performs a *single traversal* of its input. We keep the former constraint, but relax the latter.

▶ **Theorem 7.2.** *A function* $\Gamma^* \to \Sigma^*$ *is* comparison-free polyregular *if and only if it is definable by a* $\lambda\ell^{\oplus\&}$*-term of type* $\mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$ *for some* purely linear *type* τ*.*

The left-to-right direction consists in a straightforward coding exercise while the converse is much more involved; let us give a few ideas used in the proof.

Without loss of generality, we may consider functions defined by *normalized* terms of the shape $\lambda s.t : \mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$. The main argument then goes by induction on the number of occurrences of s in t, using a syntactic characterization of normal forms (Lemma F.5). This however does not answer the question of what is the exact statement of the inductive hypothesis, as subterms containing s may have types wildly different from $\mathtt{Str}_\Sigma$. To address this, we generalize the notion of comparison-free functions to allow for codomains of the shape $\sum_{q\in Q}[R_q \to \Sigma^*]$. We then call a function $\Gamma^* \to \sum_{q\in Q}[R_q \to \Sigma^*]$ comparison-free when the induced function $\Gamma^* \to Q$ is regular and the induced partial functions $\Gamma^* \rightharpoonup \Sigma^*$, one for every $q \in Q$ and $r \in R_q$, can be extended to a comparison-free polyregular function.

With this generalized definition and a semantic interpretation $\tau \mapsto \sum_{Q_\tau}[R_\tau \to \Sigma^*]$ of purely linear types from [21], we can thus prove an invariant entailing Theorem 7.2. The CbS combinator is used once at each inductive step, so we may note that the rank of the obtained comparison-free polyregular is bounded by the number of occurrences of the main argument s in the corresponding normal $\lambda\ell^{\oplus\&}$-term.

8 Perspectives

We demonstrated that comparison-free polyregular functions constitute a robust class of transductions by giving three equivalent characterizations. Further, we compared this new class of string functions with polyregular and HDT0L transductions, proving strict inclusions where possible. We give a synthetic summary of these results in Figure 2.

This naturally leads to many further questions regarding alternative characterizations and generalizations of comparison-free polyregular functions; we list only a few.

[8] The name comes from Church's historical representation of natural numbers (see e.g. [14]) but the generalization to other algebraic data types came later [6].

[9] To be pedantic, we should say a *closed* $\lambda\ell^{\oplus\&}$-term, i.e. without free variables.

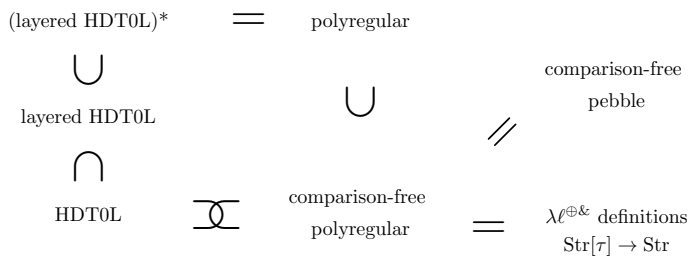

Figure 2 Summary of our results. Inclusions $\subset$ where shown to be strict, and $\between$ means that there is no inclusion either way. Finally C^* denotes the closure under composition of the class C.

Interpretations Regular and polyregular string functions also enjoy logical characterizations: regular functions correspond to MSO transductions [9] while polyregular functions correspond to MSO interpretations [5]. The basic conceit behind these definitions is that a string w may be regarded as a finite model $\mathfrak{M}(w)$ for MSO over a signature containing the order relation $\leq$ on positions and predicates encoding their labeling.

We conjecture that comparison-free polyregular functions enjoy a similar characterization using the direct product of structures. Given a finite model $\mathfrak{U} = (U, R, \ldots)$, we write $\mathfrak{U}^k$ for the k^{th} power $(U^k, R_1, \ldots, R_k, \ldots)$ where $R_i(x_1, \ldots, x_m)$ of arity m is defined as $R(\pi_i(x_1), \ldots, \pi_1(x_m))$ for $1 \leq i \leq k$.

▷ Conjecture 8.1. A function $f : \Sigma^* \to \Gamma^*$ is comparison-free polyregular if and only if there exist a regular function $g : \Sigma^* \to \Delta^*$, $k \in \mathbb{N}$ and a one-dimensional first-order (not MSO!) interpretation φ such that $\mathfrak{M}(f(w)) = \varphi\left(\mathfrak{M}(g(w))^k\right)$ for every $w \in \mathbb{N}$.

Our proposed class of interpretations is easily seen to be contained in MSO interpretations and rich enough to encode CbS. On an intuitive level, it seems to capture the inability of comparing the positions of two heads of comparison-free pebble transducers.

Polynomial tree transductions Interestingly, our notion of comparison-free polyregular functions gracefully extends to the setting of ranked trees in two equivalent ways. For this discussion, let us write $\mathbf{\Sigma} = (\Sigma, \mathrm{ar})$ for ranked alphabet, where Σ is an alphabet and $\mathrm{ar} : \Sigma \to \mathbb{N}$, and $\mathcal{T}^{(k)}_{\mathbf{\Sigma}}$ for the set of trees with k holes with nodes in $\mathbf{\Sigma}$. We write $\mathcal{T}_{\mathbf{\Sigma}}$ for the set of trees over $\mathbf{\Sigma}$ $\left(= \mathcal{T}^{(0)}_{\mathbf{\Sigma}}\right)$.

First, the notion of $\lambda\ell^{\oplus\&}$-definability readily extends to trees; there is a type of Church encoding of trees $\mathtt{Tree}_{\mathbf{\Sigma}}$. The $\lambda\ell^{\oplus\&}$-terms of type $\mathtt{Tree}_{\mathbf{\Sigma}}[\tau] \multimap \mathtt{Tree}_{\mathbf{\Gamma}}$ (with τ purely linear) correspond to regular tree functions [21, Theorem 1.2]. By analogy with the case of strings, one may consider the corresponding class of functions implemented by terms $\mathtt{Tree}_{\mathbf{\Sigma}}[\tau] \to \mathtt{Tree}_{\mathbf{\Gamma}}$.

The notion of composition by substitution also readily extends to trees (with holes): given a function $f : \mathcal{T}_{\mathbf{\Sigma}} \to \mathcal{T}_{\mathbf{I}}$ and a family $g_i : \mathcal{T}_{\mathbf{\Sigma}} \to \mathcal{T}^{(\mathrm{ar}(i))}_{\mathbf{\Gamma}}$, one may form a function $\mathrm{CbS}(f, (g_i)_{i \in I}) : \mathcal{T}_{\mathbf{\Sigma}} \to \mathcal{T}_{\mathbf{\Gamma}}$ using the same procedure as for strings: on a given input t, substitute every node in $f(t)$ by $g_i(t)$. The closure of regular functions over trees (with holes) by this extended CbS operator is another natural class of tree functions to consider.

The proof of Theorem 7.2 can be adapted to show that these two classes coincide. However, we do not have equivalent logical or transducer-based characterizations.

Relationship to polyregular functions We presented comparison-free polyregular functions as a strict subclass of polyregular functions. This leads to a natural *membership problem*:

▷ Problem 8.2. 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 general and comparison-free polyregular functions. We believe that investigating such issues may also lead to machine/syntax-free characterizations of the containment between the two classes. One might for instance hope to generalize Definition 5.7 to obtain such a characterization.

Polyregular functions and λ-calculus A natural question is whether it is possible to give a characterization of general polyregular functions in terms of λ-terms and Church encodings. Theorem 7.2 implies that terms $t : \mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$ with τ purely linear are not enough while simply lifting the linearity restriction on τ can lead to outputs of hyperexponential size [23].

We conjecture that polyregular functions can be captured by terms $t : \mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$ for general τ in (a variant of) the *parsimonious* λ-calculus [17, 18]. The difference with $\lambda\ell^{\oplus\&}$ is that terms of type $\tau \to \sigma$ are subject to a stricter tiering discipline, which seems closely related to that of layered SSTs.

References

1 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, pages 1–12, 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 Mikołaj Bojańczyk. Polyregular Functions. *CoRR*, abs/1810.08760, October 2018. arXiv:1810.08760.

4 Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox. Lecture notes for a course at the University of Warsaw, 2018. URL: https://www.mimuw.edu.pl/~bojan/paper/automata-toolbox-book.

5 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.

6 Corrado Böhm and Alessandro Berarducci. Automatic synthesis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39:135–154, January 1985. doi:10.1016/0304-3975(85)90135-5.

7 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.

8 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.

9 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.

10 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.

11 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.

12 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.

13 Emmanuel Filiot and Pierre-Alain Reynier. Copyful Streaming String Transducers. To appear in Fundamenta Informaticae (long version of a paper in Proc. of 11th International Workshop on Reachability Problems (RP 2017)), 2017. URL: http://pageperso.lif.univ-mrs.fr/~pierre-alain.reynier/files/copyful_submitted.pdf.

14 Oleg Kiselyov. Many more predecessors: A representation workout. *Journal of Functional Programming*, 30, 2020. doi:10.1017/S095679682000009X.

15 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.

16 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.

17 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.

