Sequence Types for Hereditary Permutators

Pierre Vial
Inria, Nantes, France
pierre.vial@inria.fr

Abstract

The invertible terms in Scott's model $\mathcal{D}_\infty$ are known as the hereditary permutators. Equivalently, they are terms which are invertible up to $\beta\eta$-conversion with respect to the composition of the λ-terms. Finding a type-theoretic characterization to the set of hereditary permutators was problem # 20 of TLCA list of problems. In 2008, Tatsuta proved that this was not possible with an inductive type system. Building on previous work, we use an infinitary intersection type system based on sequences (i.e., families of types indexed by integers) to characterize hereditary permutators with a unique type. This gives a positive answer to the problem in the coinductive case.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases hereditary permutators, Böhm trees, intersection types, coinduction, ridigity, sequence types, non-idempotent intersection

Digital Object Identifier 10.4230/LIPIcs.FSCD.2019.33

Funding This research has been partially funded by the CoqHoTT ERC Grant 637339.

0 Introduction

The study of $\beta\eta$-invertible terms goes back to Curry and Feys [7], who showed that the only regular *combinators* having an inverse are of the form $\lambda x\, x_1 \dots x_n . x\, x_{\sigma(1)} \dots x_{\sigma(n)}$ with σ a permutation. Building on this work, Dezani [9] gave a characterization of the normal forms of all the invertible *normalizing* terms. This characterization was extended by Bergstra and Klop [3] for *any* term: $\beta\eta$-invertible terms were proved to have Böhm trees of a certain form, generalizing that given by Curry and Feys and suggesting to name them *hereditary permutators*.

On another hand, intersection types systems were introduced by Coppo and Dezani [6, 12] around 1980 (see [16] for a survey). They were extensively used to characterize various sets of terms having common semantic properties (including head, weak, strong normalization) in different calculi. Yet, hereditary permutators resisted such a characterization, so that the problem of finding a type system assigning a unique type to all hereditary permutators (and only to them) was inscribed in TLCA list of open problems by Dezani in 2006 (Problem # 20). Two years later, Tatsuta [14] proved that the set `HP` of hereditary head permutators is not recursively enumerable. This entails that `HP` cannot be characterized in an *inductive* type system.

However, in [17], using a *coinductive intersection type system* named system `S`, we characterized the so-called set of *hereditary head normalizing (HHN)* terms which is also a set of terms having Böhm trees of certain form (without the constant $\bot$), whereas this set was also proved not to be recursively enumerable by Tatsuta [13]. As in the finitary case, infinite types bring simpler semantic proofs of well-known theorems, e.g., system `S` helps proving that an asymptotic reduction strategy produces the infinitary normal form of a term when it exists. In this paper, we extend system `S` with a type constant characterizing the set of hereditary permutators and we thus give a positive answer to TLCA Problem # 20 in the coinductive case. This also proves that infinitary type systems may be used to characterize other sets of Böhm trees.


© Pierre Vial;
licensed under Creative Commons License CC-BY

Before properly starting the article, a few words should be said on system $\mathtt{S}$ and infinitary typing: intersections are represented by families of types indexed by sets K of integers $\geqslant 2$. These indexes are called *tracks*. Thus, system $\mathtt{S}$ is close to *non-idempotent* intersection, introduced by Gardner [10] and de Carvalho [5], for which $A \wedge A \neq A$. In the finite case, non-idempotency gives very simple proofs of normalization (see [4] for a survey). Tracks allow tracing occurrences of a given type in a derivation (*rigidity*) while ensuring syntax-direction, whereas having both is not possible when non-idempotent intersection is represented by lists or multisets. Rigidity is crucial in the infinitary case, because coinductive type grammars give birth to unsound derivations, e.g., the unsolvable term $\Omega := (\lambda x.x\,x)(\lambda x.x\,x)$ becomes typable. However, rigidity allows defining a validity criterion, called approximability, which brings back semantic soundness. This is why system $\mathtt{S}$ provides a good framework to characterize hereditary permutators.

Last, Tatsuta defines a type system with a family of type constants $\mathtt{ptyp}_d$ (with $d \in \mathbb{N}$) such that $t : \mathtt{ptyp}_d$ iff t is a hereditary permutator on d levels. Then, a term is a hereditary permutator iff $t : \mathtt{ptyp}_d$ is derivable for all $d \in \mathbb{N}$. However, given a hereditary permutator t, there is no explicit relation between the different typings $t : \mathtt{ptyp}_d$ when d ranges over $\mathbb{N}$. We reuse this idea here, but the notion of approximability hinted at above allows formally expressing the typing derivations concluding with $t : \mathtt{ptyp}_d$ as *extensions* of those concluding with $t : \mathtt{ptyp}_{d_0}$ with $d_0 < d$. Actually, we define a type constant $\mathtt{ptyp}$, which can be assigned to hereditary permutators and to them only, which is the "supremum" of all $\mathtt{ptyp}_d$ i.e., such that a typing $t : \mathtt{ptyp}$ is an extension of typings $t : \mathtt{ptyp}_d$ for all $d \in \mathbb{N}$.

Structure of the paper. We conclude this introduction with some technical background on hereditary permutators. Section 1 recalls some basic definitions about Böhm trees and the infinite λ-calculus, but also on system $\mathtt{S}$ and infinite types. In Section 2, we give a type-theoretic characterization of hereditary permutations in system $\mathtt{S}$. In Section 3, we introduce system $\mathtt{S}_{\mathtt{hp}}$, an extension of system $\mathtt{S}$, such that hereditary permutators have a unique type. The technical contributions of this paper are found mainly in Section 2 and 3.1.

Hereditary Permutators

Let $\mathcal{V}$ be a set of term variables. For all $n \in \mathbb{N}$, $\mathfrak{S}_n$ denotes the set of permutations of $\{1, \dots, n\}$, $\to_{\mathtt{h}}$ denotes head reduction and the reflexive-transitive closure of a reduction $\to_{\mathcal{R}}$ is denoted $\to^*_{\mathcal{R}}$. To define hereditary permutators, we first consider *headed* hereditary permutators, i.e., hereditary permutators whose head variables have not been bound yet.

▶ **Definition 1.**

- *For all $x \in \mathcal{V}$, the sets $\mathtt{HP}(x)$ of **x-headed Hereditary Permutators (x-HP)** ($x \in \mathcal{V}$) are defined by mutual coinduction:*

$$\frac{h_1 \in \mathtt{HP}(x_1)\ \dots\ h_n \in \mathtt{HP}(x_n) \qquad (n \geqslant 0, \sigma \in \mathfrak{S}_n,\ x_i \neq x,\ x_i \text{ pairwise distinct}) \\ \text{and } h \to^*_{\mathtt{h}} \lambda x_1 \dots x_n.x\, h_{\sigma(1)} \dots h_{\sigma(n)}}{h \in \mathtt{HP}(x)}$$

- *A closed hereditary permutator, or simply, a **Hereditary Permutator (HP)** is a term of the form $h = \lambda x.h_0$ with $h_0 \in \mathtt{HP}(x)$ for some x.*

A headed hereditary permutator is the head reduct of a hereditary permutator applied to a variable.

▶ **Theorem 2** ([3]). *A λ-term t is a hereditary permutator iff t is invertible modulo $\beta\eta$-conversion for the operation $\cdot$ defined by $u \cdot v = \lambda x.u\,(v\,x)$, whose neutral element is $I = \lambda x.x$.*

Thus, u is invertible when there exists v such that $\lambda x.u(v\,x) =_{\beta\eta} = \lambda x.v(u\,x) =_{\beta\eta} I$. An extensive presentation of hereditary permutators and their properties is given in Chapter 21 of [2].

1 Infinite terms and types

In this section, we present Böhm trees (Chapter 10 of [2]) and the construction of one of the infinitary calculi introduced in [11]. See also [8, 1] for alternative presentations. We then present system $\mathtt{S}$, an infinitary intersection type system with a validity criterion (approximability) discarding unsound coinductive derivations and using sequences to represent intersection. Some more details can also be found in [17].