18 Damiano Mazza. *Polyadic Approximations in Logic and Computation*. Habilitation à diriger des recherches, Université Paris 13, November 2017. URL: https://lipn.fr/~mazza/papers/Habilitation.pdf.

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

20 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.

21 Lê Thành Dũng Nguyễn, Camille Noûs, and Pierre Pradic. Implicit automata in typed λ-calculi II: streaming transducers vs categorical semantics. *CoRR*, abs/2008.01050v1, 2020. arXiv:2008.01050v1.

22 Gabriel Scherer. *Which types have a unique inhabitant? : Focusing on pure program equivalence.* PhD thesis, Paris Diderot University, France, 2016. URL: https://tel.archives-ouvertes.fr/tel-01309712.

23 Helmut Schwichtenberg. Complexity of normalization in the pure typed lambda–calculus. In *Studies in Logic and the Foundations of Mathematics*, volume 110, pages 453–457. Elsevier, 1982. doi:10.1016/S0049-237X(09)70143-0.

24 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.

25 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.

A Technical preliminaries on SSTs

We recall here some classical objects and constructions related to streaming string transducers, that will be helpful for the various proofs involving SSTs. Most of the propositions are straightforward consequences of the definitions.

A.1 Transition monoids

Register assignments (Definition 2.2) can be given a monoid structure as follows.

▶ **Definition A.1.** *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:*

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

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 A.2.** *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 A.3.** *$\mathcal{M}^{\mathrm{cl}}_{R,\Sigma}$ 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 $(-)^\dagger$ interprets register assignments as functions on tuples of strings, i.e. it sends $\mathcal{M}_{R,\Sigma}$ to $(\Sigma^*)^R \to (\Sigma^*)^R$.

▶ **Proposition A.4.** *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 A.5.** *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 A.6.** *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 A.7.** *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 A.8. 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 [7, Section 3].

▶ **Proposition A.9.** *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 $|\alpha(r)| \leq |R|$. ◀

A.2 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 the previous subsection. We leave the details to the reader.

B Proofs for Section 3

B.1 Proof of Theorem 3.2

Proof of ($\Rightarrow$). The translation from SSTs to HDT0L systems given by [13] turns out to work. 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

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

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[10]. 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 [13]. 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. ◀

Proof of ($\Leftarrow$)**.** The translation from HDT0L systems to single-state SSTs mentioned in Remark 2.6 (see Appendix A.2) 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 A.2).

We define a k-layered SST with:

[10] 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.

- $\mathcal{M}^{\mathrm{cl}}_{\Delta_0,\varnothing}$ as the set of states (finite by Proposition A.9), 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 ($0 \leq i \leq k$);
- 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

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.2. As can readily be seen from the definitions, this construction preserves the 1-layered property.

B.3 Proof of Proposition 3.5

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 1. 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, $|\omega_n|_r \leq |\omega|_r$ 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 1, 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 3.6

It is stated in the introduction to [3] that all regular functions are polyregular. One way to see this is discussed in Section 6: the characterization by pebble transducers given in [3] 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 2.9 is copyless, the iterated reverse function is regular.

B.5 Proof of Theorem 3.7

Proof of (i) ⇒ (iii). Thanks to Proposition 3.6, 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

$$1234 \mapsto \underline{4}321\underline{3}21\underline{2}11 \mapsto \underline{1}2341\underline{2}3412\underline{3}41234\underline{4}$$

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 2.10, polyregular functions are closed under composition). ◀

C Proofs for Section 4

C.1 Proof of Theorem 4.5

We start by briefly recalling the definition of *polynomial list functions* from [3, 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 $\texttt{is}_a^S$ and we have

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

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 $\texttt{is}_a^S$ receives the type

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

See [3, 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 4.6

This is equivalent to claiming that the class of comparison-free polyregular function is the least class containing regular functions and such that for every regular function $f : \Gamma^* \to I^*$, if the class contains $g_i : \Gamma^* \to \Sigma^*$, we have $\mathrm{CbS}(f, (g_i)_i)$ in the class. This can be shown by induction using the equation

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

D Proofs for Section 5

D.1 Proof of Theorem 5.1

As a corollary of Theorem 7.1 Before giving a direct proof, let us note that Theorem 5.1 also follows as a straightforward corollary of Theorem 7.1.

Proof of Theorem 5.1 from Theorem 7.1. Suppose that we have comparison-free polyregular functions $f : \Gamma^* \to \Sigma^*$ and $h : \Sigma^* \to \Delta^*$. By the easy direction of Theorem 7.1, we know that we have purely linear types τ, κ and $\lambda\ell^{\oplus\&}$-terms $t : \mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$ and $u : \mathtt{Str}_\Sigma[\kappa] \to \mathtt{Str}_\Delta$ implementing f and h respectively. Then, $\lambda^! w.\ t\ (u\ w) : \mathtt{Str}_\Gamma[\tau[\kappa]]$ implements $h \circ f$, which is thus comparison-free polyregular by Theorem 7.1. ◀

We point this out because we provide a proof of Theorem 7.1 which does not rely on Theorem 5.1; together with the argument above, this yields an alternative proof of Theorem 5.1. We give a more direct argument below, not going through λ-calculus, for clarity's sake; another advantage of the proof below is that it is self-contained.

Direct proof of Theorem 5.1 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 D.1.** *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 relies on some properties of the transition monoids introduced in Appendix A.1.

▶ **Proposition D.2.** *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 $r'_1 \dots r'_n = \mathtt{erase}_\Delta(\beta)(r) \in R^*$ (cf. Proposition A.7). Hence*

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

In other words, register assignments in $\mathcal{M}^{\mathrm{cl}}_{R,\Delta}$ can be decomposed into a "shape" in $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ plus finitely many string labels. Through this bijection, $\mathtt{erase}_\Delta \in \mathrm{Hom}(\mathcal{M}^{\mathrm{cl}}_{R,\Delta} \wr Q,\ \mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)$ can be seen as simply removing the labels, i.e. the $\vec{\ell}$ component.

▶ **Lemma D.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*

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

and $\varphi_\delta = \mathtt{erase}_\Delta \circ \psi_\delta$ as in Remark A.8, $q \in Q$, $r \in R$, $\alpha \in \mathcal{M}^{\mathrm{cl}}_{R,\varnothing}$ and $j \in \{0, \dots, |\alpha(r)|\}$. Then the following function $\Sigma^* \to \Delta^*$, defined thanks to Proposition D.2, is regular:

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

(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 A.9). 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 [21], 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

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

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 D.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 D.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 D.1, which suffices to finish the proof of Theorem 5.1.

Proof of Lemma D.1. 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

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

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, we claim that 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*

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

Let us deduce the conclusion from this claim. 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 substitutions, 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 5.2). This argument relies on the finiteness of the indexing set $(\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q)^I$ – a consequence of Proposition A.9 – 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 [3];
  - 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.

Proof of the existence claim. Proposition D.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

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

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

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

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 D.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 $i_1 \dots i_n$. (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 D.1.) This directly implies the property relating h, $h''_{\vec{\mu}}$ and $(l_\lambda)_{\lambda \in \Lambda_{\vec{\mu}}}$ that we wanted. ◀

D.2 Proof of Theorem 5.3

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 substitutions 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 $\mathtt{letters}(w)$ 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 $\mathtt{letters}(h^n(d))$ 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^kb 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 $a^n \mapsto (a^n b)^{n+1}$.

Note that the assumption $|\Sigma| \geq 2$ is necessary: in the unary case, $a^n \mapsto (a^n)^{|a^n|} = a^{n^2}$ is comparison-free polyregular.

(iii) is not HDT0L. (This is claimed without proof in [8, 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 the 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 $(a^{\min(X)} \|)^m$.

D.3 Proof of Lemma 5.5

We consider 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 Appendix A.1, which contains $\mu = \texttt{erase}_\Sigma(\delta(-, a))$. Since $\mathcal{M}^{\mathrm{cl}}_{R,\varnothing} \wr Q$ is finite (Proposition A.9), 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 $p \geq m$. Let $(q, \alpha) = \mu^{\bullet p}(q_0)$ where q_0 is the initial state of the SST. Since $p - m \geq 0$, we have $\mu^{\bullet p} \bullet \mu^{\bullet m} = \mu^{\bullet (p+m)} = \mu^{\bullet (p-m)} \bullet \mu^{\bullet 2m} = \mu^{\bullet (p-m)} \bullet \mu^{\bullet m} = \mu^{\bullet p}$ 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^{mn+p} 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 $\texttt{erase}_\Sigma(\gamma) = \beta$.

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+mn+p} (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 $\vec{w}_p$ be the register values after it has read a prefix a^p. Then after reading a^{mn+m+p}, the new register values are $(\gamma^{\bullet(n+1)})^\dagger(\vec{w}_p)$. More precisely, the register r contains:

- $\gamma(r) \in \Sigma^*$ if $\beta(r) = \varepsilon$;
- $(\gamma^{\odot}(u_r))^n \cdot ((u_r r v_r)^\dagger(\vec{w}_p)) \cdot (\gamma^{\odot}(v_r))^n$ 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 $f(a^{mn+m+p})$. 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 $r \in R$. Note that the number of factors is $O(|R|)$, so the k from the lemma statement can indeed be chosen independently from p. (Furthermore, what plays the role of the p in that statement is $m + p$ here; since we took $p \geq m$, we should set $p_0 = 2m$.)