General notations. The set of finite words on $\mathbb{N}$ is denoted with $\mathbb{N}^*$, ε is the empty word, $a \cdot a'$ the concatenation of a and a'. The prefix order $\preceq$ is defined on $\mathbb{N}^*$ by $a \preceq a'$ if there is a_0 such that $a' = a \cdot a_0$, e.g., $2{\cdot}3 \preceq 2{\cdot}3{\cdot}0{\cdot}1$.

Intuitively, 0 is dedicated to the constructor λx, 1 is dedicated to the left-hand side of applications and all the $k \geqslant 2$ to the possibly multiple typings of the arguments of applications. This also explains why 0 and 1 will have a particular status in the definitions to come. For instance, the **applicative depth** $\mathtt{ad}(a)$ of $a \in \mathbb{N}^*$ is the number of nestings inside arguments, i.e., $\mathtt{ad}(a)$ is defined inductively by $\mathtt{ad}(\varepsilon) = 0$, $\mathtt{ad}(a \cdot k) = \mathtt{ad}(a)$ if $k = 0$ or $k = 1$ and $\mathtt{ad}(a \cdot k) = \mathtt{ad}(a) + 1$ if $k \geqslant 2$. The **collapse** is defined on $\mathbb{N}$ by $\overline{k} = \min(k, 2)$ and on $\mathbb{N}^*$ inductively by $\overline{\varepsilon} = \varepsilon$, $\overline{a \cdot k} = \overline{a} \cdot \overline{k}$, e.g., $\overline{7} = 2$, $\overline{1} = 1$ and $\overline{2{\cdot}3{\cdot}0{\cdot}1} = 2{\cdot}2{\cdot}0{\cdot}1$. These notions are straightforwardly extended to words of infinite length, e.g., 2^ω, which is the infinite repetition of 2.

1.1 Infinite Lambda Terms

The set Λ^∞ of infinitary λ-terms is coinductively defined by:

$$t, u \;:=\; x \in \mathcal{V} \parallel (\lambda x.t) \parallel (t\,u)$$

When there is no ambiguity, we usually just write $\lambda x.t$ and $t\,u_1 \dots u_n$ instead of $(\lambda x.t)$ and $(\dots(t\,u_1)\dots u_n)$. If t is an infinitary term, then $\mathtt{supp}(t)$, the **support** of t (the set of positions in t) is defined in the usual way, i.e., coinductively, $\mathtt{supp}(x) = \{\varepsilon\}$, $\mathtt{supp}(\lambda x.t) = \{\varepsilon\} \cup 0 \cdot \mathtt{supp}(t)$ and $\mathtt{supp}(t\,u) = \{\varepsilon\} \cup 1 \cdot \mathtt{supp}(t) \cup 2 \cdot \mathtt{supp}(u)$. If $\overline{a} \in \mathtt{supp}(t)$, the subterm (resp. the constructor) of t at position $\overline{a}$ is denoted $t|_a$ (resp. $t(a)$), e.g., if $t = \lambda x.(x\,y)z$ and $a = 0 \cdot 1$ (resp. $a = 0 \cdot 4$), then $t|_a = x\,y$ and $t(a) = @$) (resp. $t|_a = t(a) = z$).

▶ **Definition 3** (001-Terms). *Let $t \in \Lambda^\infty$. Then t is a **001-term**, if, for all infinite branches γ in $\mathtt{supp}(t)$, $\mathtt{ad}(\gamma) = \infty$.*

Once again, the vocable "001-term" comes from [11]. For instance, the 001-term f^ω is formally defined as the tree such that $\mathtt{supp}(f^\omega) = \{2^n \mid n \in \mathbb{N}\} \cup \{2^n \cdot 1 \,|\, n \in \mathbb{N}\}$, $f^\omega(2^n) = @$ and $f^\omega(2^n \cdot 1) = f$ for all $n \in \mathbb{N}$. Its unique infinite branch is 2^ω (since all the finite prefixes of 2^ω are in $\mathtt{supp}(f^\omega)$), which satisfies $\mathtt{ad}(2^\omega) = \infty$. In contrast, the infinite term t defined by $t = t\,x$, so that $t = (((\dots)x)x)x$, is *not* a 001-term: indeed, $\mathtt{supp}(t) = \{1^n \,|\, n \in \mathbb{N}\} \cup \{1^n \cdot 2 \,|\, n \in \mathbb{N}\}$, so $\mathtt{supp}(t)$ has the infinite branch 1^ω (this indicates a leftward infinite branch), which satisfies $\mathtt{ad}(1^\omega) = 0$ since 2 does not occur in 1^ω.

1.2 The computation of Böhm trees

The notation $t[u/x]$ denotes the term obtained from t by the capture-free substitution of the occurrences of x with u ([11] gives a formal definition in the infinitary calculus). The β-reduction $\to_\beta$ is obtained by the contextual closure of $(\lambda x.t)u \to_\beta t[u/x]$ and $t \overset{b}{\to}_\beta t'$ denotes the reduction of a redex at position b in t, e.g., $\lambda y.((\lambda x.x)u)v \overset{0\cdot1}{\to}_\beta \lambda y.u\,v$. A **001-Normal Form (001-NF)** is a 001-term that does not contain a redex. A 001-term is **solvable** if $t \to_{\mathtt{h}}^* \lambda x_1 \dots x_p.x\,t_1 \dots t_q$, which is a head normal form (HNF) of **arity** p.

▶ **Definition 4** (Böhm tree of a term). *Let t be a 001-term.*
The Böhm tree $\mathtt{BT}(t)$ of t is coinductively defined by:
- $\mathtt{BT}(t) = \lambda x_1 \dots x_p.x\,\mathtt{BT}(t_1) \dots \mathtt{BT}(t_q)$ *if* $t \to_{\mathtt{h}}^* \lambda x_1 \dots x_p.x\,t_1 \dots t_q$.
- $\mathtt{BT}(t) = \bot$ *if t is unsolvable.*

For instance, $\mathtt{BT}(\Omega) = \bot$ where $\Omega = (\lambda x.x\,x)(\lambda x.x\,x)$ and $\mathtt{BT}(t) = t$ if t a 001-normal form. Definition 1 can be read as the specification of a set of terms whose Böhm trees have a particular form. Intuitively, the computation of Böhm trees is done by a possibly infinite series of head reductions at deeper and deeper levels. This corresponds to an asymptotic reduction strategy known as *hereditary head reduction*.

Some reduction paths are of infinite length but asymptotically produce a term.

▶ **Definition 5** (Productive reduction paths). *Let $t = t_0 \overset{b_0}{\to}_\beta t_1 \overset{b_1}{\to}_\beta t_2 \dots t_n \overset{b_n}{\to}_\beta t_{n+1} \dots$ be a reduction path of length $\ell \leqslant \omega$.*
*Then, this reduction path is said to be **productive** if either it is of finite length ($\ell \in \mathbb{N}$), or $\ell = \infty$ and $\mathtt{ad}(b_n)$ tends to infinity (recall that $\mathtt{ad}(\cdot)$ is applicative depth).*

A productive reduction path is called a *strongly converging reduction sequence* in [11], in which numerous examples are found. When $\mathtt{BT}(t)$ does not contain $\bot$, the hereditary head reduction strategy on a term t gives a particular case of productive path.

▶ **Lemma 6** (Limits of productive paths). *Let $t = t_0 \overset{b_0}{\to}_\beta t_1 \overset{b_1}{\to}_\beta t_2 \dots t_n \overset{b_n}{\to}_\beta t_{n+1} \dots$ be a productive reduction path of infinite length.*
Then, there is a 001-term t' such that, for every $d \geqslant 0$, there is $N \in \mathbb{N}$ such that, for all $n \geqslant N$, $\mathtt{supp}(t_n) \cap \{b \in \{0,1,2\}^* \mid \mathtt{ad}(b) \leqslant d\} = \mathtt{supp}(t') \cap \{b \in \{0,1,2\}^* \mid \mathtt{ad}(b) \leqslant d\}$.

The term t' in the statement of Lemma 6 is called the **limit** of the productive path. Intuitively, when t' is the limit of $(t_n)_{n\geqslant 0}$, then t' induces the same tree as t_n at fixed applicative depth after sufficiently many reduction steps. We then write $t \to_\beta^\infty t'$ if $t \to_\beta^* t'$ or t is the limit of a productive path starting at t. For instance, if $\Delta_f = \lambda x.f(x\,x)$, $\mathtt{Y}_f = \Delta_f\,\Delta_f$ (with $f \in \mathcal{V}$), then $\mathtt{Y}_f \overset{\varepsilon}{\to}_\beta f(\mathtt{Y}_f)$, which gives the productive path $\mathtt{Y}_f \overset{\varepsilon}{\to}_\beta f(\mathtt{Y}_f) \overset{2}{\to}_\beta \dots f^n(\mathtt{Y}_f) \overset{2^n}{\to}_\beta f^{n+1}(\mathtt{Y}_f) \dots$ since $\mathtt{ad}(2^n) \longrightarrow \infty$. The limit of this path – which implements hereditary head reduction on $\mathtt{Y}_f$ – is f^ω, i.e., $\mathtt{Y}_f \to_\beta^\infty f^\omega$ and also $\mathtt{BT}(\mathtt{Y}_f) = f^\omega$.

A 001-term t is said to be **infinitary weakly normalizing ($\mathbf{WN}_\infty$)** if there is a 001-NF t' such that $t \to_\beta^\infty t'$. It turns out that t is WN_∞ iff its Böhm tree does not contain $\bot$. The result is proved in [11] in a syntactical way, but we give a semantic proof of this fact in [17].

1.3 System S (sequential intersection)

A **sequence** of elements of a set X is a family $(x_k)_{k\in K}$ with $K \subseteq \mathbb{N} \setminus \{0,1\}$. In this case, if $k_0 \in K$, x_{k_0} is the element of $(x_k)_{k\in K}$ **on track** k. We often write $(k{\cdot}x_k)_{k\in K}$ for $(x_k)_{k\in K}$, which, for instance, allows us to denote by $(2\cdot a, 4\cdot b, 5\cdot a)$ or $(4\cdot b, 2\cdot a, 5\cdot a)$ the sequence

$(x_k)_{k\in K}$ with $K = \{2, 4, 5\}$, $x_2 = x_5 = a$ and $x_4 = b$. In this sequence, the element on track 4 is b. Sequences come along with a **disjoint union** operator, denoted $\uplus$. Let $(x_k)_{k\in K}$ and $(x'_k)_{k\in K'}$ be two sequences:

- If $K \cap K' = \emptyset$, then $(x_k)_{k\in K} \uplus (x'_k)_{k\in K'}$ is $(x'')_{k\in K''}$ with $K'' = K \cup K'$ and $x''_k = x_k$ when $k \in K$ and $x''_k = x'_k$ when $k \in K'$.
- If $K \cap K' \neq \emptyset$, $(x_k)_{k\in K} \uplus (x'_k)_{k\in K'}$ is not defined.

The operator $\uplus$ is partial, associative and commutative.

Let $\mathcal{O}$ be a set of type atoms o. The set of $\mathtt{S}$-types is coinductively defined by:

$$T, S_k \quad ::= \quad o \in \mathcal{O} \parallel (S_k)_{k\in K} \to T$$

A sequence of types $(S_k)_{k\in K}$ is called a **sequence type** and it represents an intersection of types. The types of system $\mathtt{S}$ collapse on usual non-idempotent intersection types built on multisets [4], e.g., the $\mathtt{S}$-types $(2 \cdot o, 3 \cdot o', 4 \cdot o) \to o$ and $(2 \cdot o', 8 \cdot o, 9 \cdot o) \to o$ collapse on $[o, o, o'] \to o$. The system is *strict* [12, 15, 16] since intersections occur only on left-hand sides of arrows. The **domain** and **codomain** of an arrow type are defined by $\mathtt{dom}((S_k)_{k\in K} \to T) = (S_k)_{k\in K}$ and $\mathtt{codom}((S_k)_{k\in K} \to T) = T$. The **arity** of a type is coinductively defined by $\mathtt{ar}(o) = 0$ and $\mathtt{ar}((S_k)_{k\in K} \to T) = \mathtt{ar}(T) + 1$. For instance, if T is defined by $T = (2 \cdot o) \to T = (2 \cdot o) \to (2 \cdot o) \to \ldots$, then $\mathtt{ar}(T) = \infty$.

The **support** of a type or a sequence type U is coinductively defined by $\mathtt{supp}(o) = \{\varepsilon\}$, $\mathtt{supp}((S_k)_{k\in K}) = \cup_{k\in K} k \cdot \mathtt{supp}(S_k)$ and $\mathtt{supp}((S_k)_{k\in K} \to T) = \{\varepsilon\} \cup \mathtt{supp}((S_k)_{k\in K}) \cup 1 \cdot \mathtt{supp}(T)$. Since $1 \notin K$ by convention, this definition is correct. If $c \in \mathtt{supp}(U)$, then $U|_c$ denotes the type or sequence type rooted at position c in U ($U|_c$ is a type when U is a type or $c \neq \varepsilon$). For instance, if $U = (2 \cdot o) \to (2 \cdot o, 3 \cdot o') \to o$ and $c = 1$, then $U|_c = (2 \cdot o, 3 \cdot o') \to o$. Likewise, $U(c)$ denotes the type constructor ($o \in \mathcal{O}$ or $\to$) at position c. With the same example, $U(\varepsilon) = U(1) = \to$, $U(2) = o$ and $U(1{\cdot}3) = o'$

A **001-type** is a $\mathtt{S}$-type T such that, for all $c \in \mathtt{supp}(T)$, $\mathtt{ar}(T|_c) < \infty$, where $T|_c$ is the subtree rooted at c in T ($T|_c$ is a type). The **target type** $\mathtt{targ}(T)$ of a 001-type S is *inductively* defined by $\mathtt{targ}(o) = o$ and $\mathtt{targ}((S_k)_{k\in K} \to T) = \mathtt{targ}(T)$.

An $\mathtt{S}$-context C (or D) is a total function from $\mathcal{V}$ to the set of $\mathtt{S}$-types. The operator $\uplus$ is extended point-wise. An $\mathtt{S}$-judgment is a triple $C \vdash t : T$, where C, t and T are respectively an $\mathtt{S}$-context, a 001-term and an $\mathtt{S}$-type. A **sequence judgment** is a sequence of judgments $(C_k \vdash t : T_k)_{k\in K}$ with $K \subseteq \mathbb{N} \setminus \{0, 1\}$. For instance, if $8 \in K$, the judgment $C_8 \vdash t : T_8$ is specified on track 8. The set of $\mathtt{S}$-derivations is defined coinductively by:

$$\frac{}{x : (k \cdot T) \vdash x : T}\,\mathtt{ax} \qquad \frac{C; x : (S_k)_{k\in K} \vdash t : T}{C \vdash \lambda x.t : (S_k)_{k\in K} \to T}\,\mathtt{abs}$$

$$\frac{C \vdash t : (S_k)_{k\in K} \to T \qquad (D_k \vdash u : S_k)_{k\in K}}{C \uplus (\uplus_{k\in K} D_k) \vdash t\,u : T}\,\mathtt{app}$$

In $\mathtt{app}$, K may be empty, and then u is untyped. We call $\mathtt{S}_0$, the restriction of system $\mathtt{S}$ to finite types and contexts, but allowing infinite terms. The derivation $P_{\mathtt{ex}}$ below is in $\mathtt{S}_0$.

Let P be a $\mathtt{S}$-derivation typing a term t. The **support** of P is the set of positions of judgments inside P defined in the expected way: 0 to visit the premise of an $\mathtt{abs}$-rule, 1 to visit the left-hand side of an $\mathtt{app}$-rule and $k \geqslant 2$ to visit an argument judgment on track k on the right-hand side of the $\mathtt{app}$-rule. Thus, if $a \in \mathtt{supp}(P)$, $P(a)$, which denotes the judgment at position a in P, types the subterm $t|_a$. We denote the type and the context of $P(a)$ by $\mathtt{T}^P(a)$ and $\mathtt{C}^P(a)$, so that $P(a) = \mathtt{C}^P(a) \vdash t|_a : \mathtt{T}^P(a)$. Moreover:

- If $a \in \mathtt{supp}(P)$ and $c \in \mathtt{supp}(\mathtt{T}^P(a))$, then the pair (a, c) is a **right biposition** of P and $P(a, c)$ denotes $\mathtt{T}^P(a)(c)$, which is a type constructor ($o \in \mathcal{O}$ or $\to$).