D.4 Proof of Lemma 5.8

Before giving the proof, let us have a couple of auxiliary definitions and lemmas. For any pair of integers $m > 0$ and $p \in \mathbb{N}$, call $\phi_{p,m} : \{a\}^* \to \{a\}^*$ the function $a^n \mapsto a^{mn+p}$.

▷ Claim D.4. Let $m > 0$ and p be natural numbers. A function $f : \{a\}^* \to \Sigma^*$ is poly-uniform if and only if $f \circ \phi_{l,m}$ is poly-uniform for every $p \leq l < m$.

Proof. For the direct implication, assuming that we have a finite set $A_{f,c} \subseteq \mathbb{Q}[X]$, we may take $A_{f \circ \phi_{l,m},c} = \{P(mX + l) \mid P \in A_{f,c}\}$. For the converse, we may take

$$A_{f,c} \quad = \quad \bigcup_{p \leq l < m} \left\{ P\left(\frac{X - l}{m}\right) \mid P \in A_{f \circ \phi_{l,m},c} \right\} \cup \bigcup_{n < p} \beta_c(f(a^n))$$

which is finite whenever the $A_{f \circ \phi_{l,m},c}$ are. ◀

▷ Claim D.5. Given two poly-uniform functions f and g, the pointwise concatenation $a^n \mapsto f(a^n)g(a^n)$ is poly-uniform.

Proof. Calling h the pointwise concatenation of f and g, we may take

$$A_{h,c} \quad = \quad \{P + Q \mid P \in A_{f,c}, Q \in A_{g,c}\}$$

◀

▷ Claim D.6. Fix a word w. For any family of poly-uniform functions g_i and function, $\mathrm{CbS}((a^n \mapsto w^n), (g_i)_{i\in I})$ is poly-uniform.

Proof. Suppose that we have finite sets $A_{g_i,c} \subseteq \mathbb{Q}[X]$ for $i \in I$ and $c \in \Sigma$ witnessing that the g_is are poly-uniform. For any word $u \in I^*$, define the finite sets $A_{u,c} \subseteq \mathbb{Q}[X]$ by induction on the length of u by $A_{\varepsilon,c} = \{0\}$ and $A_{ui} = \{P + Q \mid P \in A_u, Q \in A_{g_i,c}\} \cup A_u$. It can be then be checked that $A_{h,c} = A_{ww} \cup \{X \cdot P(X) \mid P \in A_w\}$ witnesses that $\mathrm{CbS}((a^n \mapsto w^n), (g_i)_{i\in I})$ is poly-uniform. ◀

We are now ready to prove Lemma 5.8 by induction over the rank of the comparison-free polyregular function under consideration. First, let us prove it for regular functions (rank 0).

▶ **Lemma D.7.** *If $f : \{a\}^* \to \Sigma^*$ is regular, then it is poly-uniform.*

Proof. We apply Lemma 5.5 to get $p \in \mathbb{N}$ and a period $m \in \mathbb{N}$, as well as families of words $u_{l,0}, \ldots, u_{l,k}$ and $v_{l,1}, \ldots, u_{l,k}$ for $l < m$ (we can take $u_{l,i}$ or $v_{l,i}$ empty to get k independent from l) such that

$$f(a^{mn+p+l}) = (f \circ \phi_{p+l,m})(a^n) = u_{l,0}(v_{l,1})^n \ldots (v_{l,k})^n u_{l,k}$$

By Claim D.4, it suffices to show that for every $(u_i)_{i=0}^K$ and $(v_i)_{i=1}^k$, the map

$$a^n \mapsto u_0 v_1^n \ldots v_k^n u_k$$

is poly-uniform. This can be done by induction over k, applying Claim D.5 and noticing that the maps $a^n \mapsto w^n$ are poly-uniform (this is a corollary of Claim D.6). ◀

Now we can proceed similarly for the inductive step.

Proof of Lemma 5.8. Suppose that we have a comparison-free polyregular function f. We have just treated the case where it is of rank 0, so suppose it is of rank $k > 0$ and we have $f = \mathrm{CbS}(h, (g_i)_i)$ with h regular and the g_is poly-uniform. Apply Lemma 5.5 to obtain $p \in \mathbb{N}$ and a period $m \in \mathbb{N}$, as well as families of words $u_{l,0}, \ldots, u_{l,k}$ and $v_{l,1}, \ldots, u_{l,k}$ for $l < m$ such that

$$h(a^{mn+p+l}) = (h \circ \phi_{p+l,m})(a^n) = u_{l,0}(v_{l,1})^n \ldots (v_{l,k})^n u_{l,k}$$

By Claim D.4, it suffices to show that $f \circ \phi_{l,m} = \mathrm{CbS}(h \circ \phi_{l,m}, (g_i \circ \phi_{l,m})_i)$ is poly-uniform for every $p \leq l < p + m$. By a simple induction over k and using Claim D.5, it suffices to show that $\mathrm{CbS}((n \mapsto (v_{l,j})^n), (g_i \circ \phi_i)_i)$ and $\mathrm{CbS}((n \mapsto u_{l,j}), (g_i \circ \phi_i)_i)$ are poly-uniform. The latter case is also treated using repeatedly Claim D.5, while the former corresponds exactly to Claim D.6. ◀

E Proofs for Section 6

E.1 Proof of Proposition 6.4

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

E.2 Proof of Theorem 6.5

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 [19, 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 5.2, 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 $\mathcal{T}_q$.

E.3 Proof of Corollary 6.6

The proof goes by induction over $k \in \mathbb{N}$. By Theorem 6.3, 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 6.5 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 6.3, h is computable by a 1-CFPT, so by Proposition 6.4, f is computed by a $(k+1)$-CFPT

F Proof of Theorem 7.1

F.1 Extensional completeness

The easy direction, i.e., the fact that every comparison-free polyregular function is implemented by a $\lambda\ell^{\oplus\&}$-term is proven by induction on the rank. For rank 0, this is a consequence of [21, Theorem 1.1]: every regular function $\Sigma^* \to \Pi^*$ corresponds to a term of type $t : \mathtt{Str}_\Sigma[\tau] \multimap \mathtt{Str}_\Pi$ for some purely linear τ, so it is also implementable by the term $\lambda^! x.t\ x : \mathtt{Str}_\Sigma[\tau] \to \mathtt{Str}_\Pi$. We may further assume that τ is *inhabited*, i.e., that there is a closed term $t : \tau$; we shall maintain that invariant inductively.

The inductive step of the argument is then captured in the following lemma:

▶ **Lemma F.1.** *If f is computed by $t : \mathtt{Str}_\Sigma[\tau] \to \mathtt{Str}_I$ and g_i is computed by $u_i : \mathtt{Str}_\Sigma[\sigma_i] \to \mathtt{Str}_\Pi$ then $\mathrm{CbS}(f, (g_i)_{i \in I})$ is computed by some term of type $\mathtt{Str}_\Sigma[\kappa] \to \mathtt{Str}_\Pi$ (where τ, σ_i, κ all purely linear and inhabited).*

We first prove Lemma F.1 in the particular case where $\tau = \sigma_i$ for every $i \in I$.

▶ **Proposition F.2.** *If f is computed by $t : \mathtt{Str}_\Sigma[\tau] \to \mathtt{Str}_I$ and g_i is computed by $u_i : \mathtt{Str}_\Sigma[\tau] \to \mathtt{Str}_\Pi$ then $\mathrm{CbS}(f, (g_i)_{i \in I})$ is computed by some term of type $\mathtt{Str}_\Sigma[\tau[\mathbb{o} \multimap \mathbb{o}]] \to \mathtt{Str}_\Pi$ (where τ is purely linear and inhabited).*

Proof. Write κ for $\tau[\mathbb{o} \multimap \mathbb{o}]$; by the substitution lemma, we have $u_i : \mathtt{Str}_\Sigma[\kappa] \to \mathtt{Str}_\Pi[\mathbb{o} \multimap \mathbb{o}]$ and $t : \mathtt{Str}_\Sigma[\kappa] \to \mathtt{Str}_I[\mathbb{o} \multimap \mathbb{o}]$ that allows us to define the term

$$\lambda^! s.\lambda^! h_1 \dots \lambda^! h_{|\Pi|}.\lambda^! \varepsilon.\ t\ (\hat{u}_1\ s\ h_1\ \dots\ h_{|\Pi|}) \dots\ (\hat{u}_{|I|}\ s\ h_1\ \dots\ h_{|\Pi|})\ (\lambda y.\ y)\ \varepsilon : \mathtt{Str}[\kappa] \to \mathtt{Str}_\Pi$$

where the terms $\hat{u}_i : \mathtt{Str}[\kappa] \to (\mathbb{o} \multimap \mathbb{o}) \to \dots \to (\mathbb{o} \multimap \mathbb{o}) \to ((\mathbb{o} \multimap \mathbb{o}) \multimap (\mathbb{o} \multimap \mathbb{o}))$ (for $i \in I$) defined as $\hat{u}_i = \lambda^! s.\lambda^! h_1 \dots.\lambda^! h_{|\Pi|}.\lambda x.\ u_i\ s\ (\lambda f.\lambda y.\ h_1\ (f\ y))\ \dots\ (\lambda f.\lambda y.\ h_{|\Pi|}\ (f\ y))\ (\lambda y.\ y)$. The above term implements $\mathrm{CbS}(f, (g_i)_{i \in I})$. ◀