- If $a \in \mathtt{supp}(P)$, $x \in \mathcal{V}$ and $k \cdot c \in \mathtt{supp}(\mathtt{C}^P(a)(x))$, then the triple $(a, x, k \cdot c)$ is a **left biposition** in P and $P(a, x, c)$ denotes $\mathtt{C}^P(a)(x)(c)$.

The set of bipositions of P is called the **bisupport** of P and is denoted by $\mathtt{bisupp}(P)$. An $\mathtt{S}$-derivation P is **finite**, i.e., is a derivation of system $\mathtt{S}_0$, iff $\mathtt{bisupp}(P)$ is a finite set. If $a \in \mathtt{supp}(P)$ and $t(a) = x$, $P(a)$ is an $\mathtt{ax}$-rule and $\mathtt{C}^P(a) = x : (k \cdot \mathtt{T}^P(a))$.

▶ **Example 7.** In the derivation, the notation [7] indicates that the judgment $f : 2\cdot(\,)\to o \vdash f^\omega : o$ is on track 7.

$$P_{\mathtt{ex}} = \dfrac{\dfrac{\dfrac{}{x : (2\cdot(7\cdot o) \to o') \vdash x : (7\cdot o) \to o'}\,\mathtt{ax} \qquad \dfrac{\dfrac{}{f : (2\cdot(\,) \to o) \vdash f : (\,) \to o}\,\mathtt{ax}}{f : (2\cdot(\,) \to o) \vdash f^\omega : o\ [7]}\,\mathtt{app}}{x : (2\cdot(7\cdot o) \to o'), f : (2\cdot(\,) \to o) \vdash x\, f^\omega : o'}\,\mathtt{app}}{x : (2\cdot(7\cdot o) \to o') \vdash \lambda f.x\, f^\omega : (2\cdot(\,) \to o) \to o'}\,\mathtt{abs}$$

We have $\mathtt{supp}(P_{\mathtt{ex}}) = \{\varepsilon, 0, 0{\cdot}1, 0{\cdot}7, 0{\cdot}7{\cdot}1\}$. Remark how f^ω is typed in the derivation using the type $(\,) \to o$. We have $P_{\mathtt{ex}}(0{\cdot}7{\cdot}1) = f : (2\cdot(\,)\to o) \vdash f : (\,)\to o$ and $\mathtt{T}^{P_{\mathtt{ex}}}(0{\cdot}1) = (7\cdot o)\to o$. Since $\mathtt{supp}(\mathtt{T}^{P_{\mathtt{ex}}}) = \{\varepsilon, 7, 1\}$ and $\mathtt{T}^{P_{\mathtt{ex}}}(\varepsilon) = \to$, $\mathtt{T}^{P_{\mathtt{ex}}}(7) = o$ and $\mathtt{T}^{P_{\mathtt{ex}}}(1) = o'$, we have $(0{\cdot}1, \varepsilon), (0{\cdot}1, 7), (0{\cdot}1, 1) \in \mathtt{bisupp}(P_{\mathtt{ex}})$ and $P_{\mathtt{ex}}(0{\cdot}1, \varepsilon) = \to$, $P_{\mathtt{ex}}(0{\cdot}1, 7) = o$, $P_{\mathtt{ex}}(0{\cdot}1, 1) = o'$. Likewise, $\mathtt{T}^{P_{\mathtt{ex}}}(0{\cdot}7) = o$ and $\mathtt{C}^{P_{\mathtt{ex}}}(0) = x : (2\cdot(7\cdot o)\to o), f : (2\cdot(\,)\to o)$, so that, e.g., $(0, x, 2{\cdot}7), (0, f, 2) \in \mathtt{bisupp}(P_{\mathtt{ex}})$, $P_{\mathtt{ex}}(0, x, 2{\cdot}7) = o$, $P_{\mathtt{ex}}(0, f, 2) = \to$.

A derivation P is quantitative when the context is computable from the axiom rules:

▶ **Definition 8** (Quantitative derivation). *Let P be a $\mathtt{S}$-derivation. Then P is **quantitative** if, for all $a \in \mathtt{supp}(P)$, $x \in \mathcal{V}$, $k \in \mathbb{N} \setminus \{0, 1\}$ such that $(a, x, k) \in \mathtt{bisupp}(P)$, there is $a_0 \succeq a$ such that $P(a_0) = x : (k \cdot S) \vdash x : S$.*

Observe that, in a quantitative derivation $P \rhd C \vdash t : T$, if $C(x) = (k \cdot S)$ i.e., x is assigned a singleton sequence type, then there is exactly one $\mathtt{ax}$-rule typing x (we use this in the proof of Claim 21).

An example of non-quantitative derivations is given by the family $(P_k)_{k\geqslant 2}$ defined by:

$$P_k = \dfrac{\dfrac{}{f : (k\cdot(2\cdot o)\to o) \vdash f : (2\cdot o)\to o}\,\mathtt{ax} \qquad P_{k+1} \rhd f : (\ell\cdot(2\cdot o)\to o)_{\ell\geqslant k+1}, x{:}(2\cdot o) \vdash f^\omega : o\ [2]}{f : (\ell\cdot(2\cdot o)\to o)_{\ell\geqslant k}, x : (2\cdot o) \vdash f^\omega : o}\,\mathtt{app}$$

The P_k type f^ω with o but they assign a non-empty sequence type to $x \notin \mathtt{fv}(f^\omega)$: this is why there are not quantitative. Indeed, $(\varepsilon, x, 2) \in \mathtt{bisupp}(P_k)$, but there is no $a_0 \in \mathtt{supp}(P_k)$ such that $P_k(a_0) = x : (2\cdot o) \vdash x : o$. Remark how the infinite branch of f^ω is used to assign a type to x whereas it does not occur in the subject. In contrast, if t is a finite λ-term, every derivation typing t is quantitative. However, Lemma 16 below states that quantitativity is a sufficient condition for soundness for normal forms.

1.4 Approximability

▶ **Definition 9** (Approximation). *Let P_0 and P be two $\mathtt{S}$-derivations typing the same term t. Then P_0 is an **approximation** of P if $\mathtt{bisupp}(P_0) \subseteq \mathtt{bisupp}(P)$ and, for all $\mathtt{p} \in \mathtt{bisupp}(P_0)$, $P_0(\mathtt{p}) = P(\mathtt{p})$. When this holds, we write $P_0 \leqslant P$.*

Intuitively, $P_0 \leqslant P$ if the derivation P_0 can be obtained from the derivation P by erasing some symbols inside P. For instance, let:

$$P^0_{\mathtt{ex}} = \dfrac{\dfrac{\dfrac{}{x : (2\cdot(\,) \to o') \vdash x : (\,) \to o'}\,\mathtt{ax}}{x : (2 \cdot (\,) \to o') \vdash x\, f^\omega : o}\,\mathtt{app}}{x : (2 \cdot (\,) \to o') \vdash \lambda f.x\, f^\omega : (\,) \to o'}\,\mathtt{abs}$$

Then $P^0_{\mathtt{ex}} \leqslant P_{\mathtt{ex}}$, since $P^0_{\mathtt{ex}}$ has been obtained from $P_{\mathtt{ex}}$ by erasing all typing information on f^ω. Indeed, we check that $\mathtt{supp}(P^0_{\mathtt{ex}}) \subseteq \mathtt{supp}(P_{\mathtt{ex}})$, $\mathtt{bisupp}(P^0_{\mathtt{ex}}) \subseteq \mathtt{bisupp}(P_{\mathtt{ex}})$ and $P^0_{\mathtt{ex}}(\mathtt{p}) = P_{\mathtt{ex}}(\mathtt{p})$ for all $\mathtt{p} \in \mathtt{bisupp}(P^0_{\mathtt{ex}})$.

The relation $\leqslant$ is an order. There are $\mathtt{S}$-derivations P that do not have finite approximations, e.g., any derivation typing Ω (see [17] for an example), but these derivation are *unsound*: they do not ensure any form of finitary or infinitary normalization. In contrast, a *finite* derivation is sound.