Now we need to account for the general case where A and the B_is might differ. We reduce it to Proposition F.2 using a few auxiliary claims. Given arbitrary terms $s : \tau \multimap \sigma$ and $r : \sigma \multimap \tau$, write $\mathtt{cast}_{s,r} : \mathtt{Str}_\Sigma[\sigma] \multimap \mathtt{Str}_\Sigma[\tau]$ for

$$\lambda w.\lambda^! a_1 \dots \lambda^! a_{|\Sigma|}.\lambda^! \varepsilon.\ r\ (w\ (\lambda x.s\ (a_1\ (r\ x)))\ \dots\ (\lambda x.s\ (a_{|\Sigma|}\ (r\ x)))\ (s\ \varepsilon))$$

We call a type τ a $\lambda\ell^{\oplus\&}$-definable retract of σ if there are terms $s : \tau \multimap \sigma$ and $r : \sigma \multimap \tau$ such that $\lambda a.\ r\ (s\ a)$ be $\beta\eta$ equivalent to the identity function; in such a case, we call (s, r) a $\lambda\ell^{\oplus\&}$-section-retraction pair from τ to σ.

▷ Claim F.3. If (s, r) is a $\lambda\ell^{\oplus\&}$-section-retraction pair and $\underline{w}$ is the Church encoding of some word $w \in \Sigma$ then $\mathtt{cast}_{s,r}\ \underline{w} \quad =_{\beta\eta} \underline{w}$.

Proof. Easy induction over w. ◀

This means in particular that if a function f has a $\lambda\ell^{\oplus\&}$-definition $t : \mathtt{Str}_\Sigma[A] \to \mathtt{Str}_\Pi$, and (s, r) witnesses that A is a retract of B, then $\lambda^!w.t\ (\mathtt{cast}_{s,r}\ w) : \mathtt{Str}_\Sigma[B] \to \mathtt{Str}_\Pi$ is also a valid definition of f.

To reduce Lemma F.1 to Proposition F.2, it is then sufficient to note that if τ and the σ_is are all purely linear and inhabited, then so is $\kappa = (\tau \,\&\, \mathbf{I}) \otimes (\sigma_1 \,\&\, \mathbf{I}) \,\&\, \ldots \otimes (\sigma_{|I|} \,\&\, \mathbf{I})$. Further, τ is a retract of κ. The retraction is given by discarding every component of the tensor but the first and then projecting. The section is obtained by pairing the input with $()$ and the default elements of the σ_i witnessing that they are inhabited. Similarly, every σ_i is a retract of κ, so we may conclude.

F.2 Soundness

We prove that all definable functions in the sense of Theorem 7.2 are comparison-free polyregular using a combination of a syntactic analysis of normal forms for $\lambda\ell^{\oplus\&}$-terms and semantic evaluation. We only give key results for the former in the body of the paper, with proofs deferred to the appendix. Using these results, we explain how to derive our characterization, leveraging our previous work [21] on semantic interpretation of purely linear $\lambda\ell^{\oplus\&}$-terms in terms of generalized SSTs.

Syntactic analysis The first step toward syntactic analysis is to normalize $\lambda\ell^{\oplus\&}$-terms. In fact, in order to make the subsequent proofs easier, it will be convenient to have a refined version of normal forms which does not only eliminate all β-redexes, but also orders the use of eliminators (term applications $t\ u$, case, $\ldots$) in a principled way according to a notion of *polarity*. In proof theory, this is called *focusing*. We give the precise definition of *partially focused* normal forms in Appendix F.3 and argue that the following holds:

▶ **Theorem F.4.** *Any typed term t is $\beta\eta$-equivalent to a term in* focused *normal form.*

Once this is done, we write $\Psi;\ \Delta \vdash_{\mathrm{NF_f}} t : A$ to mean that t is a focused normal form with type A in the context $\Psi;\ \Delta$. With this refined characterization, we then prove the following lemma that allows use to isolate occurrences of the first argument of normal terms of type $\mathtt{Str}_\Sigma[A] \to \mathtt{Str}_\Gamma$.

▶ **Lemma F.5.** *Let $\tau = \kappa_1 \to \ldots \to \kappa_k \to \kappa'$ be a type with κ' purely linear and s a distinguished variable of type τ. Let Δ and Ψ be purely linear contexts and t be a term such that $\Psi,\ s : \tau; \Delta \vdash_{\mathrm{NF_f}} t : \sigma$ for some purely linear σ. Suppose further that there is at least one occurrence of s in t. Then, there are terms o, d_1, $\ldots$, d_k such that $t =_{\beta\eta} o\ \langle s\ d_1\ \ldots\ d_k, ()\rangle$ and*

$$\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF_f}} o : (\kappa' \,\&\, \mathbf{I}) \multimap \sigma \qquad\qquad \Psi, s : \tau;\ \cdot \vdash_{\mathrm{NF_f}} d_i : \kappa_i \quad \textit{for } i \in \{1, \ldots, k\}$$

Furthermore, there are no more occurrences of s in $o\ \langle s\ d_1\ \ldots\ d_k, ()\rangle$ than in t.

The counting of the number of occurrences of the main argument $s : \tau$ in this lemma is important, as this number of occurrences is not invariant under $\beta\eta$. Further, this parameter will turn out to be a bound on the rank of the comparison-free polyregular function implemented by a $\lambda\ell^{\oplus\&}$-definition.

For the rest of the proof, we fix $\Psi_\Sigma = \{c : \mathbb{o} \multimap \mathbb{o} \mid c \in \Sigma\} \cup \{\varepsilon : \mathbb{o}\}$. Any $\lambda\ell^{\oplus\&}$-term of type $\mathtt{Str}_\Gamma[\tau] \to \mathtt{Str}_\Sigma$ can be normalized into a focused normal form (Theorem F.4) and η-expanded into

$$\lambda^!s.\, \lambda^!c_1.\, \ldots\, \lambda^!c_{|\Sigma|}.\, \lambda^!\varepsilon.\, t \qquad \text{where} \quad \Psi_\Sigma,\ s : \mathtt{Str}_\Gamma[\tau];\ \cdot \vdash_{\mathrm{NE_f}} t : \mathbb{o}$$

This fits the assumption of Lemma F.5 for $\kappa_1 = \cdots = \kappa_{|\Sigma|} = \mathbb{o} \multimap \mathbb{o}$ and $\kappa_{|\Sigma|+1} = \kappa' = \mathbb{o}$.

Semantic evaluation We rely on the semantic interpretation of the purely linear fragment of $\lambda\ell^{\oplus\&}$ described in [21]: for every alphabet Σ, we have a category $\mathcal{SR}(\Sigma)_{\oplus\&}$ in which purely linear types τ are interpreted by objects $[\![\tau]\!]$ and $\lambda\ell^{\oplus\&}$-terms $\Psi_\Sigma;\ \cdot \vdash t : \tau \multimap \sigma$ may be interpreted as morphisms in a way which is compatible with $\beta\eta$ conversion; this amounts to the fact that $\mathcal{SR}(\Sigma)_{\oplus\&}$ has a symmetric monoidal closed structure, as well as finite sums and products.

Formally speaking, the objects of $\mathcal{SR}(\Sigma)_{\oplus\&}$ are triples $(Q, (X_q)_{q\in Q}, (R_{q,x})_{q\in Q, x\in W_q})$ where Q, X_q and $R_{q,x}$ are finite sets. Let us take the convention that when we have an object A of $\mathcal{SR}(\Sigma)_{\oplus\&}$, we write $Q(A)$, $X(A)$ and $R(A)$ for the set and families of sets such that

$$A \;=\; \big(Q(A), (X(A)_q)_{q\in Q(A)}, (R(A)_{q,x})_{q\in Q(A), x\in X(A)_q}\big)$$

To simplify matters, assume that the $R_{q,x}$ are always disjoint from each other and Σ, and set R_q to be $\bigcup_{x\in X_q} R_{q,x}$. Morphisms from A to B can be regarded as pairs $(f, (\alpha_q)_{q\in Q(A)})$ where $f : Q(A) \to Q(B)$ is a set-theoretic map and $\alpha_q : R(B)_{f(q)} \to (\Sigma \cup R(A)_q)^*$ is a register assignment satisfying an additional *single-use restriction relative to* $X(A)_q$ *and* $X(B)_{f(q)}$: a register $x \in R(A)_{q,x}$ is only allowed to occur at most once in each component $\alpha_q(R(B)_{f(q),x'})$ for $x' \in X(B)_{f(q)}$. Write $A \to_\Sigma B$ for the set of such pairs. The composition $(g, (\beta_{q'})_{q'}) \circ (f, (\alpha_q)_q)$ is defined as $(c \circ f, (\gamma_q)_q)$ with γ_q such that $\gamma_q^\dagger = \beta^\dagger_{f(q)} \circ \alpha_q^\dagger$.

We leave checking that the above can be extended to a category, and that additionally, this category is isomorphic to $\mathcal{SR}(\Sigma)_{\oplus\&}$ as defined in [21], to the reader. The reason why this is not the exact same category is because we opted for a slightly different description of the homsets $A \to_\Sigma B$ which is more convenient for our purposes. We also refer to [21] for the details of the interpretation of purely linear types $\tau \mapsto [\![\tau]\!]$ and associated $\lambda\ell^{\oplus\&}$ terms in $\mathcal{SR}(\Sigma)_{\oplus\&}$. The object part of the symmetric monoidal structure, which serves to interpret $\otimes$ and $\mathbf{I}$ at the type level, is given by $\mathbf{I} = (\{\bullet\}, \{\bullet\}, \emptyset)$ and

$$A \otimes B = \big(Q(A) \times Q(B), (X(A)_q \times X(B)_{q'})_{(q,q')}, (R(A)_{q,x} \sqcup R(B)_{q',x'})_{(q,q'),(x,x')}\big)$$

These data, along with the obvious natural isomorphisms $(A \otimes B) \otimes C \cong A \otimes (B \otimes C)$, $\mathbf{I} \otimes A \cong A \otimes \mathbf{I} \cong A$ and $A \otimes B \cong B \otimes A$ fully describe the monoidal structure of $\mathcal{SR}(\Sigma)_{\oplus\&}$. The products $A \,\&\, B$, coproducts $A \oplus B$ and the closed structures $A \multimap B$ are completely determined up to unique isomorphism thanks to their universal properties; we refer to [21] for proofs that they do exist. The non-trivial result there is the existence of the closed structure $A \multimap B$, which relies on combinatorics similar to those involved in Lemma D.3 (they also appear in the proof of [21, Lemma 3.32]).

Regarding the interpretation of purely linear types, we fix $[\![\mathbb{o}]\!] = (\{\bullet\}, \{\bullet\}, \{\bullet\})$ and thus $\mathbf{I} \to_\Sigma [\![\mathbb{o}]\!] \cong \Sigma^*$ via a canonical isomorphism. The rest is determined by a chosen monoidal closed structure, products and coproducts in $\mathcal{SR}(\Sigma)_{\oplus\&}$.

For any A, $q \in Q(A)$ and $r \in R(A)_q$, we have maps $\tilde{\pi}_{q,r} : A \to_\Sigma [\![\mathbb{o}]\!]$ "reading" the value of the register r. Its second component is a $Q(A)$-indexed family of functions $\{\bullet\} \to \{r\}^*$; the component q takes the single-lettered word r as value and the other the empty word. For any object A of $\mathcal{SR}(\Sigma)_{\oplus\&}$, note that $\mathbf{I} \to_\Sigma A$ injects into $\sum_{q \in Q(A)} (\Sigma^*)^{R(A)_q}$; write ι for that injection and call $\lfloor A \rfloor_\Sigma$ its image. For any function $f : \Gamma^* \to \lfloor A \rfloor_\Sigma$, write $L[f]_q \subseteq \Gamma^*$ for the language $(f \circ \pi_1)^{-1}(q)$ when $q \in Q(A)$. For $r \in R(A)_q$, write $\rho[f]_{q,r}$ for the function $w \mapsto \tilde{\pi}^\dagger_{q,r}(w)(\bullet)$.

▶ **Definition F.6.** *We say that $f : \Gamma^* \to \lfloor A \rfloor_\Sigma$* is comparison-free polyregular *when:*

- *$L[f]_q$ is regular for all $q \in Q(A)$;*
- *$\rho[f]_{q,r}$ is a comparison-free polyregular function for all q, r.*

With this notion, we can tweak the interpretation of $\lambda\ell^{\oplus\&}$ terms to state our inductive invariant. Note that we cannot just use $[\![-]\!]$ which only interprets terms typable in Ψ_Σ for a fixed Σ, while we need an interpretation for terms typable in Ψ_Σ, $s : \mathtt{Str}_\Gamma[\tau]$.

▶ **Definition F.7.** *Suppose we are given a term Ψ_Σ, $s : \mathtt{Str}_\Gamma[\tau]$; $\cdot \vdash t : \sigma$. For each word $w \in \Gamma^*$, write $\underline{w}$ for the corresponding Church encoding. For every such w, we have Ψ_Σ; $\cdot \vdash t[\underline{w}/s]$; we define the interpretation $\langle\!|t|\!\rangle : \Gamma^* \to \lfloor A \rfloor_\Sigma$ so that $\langle\!|t|\!\rangle(w) = [\![t[\underline{w}/s]]\!]$*

With these definition in mind, we can state the suitable generalization of Theorem 7.1.

▶ **Theorem F.8.** *Let t be a $\lambda\ell^{\oplus\&}$-term typable in Ψ_Σ, $s : \mathtt{Str}_\Gamma[\tau] \vdash t : \sigma$ $\langle\!|t|\!\rangle : \Gamma^* \to \lfloor A \rfloor_\Sigma$ is comparison-free polyregular.*

Theorem 7.1 is then obtained as a corollary (taking $\sigma = \mathbb{o}$). The heart of the argument is contained in the following lemma.

▶ **Lemma F.9.** *Let $f_c : \Gamma^* \to \lfloor A \multimap A \rfloor_\Sigma$ for $c \in \Gamma$ and $g : \Gamma^* \to \lfloor A \rfloor_\Sigma$. Suppose that $L[g]_{q'}$ and $L[f_c]_{q''}$ are regular for all c, q', q'', and let $h : \Gamma^* \to \lfloor A \rfloor_\Sigma$ by defined as the map*

$$w = w[1] \dots w[n] \quad \longmapsto \quad \iota(\Lambda'(f_{w[n]}(w)) \circ \dots \circ \Lambda'(f_{w[1]}(w)) \circ (\iota^{-1}(g(w))))$$

where, for $\varphi \in \lfloor A \multimap A \rfloor_\Sigma$, we write $\Lambda'(\varphi)$ for the corresponding morphism $A \to A$ (i.e., $\mathrm{ev}_{A,A} \circ (\iota^{-1}(\varphi) \otimes \mathrm{id}_A) \circ \lambda_A^{-1}$ where $\mathrm{ev}_{A,B} : (A \multimap B) \otimes A \to_\Sigma B$ is the evaluation morphism and λ_A^{-1} is the inverse of the left unitor $\lambda_A : \mathbf{I} \otimes A \to_\Sigma A$ given by the monoidal closed structure of $\mathcal{SR}(\Sigma)_{\oplus\&}$).

For any $q \in Q(A)$ and $r \in R(A)_q$:

- *the language $L[h]_q$ is regular;*
- *the function $\rho[h]_{q,r}$ can be written as a composition by substitutions $\mathrm{CbS}(h', (\gamma_i)_{i \in I})$ where*
  - *$h' : \Sigma^* \to I^*$ is a regular function with*

$$I \quad = \quad \sum_{q' \in Q(A)} R(A)_{q'} \quad \sqcup \quad \Gamma \times \left(\sum_{q'' \in Q(A \multimap A)} R(A \multimap A)_{q''} \right)$$

  - *$\gamma_{q',r'} = \rho[g]_{q',r'}$ for $q' \in Q(A)$ and $r' \in R(A)_{q'}$*
  - *$\gamma_{c,q'',r''} = \rho[f_c]_{q'',r''}$ for $c \in \Gamma$, $q'' \in Q(A \multimap A)$ and $r'' \in R(A \multimap A)_{q''}$.*

Proof. This proof is split into two independent components: we first show that $L[h]_q$ is regular for every $q \in Q(A)$, and then how to implement $\rho[h]_{q,r}$ using composition by substitutions.

Regularity of $L[h]_q$. Let us write Finset for the category of finite sets and functions between them. The notation $Q(A)$ that we adopted for objects A of $\mathcal{SR}(\Sigma)_{\oplus\&}$ extends to a forgetful functor $Q : \mathcal{SR}(\Sigma)_{\oplus\&} \to \mathsf{Finset}$ by setting $Q(f, (\beta_q)_q) = f$ on morphisms. Recalling that ι is a bijection $(\mathbf{I} \to_\Sigma) \cong \lfloor A \rfloor_\Sigma$, we have $L[h]_q = \{w \mid Q((\iota^{-1} \circ h)(w))(\bullet) = q\}$ by definition. Unraveling the definition of h and applying the functoriality of Q, we also have

$$Q((\iota^{-1} \circ h)(w)) \quad = \quad Q(\Lambda'(f_{w[n]}(w))) \circ \cdots \circ Q(\Lambda'(f_{w[1]}(w))) \circ Q((\iota^{-1} \circ g)(w))$$