To retrieve validity, we must specify that infinitary derivations should be obtained as asymptotic extensions of *finite* derivations:

▶ **Definition 10** (Approximability). *Let P be a $\mathtt{S}$-derivation. Then P is **approximable** if P is the supremum of its finite approximations i.e., if, for all finite sets $B \subseteq \mathtt{bisupp}(P)$, there is a finite derivation P_0 such that $P_0 \leqslant P$ and $B \subseteq \mathtt{bisupp}(P_0)$.*

A term that is in the conclusion of an approximable derivation is said to be **approximably typable**. Quantitativity is of course a necessary condition for approximability, and types of infinite arity are unsound:

▶ **Lemma 11.** *If P is approximable, then P is quantitative and contains only 001-types.*

1.5 Soundness and completeness for system $\mathtt{S}$

The main characterization theorem of system $\mathtt{S}$ states the equivalence between infinitary weak normalization and typability: more precisely, t is WN_∞ iff there is an unforgetful and approximable $\mathtt{S}$-derivation typing P. This characterization is proved by the propositions below, that we will also use in this article. One recognizes usual properties that are expected from an intersection type system (subject reduction, expansion, typing of the normal forms), except that they pertain to infinitary objects and computations.

▶ **Proposition 12** (Infinitary subject reduction). *If $P \rhd C \vdash t : T$ is approximable and $t \to_\beta^\infty t'$, then there is an approximable derivation $P' \rhd C \vdash t' : T$.*

If a term is approximably typable, then, in particular, it is *finitely* typable, so that it is HN, as for usual, inductive intersection type systems:

▶ **Lemma 13** (Approximability and Head Normalization). *If $P \rhd C \vdash t : T$ is approximable, then t is head normalizing.*

Approximable $\mathtt{S}$-derivations ensure only *head* normalization because of the empty sequence $(\,)$, which allows us to leave an argument untyped. For instance, if x is assigned $(\,) \to o$, then $x\,t$ is typed with o for any term t. To ensure WN_∞, one needs to control the occurrences of $(\,)$: by definition, the empty sequence type $(\,)$ occur negatively in $(\,) \to T$ (base case), $(\,)$ occurs negatively (resp. positively) in $(S_k)_{k\in K} \to T$ if it occurs negatively (resp. positively) in T or positively (resp. negatively) in one of the S_k (inductive case).

▶ **Definition 14** (Unforgetfulness). *Let $P \rhd C \vdash t : T$ be a derivation. Then P is unforgetful when $(\,)$ does not occur negatively in C and does not occur positively in T.*

In particular, when $(\,)$ does not occur in C nor in T, then P is unforgetful.

▶ **Proposition 15** (Correctness for system $\mathtt{S}$). *If $P \rhd C \vdash t : T$ is approximable and unforgetful, then t is infinitary weakly normalizing.*

Completeness for infinitary normal forms – i.e., the fact that they are approximably and unforgetfully typable – is proved in two steps: one shows that every *quantitative* derivation typing a normal form is approximable (this is not true for non-normal forms). Since one finds quantitative unforgetful derivations for each normal form, one concludes:

▶ **Lemma 16** (Completeness for Normal Forms). *Let t be a NF_∞.*

- *If $P \rhd C \vdash t : T$ and P is quantitative, then P is approximable.*
- *t is approximably typable by derivation P such that $\mathtt{supp}(P) = \mathtt{supp}(t)$.*

Subject expansion holds for productive reduction paths:

▶ **Proposition 17** (Infinitary subject expansion). *If $t \to_\beta^\infty t'$ and $P' \rhd C \vdash t' : T$ is approximable, then there is an approximable derivation concluding with $C \vdash t : T$.*

From Lemma 16 and Proposition 17, one concludes that every infinitary weakly normalizing term is approximably and unforgetfully typable.

Last, Propositions 12 and 17 entail:

▶ **Lemma 18.** *If t is WN_∞, then t and $\mathtt{BT}(t)$ have the same approximable typings.*

2 Characterizing hereditary permutators

We now want to define the *permutator pairs* (S, T) (with S, T types of system $\mathtt{S}$) so that the judgments of the form $x : (2 \cdot S) \vdash t : T$ characterize the x-HP (i.e., there is an approximable $P \rhd x : (2 \cdot S) \vdash t : T$ iff t is an x-HP). Informally, if $h = \lambda x_1 \dots x_n . x\, h_{\sigma(1)} \dots h_{\sigma(n)}$ and h is typed with a type of arity n and $x_1, \dots, x_n$ are the respective head variables of $h_1, \dots, h_n$, then we have:

- Type of h = (type of x_1) $\to \dots \to$ (type of x_n) $\to o$ $\qquad (\mathtt{eq}_1)$
- Type of x = (type of $h_{\sigma(1)}$) $\to \dots \to$ (type of $h_{\sigma(n)}$) $\to o$ $\qquad (\mathtt{eq}_2)$

Since $x_1, \dots, x_n$ are the respective head variables of the headed hereditary permutators $h_1, \dots, h_n$, the equations $(\mathtt{eq}_1)$ and $(\mathtt{eq}_2)$, which are the golden thread of the proofs to come in the remainder of the paper, suggest the following coinductive definition:

▶ **Definition 19** (Permutators pairs).

- *When o ranges over $\mathcal{O}$ (the set of type atoms), the set $\mathtt{PP}(o)$ of **o-permutator pairs** (S, T), where S and T are $\mathtt{S}$-types, is defined by mutual coinduction:*

$$\frac{(S_1, T_1) \in \mathtt{PP}(o_1), \dots, \ (S_n, T_n) \in \mathtt{PP}(o_n) \qquad o_1, \dots, o_n, o \text{ pairwise distinct} \qquad \sigma \in \mathfrak{S}_n}{((2 \cdot T_{\sigma(1)}) \to \dots \to (2 \cdot T_{\sigma(n)}) \to o, (2 \cdot S_1) \to \dots (2 \cdot S_n) \to o) \in \mathtt{PP}(o)}$$

- *A pair $(S, T) \in \mathtt{PP}(o)$ is said to be **proper**, if, for all $o' \in \mathcal{O}$, o' occurs at most once in S and in T. The set of proper o-permutator pairs is denoted $\mathtt{PPP}(o)$.*

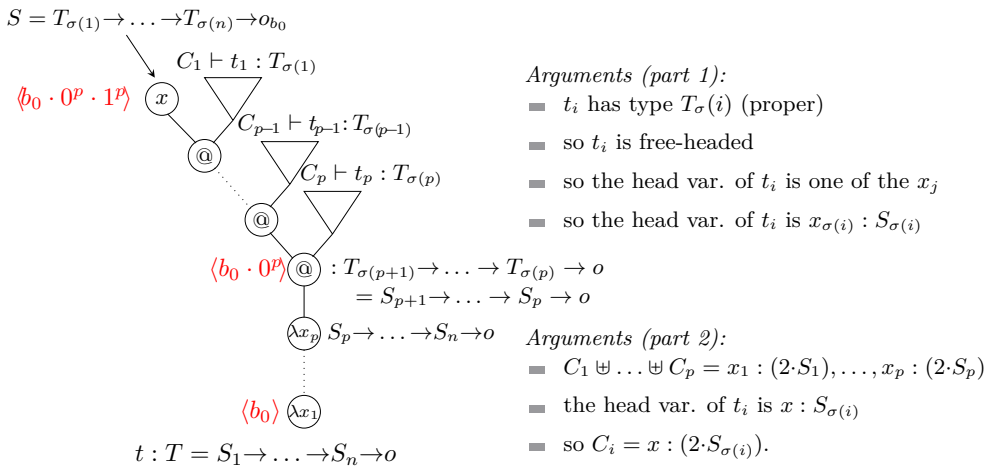

Figure 1 Hereditary permutators and permutator pairs.

Actually, we could allow other tracks than 2 in the definition (e.g., $T = (\ell_1 \cdot S_1) \to \ldots \to (\ell_n \cdot S_n) \to o$ would be fine), but it is more convenient to consider this restriction, so that we are relieved of the care of specifying the values of $\ell_1, \ldots, \ell_n$.