By hypothesis, for each $q' \in Q(A)$, the language $L_{\varepsilon,q'} = \{w \mid Q((\iota^{-1} \circ g)(w))(\bullet) = q'\}$ is regular. Further, for every $c \in \Gamma$ and $\delta \in Q(A)^{Q(A)}$, the language $L_{c,\delta} = \{w \mid Q(\Lambda'(f_c(w))) = \delta\}$ is also regular because we have

$$L_{c,\delta} \quad = \quad \bigcup_{\substack{q'' \in Q(A \multimap A) \\ \delta = Q(\mathrm{ev}_A)(q'', -)}} L[f_c]_{q''}$$

and each $L[f_c]_{q''}$ is regular by assumption. Therefore, we know that there is a finite monoid M, a monoid morphism $\varphi : \Gamma^* \to M$ and subsets $F_{\varepsilon,q'}$, $F_{c,\delta}$ (for $q' \in Q(A)$, $c \in \Sigma$ and $\delta \in Q(A)^{Q(A)}$) such that $L_{\varepsilon,q'} = \varphi^{-1}(F_{\varepsilon,q'})$ and $L_{c,\delta} = \varphi^{-1}(F_{c,\delta})$. Let us build a monoid morphism recognizing $L[h]_q$ from M. The target will be the monoid $(N, *)$ whose carrier is defined as

$$N \quad = \quad [\Gamma \to Q(A) \to Q(A)] \to M \times Q(A)^{Q(A)}$$

and whose multiplication is lifted pointwise from $M \times (Q(A) \to Q(A))$: if $n(\partial) = (m, \delta)$ and $n'(\partial) = (m', \delta')$, we have $(n * n')(\partial) = (mm', \delta \circ \delta')$. Define the morphism $\psi : \Gamma^* \to N$ as being generated by $\psi(c)(\partial) = (\varphi(c), \partial(c))$ and $F_q \subseteq N$ as the union

$$F_q = \bigcup_{q'} \bigcup_{\partial} F_{q,\partial,q'} \quad \text{where}$$

$$F_{q,\partial,q'} \quad = \quad \big\{n \mid n(\partial) = (m, \delta) \wedge \big(\forall c \in \Gamma \; m \in F_{c,\partial(c)}\big) \wedge m \in F_{\varepsilon,q'} \wedge \delta(q') = q\big\}$$

It can then be checked that we have $L[h]_q = \psi^{-1}(F_q)$, so we may conclude that $L[h]_q$ is a regular language. ◀

$\rho[h]_{q,r}$ as a CbS. Let $q_0 \in Q(A)$ and $(q_c)_{c\in\Gamma} \in Q(A \multimap A)^\Gamma$. We will show that we can define using such a composition by substitutions a function that coincides with $\rho[h]_{q,r}$ on the regular language $L \subseteq \Gamma^*$ defined as

$$L = L[g]_{q_0} \cap \bigcap_{c\in\Gamma} L[f_c]_{q_c}$$

From this, one can derive the desired conclusion concerning $\rho[h]_{q,r}$ using the closure of regular functions under regular conditionals (Lemma 5.2): combine the functions obtained for every combinations of q_0 and $(q_c)_c$, leveraging the fact that

$$\mathrm{CbS}((w \mapsto \texttt{if } w \in L' \texttt{ then } \alpha(w) \texttt{ else } \beta(w)), (\gamma_i)_{i\in I})$$

$$= \quad (w \mapsto \texttt{if } w \in L' \texttt{ then } \mathrm{CbS}(\alpha, (\gamma_i)_{i\in I})(w) \texttt{ else } \mathrm{CbS}(\beta, (\gamma_i)_{i\in I})(w))$$

For any $w \in \Gamma^*$, let $F_w : \mathcal{SR}(I)_{\oplus\&} \to \mathcal{SR}(\Sigma)_{\oplus\&}$ be the letter substitution functor induced by $i \mapsto \gamma_i(w)$. There exists $\hat{g}(q_0) : \mathbf{I} \to_I A$ *that does not depend on* w such that, if $w \in L[g]_{q_0}$, then $F_w(\hat{g}(q_0)) = g(w)$. The explicit expression is

$$\hat{g}(q_0) = \iota^{-1}(q_0, ((q_0, r))_{r \in R(A, q^0)})$$

Similarly, for each $c \in \Gamma$, there exists $\hat{f}_c(q_c) : \mathbf{I} \to_I (A \multimap A)$ such that for any $w \in L[f]_{q_c}$, we have $F_w(\hat{f}_c(q_c)) = \iota^{-1}(f_c(w))$. The functor F_w also preserves the monoidal closed structure on the nose, so we have $F_w(\Lambda'(\iota(\hat{f}_c(q_c)))) = \Lambda'(f_c(w))$.

By definition of L, all those conditions on w are implied by $w \in L$. By functoriality,

$$\forall w \in L,\ F_w(\underbrace{\tilde{\pi}_{q,r} \circ \Lambda'(\iota^{-1}(\hat{f}_{w[n]}(q_{w[n]}))) \circ \cdots \circ \Lambda'(\iota^{-1}(\hat{f}_{w[1]}(q_{w[1]}))) \circ (\hat{g}(q_0)))}_{\text{corresponds to a string } h'(w) \in I^* \text{ via } I^* \cong (\mathbf{I} \to_I [\![o]\!])} = \rho[h]_{q,r}(w)$$

Concerning the left-hand side, the function $h' : \Gamma^* \to I^*$, defined by the subexpression emphasized by the brace, has precisely the shape of a function of w computed by a *single-state* $\mathcal{SR}_{\oplus\&}$*-SST*, in the sense of [21]. As shown in [21, Section 3], such devices only compute *regular functions*. To conclude, observe that applying F_w amounts to performing a composition by substitutions: we obtain $\rho[h]_{q,r}(w) = \mathrm{CbS}(h', (\gamma_i)_{i \in I})(w)$ for $w \in L$ as advertised. ◀

◀

A couple of further lemmas stating that comparison-free polyregular functions play well with the categorical structure of $\mathcal{SR}(\Sigma)_{\oplus\&}$ are also helpful. The first one allows to combine two functions $f : \Gamma^* \to \lfloor A \rfloor_\Sigma$ and $g : \Gamma^* \to \lfloor B \rfloor_\Sigma$ into a single function $\Gamma^* \to \lfloor A \otimes B \rfloor_\Sigma$ that we write abusively $f \otimes g$. Its formal definition is

$$w \mapsto \iota((\iota^{-1}(f(w)) \otimes \iota^{-1}(g(w))) \circ \lambda_{\mathbf{I}}^{-1}) \qquad \text{(where } \lambda_{\mathbf{I}} \text{ is the unitor } \mathbf{I} \otimes \mathbf{I} \to_\Sigma \mathbf{I})$$

At the level of set-theoretic functions, this corresponds to a pairing seen through a canonical isomorphism $\lfloor A \otimes B \rfloor_\Sigma \cong \lfloor A \rfloor_\Sigma \times \lfloor B \rfloor_\Sigma$.

▶ **Lemma F.10.** *If $f : \Gamma^* \to \lfloor A \rfloor_\Sigma$ and $g : \Gamma^* \to \lfloor B \rfloor_\Sigma$ are comparison-free polyregular, so is $f \otimes g$.*

Proof. Recall that $Q(A \otimes B) = Q(A) \times Q(B)$; it is easy to check that we have $L[f \otimes g]_{(q,q')} = L[f]_q \cap L[g]_{q'}$, so $L[f \otimes g]_{q,q'}$ is regular. For $r \in R(A)_q$, we have $\rho[f \otimes g]_{(q,q'),r} = \rho[f]_{q,r}$ and for $r' \in R(B)_q$, we have $\rho[f \otimes g]_{(q,q'),r'} = \rho[g]_{q',r'}$; hence, for every $r \in R(A \otimes B)$, we know that $\rho[f \otimes g]_{(q,q'),r}$ is comparison-free polyregular, so we may conclude. ◀

Our second lemma pertains to the action of $\mathcal{SR}(\Sigma)_{\oplus\&}$ on the sets $\lfloor A \rfloor_\Sigma$. For a morphism $h : A \to_\Sigma B$, call $h^\circ : \lfloor A \rfloor_\Sigma \to \lfloor B \rfloor_\Sigma$ the map defined by $h^\circ(e) = \iota(h \circ (\iota^{-1}(e)))$.

▶ **Lemma F.11.** *Suppose that we have $h : A \to_\Sigma B$. If $f : \Gamma^* \to \lfloor A \rfloor_\Sigma$ is comparison-free polyregular, then so is $h^\circ \circ f$.*

Proof. Fix $q \in Q(B)$ and suppose that $h = (\delta, \alpha)$ (recall that by definition we have $\delta : Q(A) \to Q(B)$ and $\alpha : R(B) \to (\Sigma \cup R(A))^*$). Then we have

$$L[h^\circ \circ f]_q \quad = \quad \bigcup_{q' \in \delta^{-1}(q)} L[f]_{q'}$$

so $L[h^\circ \circ f]_q$ is regular. Now, fix $r \in R(B)_q$ and consider the family $(f_i)_{i \in \Sigma \cup R(A)}$ such that $f_a(w) = a$ for $a \in \Sigma$ and $f_{r'}(w) = \rho[f]_{q,r'}$ for $r' \in R(A)$; all of these functions are comparison-free polyregular. Therefore, $\rho[h^\circ \circ f]_{q,r} = \mathrm{CbS}((w \mapsto \alpha(r)), (f_i)_i)$[11] is also comparison-free polyregular. ◀

[11] Since $w \mapsto \alpha(r)$ is a constant function $\Gamma^* \to (\Sigma \cup R(A))^*$, this use of CbS is actually somewhat spurious; it can be shown that we have $\mathrm{rk}(\mathrm{CbS}(c_r, (f_i)_i)) \leq \max_{i \in R(A)} \mathrm{rk}(f_i)$ by induction over the length of $c_r(w)$, using that the pointwise concatenation of comparison-free polyregular functions does not increase the rank. This is a crucial observation for arguing that the rank of the function built in our proof is bounded by the number of occurrences of s occurring in a normalized input λ-term (as opposed to twice the number of occurrences of s).

We are now ready for the inductive proof.

Proof of Theorem F.8. As announced, we proceed by induction on the number of occurrences of $s : \mathtt{Str}_\Gamma[\tau]$ in t. In the base case where s does not occur in t, the function $\langle\!|t|\!\rangle$ is constant and therefore comparison-free polyregular. Suppose that we have k occurrences of s for some $k > 0$. By Lemma F.5, we can suppose that t has shape

$$o\ \left\langle s\ d_1\ \ldots\ d_{|\Sigma|}\ d_\varepsilon, ()\right\rangle$$

with $o : \tau \,\&\, \mathbf{I} \multimap \sigma$. There are strictly less than k occurrences of s in o, the d_is and d_ε, to we may apply the inductive hypothesis to deduce that $\langle\!|o|\!\rangle$, the $\langle\!|d_i|\!\rangle$s and d_ε are comparison-free polyregular. By unraveling the definition of $\langle\!|-|\!\rangle$ and Lemma F.9, we may thus conclude that $\left\langle\!\middle|s\ d_1\ \ldots\ d_{|\Sigma|}\ d_\varepsilon\middle|\!\right\rangle$ is comparison-free polyregular. Now, it can be checked that we have

$$\left\langle\!\middle|o\ \left\langle s\ d_1\ \ldots\ d_{|\Sigma|}\ d_\varepsilon, ()\right\rangle\middle|\!\right\rangle \quad = \quad \mathrm{ev}^\circ_{[\![\tau]\!]\&\mathbf{I},[\![\sigma]\!]} \circ \left(\langle\!|o|\!\rangle \otimes \left(p^\circ \circ \left\langle\!\middle|s\ d_1\ \ldots\ d_{|\Sigma|}\ d_\varepsilon\middle|\!\right\rangle\right)\right)$$

where p is the canonical morphism $[\![\tau]\!] \to_\Sigma [\![\tau]\!] \,\&\, \mathbf{I}$ pairing its input with the empty tuple. We may thus conclude the argument by applying successively Lemmas F.11, F.10 and F.11. ◀

F.3 Theorem F.4 and Lemma F.5

This section is devoted to establishing the required properties of $\beta\eta$-equivalence in the $\lambda\ell^{\oplus\&}$-calculus to make the proof of Theorem 7.1 go through. We build on some material from [21, Appendices B & C], where similar analyses were carried out. A central technique that we use here, which was not present in [21], is *focusing* (we briefly mentioned it at the beginning of Appendix F.2). A detailed exposition of focusing in the context of giving canonical normal forms to λ-terms may be found in [22].

First, we need to describe the shape of *focused normal forms*. These terms are split into three categories defined by mutual recursion:

- the general focused normal forms ($\mathrm{NF_f}$). We write $\Psi;\ \Delta \vdash_{\mathrm{NF_f}} t : A$ to say that t is a focused normal form such that $\Psi;\ \Delta \vdash t : A$.
- the focused neutral terms ($\mathrm{NE_f}$). We write $\Psi;\ \Delta \vdash_{\mathrm{NE_f}} t : A$ to say that t is a focused neutral form such that $\Psi;\ \Delta \vdash t : A$.
- the focused negative neutral terms ($\mathrm{NE_f^-}$). We write $\Psi;\ \Delta \vdash_{\mathrm{NE_f^-}} t : A$ to say that t is a negative focused neutral form such that $\Psi;\ \Delta \vdash t : A$.

Typed focused normal and neutral terms are defined by the inductive clauses in Figures 3, 4 and 5 in the following pages.

Proof sketch for Theorem F.4. We must prove that for every term t such that $\Psi;\ \Delta \vdash t : \tau$, we have some $\beta\eta$-equivalent t' such that $\Psi;\ \Delta \vdash_{\mathrm{NF_f}} t' : \tau$.

Focusing and normalization can be done simultaneously, but let us sketch how to get focused forms from normal forms for $\lambda\ell^{\oplus\&}$; this allows to conclude using [21, Theorem B.1]. Reusing the notations of [21, Appendix B], suppose that we have $\Psi;\ \Delta \vdash_{\mathrm{NF}} t : \tau$. We first claim that we have t' such that $t \to_\varepsilon^* t'$ and $t' \not\to_\varepsilon$; this can be shown by noticing that the following $\mathbb{N}$-valued complexity measure $\|-\|$ on $\lambda\ell^{\oplus\&}$-terms is strictly decreasing along $\to_\varepsilon$:

$$\begin{array}{lcllcl}
\|x\| & = & 0 & \|()\| & = & 0 \\
\|\langle\rangle\| & = & 0 & \|\mathsf{in}_1(t)\| & = & \|t\| \\
\|\mathsf{in}_2(t)\| & = & \|t\| & \|\pi_1(t)\| & = & \|t\| \\
\|\pi_2(t)\| & = & \|t\| & \|t\ u\| & = & \|t\| + \|u\| \\
\|t \otimes u\| & = & \|t\| + \|u\| & \|\langle t, u\rangle\| & = & \|t\| + \|u\| \\
\|\mathsf{abort}(t)\| & = & 1 + 2\|t\| & \|\mathsf{let}\ x \otimes y = t\ \mathsf{in}\ u\| & = & 1 + 2\|t\| + \|u\| \\
\|\mathsf{case}(t, x.u, y.v)\| & = & 1 + 2\|t\| + \|u\| + \|v\| & & &
\end{array}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_\mathfrak{f}} t : \tau}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \tau} \qquad \frac{\Psi;\ \Delta,\ x : \tau \vdash_{\mathrm{NF}_\mathfrak{f}} t : \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \lambda x.t : \tau \multimap \sigma} \qquad \frac{\Psi,\ x : \tau;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \lambda^! x.t : \tau \to \sigma}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \tau \qquad \Psi;\ \Delta' \vdash_{\mathrm{NF}_\mathfrak{f}} u : \sigma}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}_\mathfrak{f}} t \otimes u : \tau \otimes \sigma}$$

$$\frac{\Psi;\ \Delta' \vdash_{\mathrm{NE}^-_\mathfrak{f}} t : \tau \otimes \sigma \qquad \Psi;\ \Delta,\ x : \tau,\ y : \sigma \vdash_{\mathrm{NF}_\mathfrak{f}} u : \kappa}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}_\mathfrak{f}} \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ u : \kappa} \qquad \frac{}{\Psi;\ \cdot \vdash_{\mathrm{NF}_\mathfrak{f}} () : \mathbf{I}}$$

$$\frac{\Psi;\ \Delta' \vdash_{\mathrm{NE}^-_\mathfrak{f}} t : \mathbf{I} \qquad \Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} u : \kappa}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}_\mathfrak{f}} \mathsf{let}\ () = t\ \mathsf{in}\ u : \kappa} \qquad \frac{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \tau \qquad \Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} u : \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \langle t, u\rangle : \tau \,\&\, \sigma}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \tau}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \mathsf{in}_1(t) : \tau \oplus \sigma} \qquad \frac{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t : \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \mathsf{in}_2(t) : \tau \oplus \sigma}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}^-_\mathfrak{f}} t : \sigma \oplus \tau \qquad \Psi;\ \Delta',\ x : \sigma \vdash_{\mathrm{NF}_\mathfrak{f}} u : \kappa \qquad \Psi;\ \Delta',\ y : \tau \vdash_{\mathrm{NF}_\mathfrak{f}} v : \kappa}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}_\mathfrak{f}} \mathsf{case}(t, x.u, y.v) : \kappa}$$

$$\frac{}{\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} \langle\rangle : \top}$$

Figure 3 Focused normal forms for $\lambda\ell^{\oplus\&}$-terms.

By [21, Lemma B.2], we have that $\Psi;\ \Delta \vdash_{\mathrm{NF}} t' : \tau$. This combined with the fact that we have $t' \not\to_\varepsilon$ allows to show that we have $\Psi;\ \Delta \vdash_{\mathrm{NF}_\mathfrak{f}} t' : \tau$ by induction on the derivation. ◀

We now turn to the proof of Lemma F.5.