The condition of properness is here to ensure that every term variable occurs at exactly one level deeper than its binder and to distinguish them from one another: it is a key point of the proof of Claim 21, because two distinct variables will have types with distinct targets.

The first implication of the characterization is quite natural to prove:

▷ Claim 20 (From hereditary permutators to permutator pairs). Let $y \in \mathcal{V}$ and t be a y-head hereditary permutator. Then there is an approximable $\mathtt{S}$-derivation P and a permutator pair (S, T) such that $P \rhd y : (2 \cdot S) \vdash h : T$.

Proof. We skip the proof of this property. Observe that Definition 19 is designed so that it holds. The converse claim (Claim 21) is more difficult to prove and requires to be carefully verified. ◁

▷ Claim 21 (From permutator pairs to hereditary permutators). Let $t \in \Lambda^{001}$ be a 001-normal form and (S, T) a permutator pair and P a quantitative $\mathtt{S}$-derivation typing t.

- If $P \rhd \vdash t : (2 \cdot S) \to T$, then t is a hereditary permutator.
- If $P \rhd x : (2 \cdot S) \vdash T$, then t is a x-headed hereditary permutator.

In both cases, $\mathtt{supp}(P) = \mathtt{supp}(t)$.

Proof. The proof uses the following observation: let us say that a HNF is **free-headed** when its head variable is free. If (S, T) is a *proper* permutator pair and $t = \lambda x_1 \ldots x_p . x_i \, t_1 \ldots t_q$ (with $1 \leqslant i \leqslant p$) is a HNF which is not free headed, then t cannot have the types S and T, since the target of the type of x_i appears twice in the type of t.

We now start the proof, whose main stages are summarized in Fig. 1, in which we abusively write S instead of $(2 \cdot S)$. Assume $S = (2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o$ and $T = (2 \cdot S_1) \to \ldots \to (2 \cdot S_n) \to o$. We first prove that the first point of the claim reduces to the second one.

Since the context in $\vdash t : (2 \cdot S) \to T$ is empty, the head variable of t is bound and the arity of t is $\geqslant 1$. Thus, $t = \lambda x_0 . \lambda x_1 \ldots x_p . x \, t_1 \ldots t_q$ with $t_1, \ldots, t_q$ normal forms whose respective head variables are denoted $y_1, \ldots, y_q$. Note that:

- x is $x_1, \dots, x_p$ or x_0 since x is bound.
- The type assigned to x_0 is S. The respective types assigned to $x_1, \dots x_p$ are $S_1, \dots, S_p$.
- The common target type of T and the type of x_0 is o.

Since (S, T) is proper, o does not occur in $S_1, \dots, S_n$, so necessarily, $x = x_0$ and $x : (2 \cdot S) \vdash \lambda x_1 \dots x_p . x\, t_1 \dots t_q : T$ is derivable by means of a quantitative derivation P_*. Thus, we are now in the second case. The type of $x\, t_1 \dots t_q$ is both $T_{\sigma(q+1)} \to \dots \to T_{\sigma(n)} \to o$ since $x : S$ and $S_{p+1} \to \dots \to S_n \to o$ since $t : T$, so $p = q$ and for $p + 1 \leqslant i \leqslant n$, $\sigma(i) = i$ and $S_i = T_i$. Let us denote $o_1, \dots, o_n$ the respective target types of $S_1, \dots, S_n$. Since the type of x is S, the respective types of $t_1, \dots, t_p$ must be $T_{\sigma(1)}, \dots, T_{\sigma(p)}$. Moreover, since the "tail" of T is made of singleton sequence types $(2 \cdot S_i)$, $t_1, \dots, t_p$ are typed once in P and the head variables $y_1, \dots, y_p$ of $t_1, \dots t_p$ are also typed exactly once. In particular, P_* has a subderivation at depth n of the form:

$$\dfrac{\dfrac{\dfrac{}{x : (2 \cdot S) \vdash x : S}\,\mathtt{ax} \qquad P_1 \rhd C_1 \vdash t_1 : T_{\sigma(1)}\,[2]}{x : (2 \cdot S), C_1 \vdash x\, t_1 : (2 \cdot S_{\sigma(2)}) \to \dots \to o}\,\mathtt{app} \qquad \dots}{\dots}\,\mathtt{app} \qquad \dfrac{\dots \qquad\qquad P_p \rhd C_p \vdash t_p : T_{\sigma(p)}\,[2]}{x : (2 \cdot S), C_1 \uplus \dots \uplus C_p \vdash x\, t_1 \dots t_p : T'}\,\mathtt{app}$$

where $T' = T_{\sigma(p+1)} \to \dots \to T_{\sigma(n)} \to o = S_{p+1} \to \dots \to S_n \to o$.

Let us prove now that, for all $1 \leqslant i \leqslant p$, the unique argument derivation of x in P typing t_i, that we denote P_i, concludes with $x_{\sigma(i)} : (2 \cdot S_{\sigma(i)}) \vdash t_i : T_{\sigma(i)}$.

First, since t_i is normal, $t_i = \lambda z_1 \dots z_{p'} . y_i\, u_1 \dots u'_q$. Since $t_i : T_{\sigma(i)}$, t_i is free-headed by the observation above. Moreover, the head variable of t_i is typed once in P_* since t_i is typed once. Thus, y_i is one of the $x_1, \dots, x_p$. The only possibility is $y_i = x_{\sigma(i)}$ since the types of $x_1, \dots, x_p$ have pairwise distinct targets.

Since P is quantitative and $(2 \cdot S_i)$ is a *singleton* sequence type, $x_1, \dots,\ x_p$ must be exactly typed once in P_*, the subderivation of P typing $x\, t_1 \dots t_p$. This entails that the $\mathtt{ax}$-rule typing $x_{\sigma(i)}$ as the head variable of t_i concludes with $x_{\sigma(i)} : (2 \cdot S_{\sigma(i)}) \vdash x_{\sigma(i)} : S_{\sigma(i)}$. Thus, P_i concludes with a judgment of the form $x_{\sigma(i)} : (2 \cdot S_{\sigma(i)}) \uplus C'_i \vdash t_i : T_{\sigma(i)}$ (2nd argument).

Since $\uplus_{1 \leqslant i \leqslant p} (x_{\sigma(i)} : (2 \cdot S_{\sigma(i)}) \uplus C'_i) = x_1 : (2 \cdot S_1), \dots, x_p : (2 \cdot S_p)$, we deduce that C'_i is empty for all $1 \leqslant i \leqslant p$ (3rd argument). Thus, P_i concludes with $x_{\sigma(i)} : (2 \cdot S_{\sigma(i)}) \vdash t_i : T_{\sigma(i)}$.

This easily implies that $x_1 : (2 \cdot S_1) \vdash t_{\sigma^{-1}(1)} : T_1, \dots x_p : (2 \cdot S_p) \vdash t_{\sigma^{-1}(p)} : T_p$ are judgments of P_*. In particular, they are approximably derivable. Thus, $t_1, \dots, t_p$ also satisfy the hypothesis of point 2 of the claim. Since $t = \lambda x_1 \dots x_p . x\, t_1 \dots t_p$, we conclude using Definition 1. ◁

The two claims, which are valid for 001-normal forms, along with infinitary subject reduction and expansion, give a type-theoretical characterization of hereditary permutators in system $\mathtt{S}$:

▶ **Theorem 22.** *Let $t \in \Lambda^{001}$. Then t is a hereditary permutator iff $\vdash t : (2 \cdot S) \to T$ is approximably derivable for some proper permutator pair (S, T).*

Proof.

- The implication $\Leftarrow$ is given by Claim 21 and Proposition 17.
- Implication $\Rightarrow$: let t be a hereditary permutator. By Definition 1, its Böhm tree is of the form $\lambda x.h$ where h is a normal x-headed hereditary permutator. By Claim 20, there is a proper permutator pair (S, T) and an approximable derivation P such that $P \rhd x : (2 \cdot S) \vdash h : T$. By Proposition 17, $\vdash t : (2 \cdot S) \to T$ is also approximably derivable. ◀

3 A unique type to rule them all

In this section, we explain how to enrich system $\mathtt{S}$ with type constants and typing rules so that there is one type characterizing the set of hereditary permutators, as expected.