▶ **Definition F.12** (See also [21, Definition C.1]). *Write $\sqsubseteq_+$ for the least preorder relation over types satisfying the following for every types τ and σ*

$$\tau, \sigma \sqsubseteq_+ \tau \otimes \sigma \quad \tau, \sigma \sqsubseteq_+ \tau \oplus \sigma \quad \tau, \sigma \sqsubseteq_+ \tau \,\&\, \sigma \quad \sigma \sqsubseteq_+ \tau \multimap \sigma \quad \sigma \sqsubseteq_+ \tau \to \sigma$$

We say that τ is a strictly positive subtype *of σ whenever $\tau \sqsubseteq_+ \sigma$.*

▶ **Lemma F.13.** *If $\Psi;\ \Delta \vdash_{\mathrm{NE}^-_\mathfrak{f}} t : \tau$, then there is a variable in $\Psi;\ \Delta$ of type σ with $\tau \sqsubseteq_+ \sigma$.*

Proof. By induction over the derivation, much like [21, Lemma C.4]. ◀

▶ **Lemma F.14.** *Let $\tau = \kappa_1 \to \ldots \to \kappa_k \to \kappa'$ be a type and s a distinguished variable of type τ and $l < k$. Let $\Psi;\ \Delta$ be a purely linear contexts such that and t be a term such that $\Psi,\ s : \tau; \Delta \vdash_{\mathrm{NE}^-_\mathfrak{f}} t : \sigma_{l+1} \to \ldots \to \sigma_k \to \sigma'$ for some purely linear σ'. Then $\sigma' = \kappa'$ and $\sigma_i = \kappa_i$ for $i \geq l$ there are $d_1, \ldots, d_l$ such that*

$$t =_{\beta\eta} s\ d_1\ \ldots\ d_{k-1} \qquad \text{and} \qquad \Psi;\ \cdot \vdash_{\mathrm{NF}_\mathfrak{f}} d_i : \kappa_i \quad \text{for } i \in \{1, \ldots, k-1\}$$

Furthermore, there are no more occurrences of s in $s\ d_1\ \ldots d_{k-1}$ than in t.

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau}{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ t : \tau} \qquad \frac{\Psi;\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau \otimes \sigma \qquad \Psi;\ \Delta,\ x : \tau,\ y : \sigma \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ u : \kappa}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ u : \kappa}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma \oplus \tau \qquad \Psi;\ \Delta',\ x : \sigma \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ u : \kappa \qquad \Psi;\ \Delta',\ y : \tau \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ v : \kappa}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ \mathsf{case}(t, x.u, y.v) : \kappa}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : 0}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}}\ \mathsf{abort}(t) : \sigma}$$

Figure 4 Focused neutral forms for $\lambda\ell^{\oplus\&}$-terms.

$$\frac{}{\Psi;\ \Delta,\ x : \tau \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ x : \tau} \qquad \frac{}{\Psi,\ x : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ x : \tau} \qquad \frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau \,\&\, \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ \pi_1(t) : \tau}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau \,\&\, \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ \pi_2(t) : \sigma} \qquad \frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau \multimap \sigma \qquad \Psi;\ \Delta' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ u : \tau}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t\ u : \sigma}$$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \tau \to \sigma \qquad \Psi;\ \cdot \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ u : \tau}{\Psi;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t\ u : \sigma}$$

Figure 5 Focused negative neutral forms for $\lambda\ell^{\oplus\&}$-terms.

Proof. Straightforward induction on the judgment $\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma_{l+1} \to \ldots \to \sigma'$. This is a simplification of [21, Lemma C.5]; the hypotheses can be simplified since we restrict to negative neutral forms. ◀

Proof of Lemma F.5. We proceed by induction over the derivation of $\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t : \sigma$ to produce

$$\Psi,\ s : \tau; \Delta,\ \alpha : \kappa' \,\&\, \mathbf{I} \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ o : \sigma \qquad \Psi,\ s : \tau;\ \cdot \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ d_i : \kappa_i$$

such that $o[\langle s\ d_1\ \ldots\ d_k), ()\rangle / \alpha] =_{\beta\eta} t$ and that there be the same number of occurrences of s in both t and $(\lambda\alpha.o)\ \langle s\ d_1\ \ldots\ d_k, ()\rangle$. Since we shall need to often emulate the weakening of the variable α, we write $w_\alpha(u)$ for the term $\mathsf{let}\ () = \pi_2(\alpha)\ \mathsf{in}\ u$, noting that it is focused normal as long as u is either $\mathrm{NF}_{\mathfrak{f}}$ or $\mathrm{NE}_{\mathfrak{f}}^-$. For brevity, we use the notations o', $d_1 \ldots$ throughout for data obtained by applying the induction hypotheses without recalling all of the relevant property and merely explain how to build a suitable o.

- If the last rule applied is a variable lookup, then the term in question must be s itself. Furthermore, we must have $k = 0$, so we may simply take $o = \alpha$ to conclude.
- If the last rule considered is the following instance of the application rule

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma' \multimap \sigma \qquad \Psi,\ s : \tau;\ \Delta' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ u : \sigma'}{\Psi,\ s : \tau;\ \Delta,\ \Delta' \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t\ u : \sigma}$$

then, we have two subcases, according to whether there is an occurrence of s in t or not.
- If there is an occurrence of s in t, we apply the induction hypothesis to t to obtain some o' and d_is and set $o = o'\ u$ to conclude.
- Otherwise, we may apply the induction hypothesis to u to obtain some o' and d_is and set $o = t\ o'$ to conclude.

- If the last rule considered is the following instance of the application rule

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma' \to \sigma \qquad \Psi,\ s : \tau;\ \cdot \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ u : \sigma'}{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t\ u : \sigma}$$

then, we use Lemma F.14 to obtain $d_1, \dots, d_{k-1}$, set $d_k = u$ and $o = \alpha$ to conclude.
- If the last rule applied is a linear λ-abstraction

$$\frac{\Psi;\ \Delta,\ x : \sigma' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t : \sigma}{\Psi;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ \lambda x.t}$$

we may apply the induction hypothesis to obtain $o', d_1, \dots$ such that $t = o'[s\ d_1\ \dots]$. Note that x does not occur in any of the d_i and that we have $\Psi;\ \Delta,\ x : \sigma' \vdash o' : \kappa' \multimap \sigma$. So we may set $o = \lambda x.o'$ to conclude.
- t cannot be non-linear λ-abstraction as σ' is assumed to be purely linear.
- If the last rule applied is a $\otimes$-introduction

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t : \sigma \qquad \Psi,\ s : \tau;\ \Delta' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t' : \sigma'}{\Psi,\ s : \tau;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t \otimes t' : \sigma \otimes \sigma'}$$

then apply the induction hypothesis to the first premise if possible to get $o', d_1, \dots$ and set $o = o' \otimes t'$; otherwise, do the analogous operation with the second premise.
- If the last rule applied is a $\otimes$-elimination

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma \otimes \sigma' \qquad \Psi,\ s : \tau;\ \Delta',\ x : \sigma,\ y : \sigma' \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ u : \sigma''}{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ u : \sigma''}$$

first note that we have we have σ and σ' purely linear because of Lemma F.13 applied to the first premise. We have then two subcases, according to whether s occurs in t or not. If it does, apply the induction hypothesis to the first premise to get $o', d_1, \dots$ and set $o = \mathsf{let}\ x \otimes y = o'\ \mathsf{in}\ u$, otherwise apply it to the second premise and set $o = \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ o'$.
- If the last rule applied is a pairing

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t : \sigma \qquad \Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ t' : \sigma'}{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF}_{\mathfrak{f}}}\ \langle t, t' \rangle : \sigma\ \&\ \sigma'}$$

then apply the induction hypothesis to the first premise $o', d_1, \dots$ and set $o = \langle o', w_\alpha(t') \rangle$. If not possible because there is no occurrences of s in t, do the analogous thing with t'.
- If the last rule applied is a projection

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma_1\ \&\ \sigma_2}{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NE}_{\mathfrak{f}}^-}\ t : \sigma_i}$$

we apply the induction hypothesis to obtain $o', d_1, \dots$ and set $o = \pi_i\ o'$.
- The last rule applied cannot be an introduction of $\top$ because we need an occurrence of s.

- If the last rule applied is an introduction of $\oplus$

$$\frac{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF_f}} t : \sigma_i}{\Psi,\ s : \tau;\ \Delta \vdash_{\mathrm{NF_f}} \mathsf{in}_i(t) : \sigma_1 \oplus \sigma_2}$$

  we may simply apply the induction hypothesis to get o' and set $o = \mathsf{in}_i(o')$.
- If the last rule applied is an elimination of $\oplus$

$$\frac{\Psi;\ \Delta \vdash_{\mathrm{NE_f^-}} t : \sigma \oplus \sigma' \qquad \Psi;\ \Delta',\ x : \sigma \vdash_{\mathrm{NF}} u : \sigma'' \qquad \Psi;\ \Delta',\ y : \sigma' \vdash_{\mathrm{NF}} v : \sigma''}{\Psi;\ \Delta,\ \Delta' \vdash_{\mathrm{NF}} \mathsf{case}(t, x.u, y.v) : \sigma''}$$

  first note that σ and σ' are necessarily purely linear because of Lemma F.13. We then try to apply the induction hypothesis to one of the premise (at least one of them has an occurrence of s) to get some $o', d_1, \dots$
  - If we can apply it to the first premise so that $t =_{\beta\eta} o'[\langle s\ d_1 \dots d_k, () \rangle / \alpha]$, we can conclude by setting $o = \mathsf{case}(o', x.u, y.v)$.
  - If it is applicable to the second premise so that $u =_{\beta\eta} o'[\langle s\ d_1 \dots d_k, () \rangle / \alpha]$, we set $o = \mathsf{case}(t, x.o', y.w_\alpha(v))$.
  - Otherwise, $v =_{\beta\eta} o'[\langle s\ d_1\ \dots\ d_k, () \rangle / \alpha]$ and we set $o = \mathsf{case}(t, x.w_\alpha(u), y.o')$.

◀

Contents