In Section 2, we proved that a term t is a hereditary permutator iff it can be assigned a type of the form $(2{\cdot}S) \to T$ where (S,T) is a proper permutator pair. To obtain a unique type for all the hereditary permutators, one idea is to identify all the types of the form $(2{\cdot}S) \to T$, where (S,T) ranges over $\mathtt{PPP}$ with a type constant $\mathtt{ptyp}$. However, since quotienting types may bring unsoundness (e.g., if o and $o \to o$ are identified), one must then verify that the correctness and the completeness of system $\mathtt{S}$ is preserved, and that the approximability criterion can be suitably extended. The main argument, given by Lemma 26, is that the notions of hereditary permutators and permutators pairs, which are infinitary, have arbitrarily big finite approximations, which are defined as trucations at some applicative depth d. Thus, we may express hereditary permutators and permutator pairs as asymptotic limits and adapt the general methods of system $\mathtt{S}$.

Our approach parallels that of Tatsuta [14], which uses a family of constants $\mathtt{ptyp}_d$, with a few differences: in the finite restriction of our system, it is easier to deal with hereditary permutators (normalization is simple to prove in finite non-idempotent type systems), but of course we have to treat the infinitary typings and we consider the constant $\mathtt{ptyp}$, which is subsumed under all the $\mathtt{ptyp}_d$, which represent hereditary permutators under level d.

3.1 Permutator schemes

Before presenting the system giving a unique type to all hereditary permutators, we must first explain how the typings of hereditary permutators are approximated in system $\mathtt{S}$.

▶ **Definition 23** (Permutator schemes). *Let $d \geqslant 0$. A term t is a x-headed (resp. closed) permutator scheme of degree d if its Böhm tree is equal to that of a hereditary permutator on $\{b \in \{0,1,2\}^* \mid \mathtt{ad}(b) \leqslant d\}$. The set of x-headed (resp. closed) permutators schemes of degree d is denoted $\mathtt{PS}_d(x)$ (resp. $\mathtt{PS}_d$).*

The sequence $(\mathtt{PS}_d)$ is decreasing, i.e., $\mathtt{PS}_d \supseteq \mathtt{PS}_{d+1}$, and $\mathtt{HP} = \cap_{d \geqslant 0} \mathtt{PS}_d$.

▶ **Definition 24** (Permutator pairs of degree d). *Let $d \in \mathbb{N}$.*

- *When o ranges over $\mathcal{O}$, the set $\mathtt{PP}_d(o)$ of **o-permutator pairs of degree d** (S,T), where S and T are $\mathtt{S}$-types, is defined by induction on d:*

$$\frac{}{(\underbrace{(\,) \to \ldots (\,)}_{n} \to o, \underbrace{(\,) \to \ldots \to (\,)}_{n} \to o) \in \mathtt{PP}_0(o)}$$

$$\frac{(S_1,T_1) \in \mathtt{PP}_{d-1}(o_1), \ldots,\ (S_n,T_n) \in \mathtt{PP}_{d-1}(o_n) \quad o_1,\ldots,o_n,o \text{ pairwise distinct} \quad \sigma \in \mathfrak{S}_n}{((2\cdot T_{\sigma(1)}) \to \ldots \to (2\cdot T_{\sigma(n)}) \to o, (2\cdot S_1) \to \ldots (2\cdot S_n) \to o) \in \mathtt{PP}_d(o)}$$

- *A pair $(S,T) \in \mathtt{PP}_d(o)$ is said to be **proper** if every type variable occurs at most once in S and T. The set of proper permutator pairs of degree d is denoted $\mathtt{PPP}_d$.*

We can also see permutator pairs of degree d as truncation of permutator pairs: let U be a $\mathtt{S}$-type or a sequence type and $d \in \mathbb{N}$. We denote by $(U)^{\leqslant d}$ the truncation of T at depth d i.e., $\mathtt{supp}((U)^{\leqslant d}) = \mathtt{supp}(U) \cap \{c \in \mathbb{N}^* \,|\, \mathtt{ad}(c) \leqslant d\}$ and $(U)^{\leqslant d}(c) = U(c)$ for all $c \in \mathtt{supp}((U)^{\leqslant d})$. It is easy to check that $(U)^{\leqslant d}$ is a correct type or sequence type. We extend the notation to $\mathtt{S}$-contexts. Note that, if $d \geqslant 1$, $((S_k)_{k\in K} \to T)^{\leqslant d} = ((S_k)^{\leqslant d-1})_{k\in K} \to (T)^{\leqslant d}$

and $d = 1$, then $((S_k)_{k\in K} \to T)^{\leqslant d} = (\,) \to (T)^{\leqslant 1}$. By induction on d, this entails that, if $(S,T) \in \mathtt{PPP}$, then $((S)^{\leqslant d}, (T)^{\leqslant d}) \in \mathtt{PPP}_d$. Indeed, the base case ($d = 0$) is obvious and if $d \geqslant 1$, $T = (2 \cdot S_1) \to \ldots \to (2 \cdot S_n) \to o$ and $S = (2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o$ with $\sigma \in \mathfrak{S}_n$, $(S_i, T_i) \in \mathtt{PPP}$ for $1 \leqslant i \leqslant n$, then:

- $(T)^{\leqslant d} = (2 \cdot (S_1)^{\leqslant d-1}) \to \ldots \to (2 \cdot (S_1)^{\leqslant d-1}) \to o$ $(\mathtt{eq}_3)$
- $(S)^{\leqslant d} = (2 \cdot (T_{\sigma(1)})^{\leqslant d-1}) \to \ldots \to (2 \cdot (T_{\sigma(n)})^{\leqslant d-1}) \to o$ $(\mathtt{eq}_4)$

so that, by Definition 24, $((S)^{\leqslant d}, (T)^{\leqslant d}) \in \mathtt{PPP}_d(o)$.

▶ **Proposition 25** (Characterizing permutation schemes)**.** *Let $d \geqslant 1$ and t be a 001-term. Then $t \in \mathtt{PS}_d$ iff $\vdash t : (2 \cdot S) \to T$ is approximably derivable for some $(S,T) \in \mathtt{PPP}_d$.*

Proof.

$\Rightarrow$ Straightforward induction on the structure of t.

$\Leftarrow$ The proof is the same as Claim 21, we also obtain that $x_i : (2 \cdot S_i) \vdash t_{\sigma^{-1}(i)} : T_i$ are judgments of P, except that $(S_i, T_i) \in \mathtt{PPP}_{d-1}$ instead of $(S_i, T_i) \in \mathtt{PPP}$. ◀

It is not enough to know that a x-headed hereditary permutator t is approximably typable in a judgment $x : (2 \cdot S) \vdash t : T$ with $(S,T) \in \mathtt{PPP}$, which implies that T is the supremum of a direct family of finite types which be assigned to t: in order to prove soundness regarding quotienting, we must prove that this typing is the supremum of typings ensuring that t is a permutator scheme of degree d, i.e., by Proposition 25, one must type t with $(S_d, T_d) \in \mathtt{PPP}_d$ for all d. The lemma below is the missing third ingredient (along with Claims 20 and 21) of this article and will allow us to define in Section 3.2 an extension of system $\mathtt{S}$ giving a unique type to hereditary permutators:

▶ **Lemma 26** (Approximations and permutator pairs)**.** *If $P \rhd x : (2 \cdot S) \vdash t : P$ is approximable, where $(S,T) \in \mathtt{PPP}$, then, for all $d \in \mathbb{N}$, there is a finite $P_d \leqslant P$ such that $P_d \rhd x : (2 \cdot S_d) \vdash t : T_d$ with $(S_d, T_d) \in \mathtt{PPP}_d$.*

Proof. Since $(\,)$ does not occur in S and T, by Lemma 18, we can assume that t is a 001-normal form without loss of generality. We then reason by induction on d.

Let us present the argument informally. Say that $t = \lambda x_1 \ldots x_p.x\, t_1 \ldots t_p$, $S = T_{\sigma(1)} \to \ldots \to T_{\sigma(n)} \to o$ and $T = S_1 \to \ldots \to S_n \to o$. Intuitively, $t : T$ with $x : S$ and for $1 \leqslant i \leqslant p$, $t_i : T_{\sigma(i)}$ is a hereditary permutator headed by $x_{\sigma(i)} : S_{\sigma(i)}$, as specified by Fig. 1. When we truncate the type of t at applicative depth d, we have now $t : (T)^{\leqslant d}$ with $x : (S)^{\leqslant d}$. But, by $(\mathtt{eq}_3)$ and $(\mathtt{eq}_4)$, we must truncate the types of $t_1, \ldots, t_p$ and $x_1, \ldots, x_p$ at applicative depth $d-1$. Inductively, this demands that we truncate the types of the arguments of $t_1, \ldots, t_p$ at applicative depth $d-2$. By proceeding so, we obtain a finite derivation $P_d \leqslant P$ concluding with $x : (2 \cdot (S)^{\leqslant d}) \vdash t : (T)^{\leqslant d}$. ◀

3.2 System $\mathtt{S}_{\mathtt{hp}}$

Let $\mathtt{ptyp}$ and $\mathtt{ptyp}_d$ ($d \in \mathbb{N}$) be a family of type constants. The set of $\mathtt{S}_{\mathtt{hp}}$-types is defined by:

$$T,\, S_k \;::=\; o \parallel \mathtt{ptyp}_d \parallel \mathtt{ptyp} \parallel (S_k)_{k\in K} \to T$$

System $\mathtt{S}_{\mathtt{hp}}$ has the same typing rules as system $\mathtt{S}$ with the addition of:

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}_d}{\vdash \lambda x.t : \mathtt{ptyp}_d}\,\mathtt{hp}_d \qquad\qquad \frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}}{\vdash \lambda x.t : \mathtt{ptyp}}\,\mathtt{hp}$$

Thus, rule $\mathtt{hp}_d$ allows assigning the constant $\mathtt{ptyp}_d$ to any normal permutator scheme of degree d and rule $\mathtt{hp}$ assign the constant $\mathtt{ptyp}$ to any normal hereditary permutator by Claims 20 and 21. Intuitively, $\mathtt{ptyp} = \mathtt{ptyp}_\infty$ and we will make this idea more precise with Definition 27. Note also that if $t : \mathtt{ptyp}_d$ or $t : \mathtt{ptyp}$, t cannot be applied to an argument u, even if t is an abstraction: the rules $\mathtt{hp}_d/\mathtt{hp}$ freeze the terms.

The notions of support, bisupport, permutator pairs *etc* naturally extend to $\mathtt{S}_{\mathtt{hp}}$. We define an order $\leqslant$ on $\mathcal{O} \cup \{\to, \mathtt{ptyp}\} \cup \{\mathtt{ptyp}_d \,|\, d \in \mathbb{N}\}$ by $o \leqslant o$, $\to \leqslant \to$, $\mathtt{ptyp}_d \leqslant \mathtt{ptyp}$ and $\mathtt{ptyp}_d \leqslant \mathtt{ptyp}_{d'}$ for $d \leqslant d'$.

▶ **Definition 27** (Approximation and Approximability in system $\mathtt{S}_{\mathtt{h}}$).

- *Let P_0 and P be two $\mathtt{S}_{\mathtt{hp}}$-derivations. We write $P_0 \leqslant P$ (P_0 is an approximation of P) if $\mathtt{bisupp}(P_0) \subseteq \mathtt{bisupp}(P)$ and, for all $\mathtt{p} \in \mathtt{bisupp}(P_0)$, $P_0(\mathtt{p}) \leqslant P(\mathtt{p})$.*
- *Let P be a $\mathtt{S}_{\mathtt{hp}}$-derivation. Then P is **approximable** if P is the supremum of its finite approximations.*

This extends Definition 10: for all $\mathtt{S}$-derivations P, P is approximable for system $\mathtt{S}$ iff it is approximable for system $\mathtt{S}_{\mathtt{hp}}$. We first notice that rules $\mathtt{hp}_{(d)}$ are invertible for HNF:

▶ **Lemma 28** (Inverting rules $(\mathtt{hp}_d)$ for head normal forms). *Let t be a HNF. If $\vdash t : \mathtt{ptyp}_d$ (resp. $P\triangleright \vdash t : \mathtt{ptyp}$) is approximably derivable, then $t = \lambda x.t_0$ with $x : (2 \cdot S) \vdash t_0 : T$ approximably derivable, for some $(S, T) \in \mathtt{PPP}_d$ (resp. $(S, T) \in \mathtt{PPP}$).*

Proof. We consider the case $\mathtt{ptyp}$ (the case $\mathtt{ptyp}_d$ is similar), i.e., we assume that $P' \triangleright \vdash t : \mathtt{ptyp}$ is approximable. For one, $t = x\, t_1 \dots x_n$ is impossible, because we would have $C(x) \neq (\,)$ since the head variable x is free in $x\, t_1 \dots t_n$. So, t is an abstraction, i.e., $t = \lambda x.t_0$ and thus, the last rule of P is either $\mathtt{abs}$, $\mathtt{hp}_d$, $\mathtt{hp}$. But it is neither $\mathtt{abs}$ (we would have an arrow type) nor $\mathtt{hp}_d$, so it is $\mathtt{hp}$ and thus, P' is of the form:

$$P' = \frac{P \triangleright x : (2 \cdot S) \vdash t_0 : T \qquad (S, T) \in \mathtt{PPP}}{\vdash t : \mathtt{ptyp}}\,\mathtt{hp}$$

Since P' is approximable, P also is. ◀

All is now in place to obtain the expected properties of system $\mathtt{S}_{\mathtt{hp}}$:

▶ **Lemma 29** (Characterizing normal hereditary permutators). *Let t be a 001-normal form.*

- *$t \in \mathtt{PS}_d$ iff $\vdash t : \mathtt{ptyp}_d$ is approximably derivable.*
- *$t \in \mathtt{HP}$ iff $\vdash t : \mathtt{ptyp}$ is approximably derivable.*

Proof. The two points are handled similarly. We do not prove the first one, which uses Proposition 25:

- If $t = \lambda x.h$ is a HP, then, by Claim 20, there is $(S, T) \in \mathtt{PPP}$ and P a $\mathtt{S}$-derivation such that $P \triangleright x : (2 \cdot S) \vdash h : T$. We then set:

  $$P' = \frac{P \triangleright h : (2 \cdot S) \vdash p : T}{\vdash t : \mathtt{ptyp}}\,\mathtt{hp}$$

  By Lemma 26, for all $d \in \mathbb{N}$, there is a finite $\mathtt{S}$-derivation $P_d \leqslant P$ such that $P_d \triangleright x : (2 \cdot S_d) \vdash h : T_d$ with $(S_d, T_d) \in \mathtt{PPP}_d$ and $P = \sup_d P_d$. We then set:

  $$P'_d = \frac{P_d \triangleright x : (2 \cdot S_d) \vdash h : T_d}{\vdash t : \mathtt{ptyp}_d}\,\mathtt{hp}_d$$

  By construction, $\sup_d P'_d = P'$.

- Conversely, assume that $P'\rhd \vdash t : \mathtt{ptyp}$ is approximable. By Lemma 28, P' concludes with the $\mathtt{hp}$-rule, so P' is of the form:

$$P' = \dfrac{P \rhd x : (2\cdot S) \vdash t_0 : T \qquad (S,T) \in \mathtt{PPP}}{\vdash t : \mathtt{ptyp}}\,\mathtt{hp}$$

  Let $d \in \mathbb{N}$. Since P' is the supremum of its finite approximations, there is a finite approximation $P'_0 \leqslant P'$ concluding with[1] $\vdash t : \mathtt{ptyp}$ or $\vdash t : \mathtt{ptyp}_{d'}$ with $d' \geqslant d$. Thus, $t \in \mathtt{PS}_{d'} \subseteq \mathtt{PS}_d$ or $t \in \mathtt{HP}$. This proves that $t \in \cap_{d\geqslant 0}\mathtt{PS}_d = \mathtt{HP}$. ◀

▶ **Lemma 30** (Soundness of system $\mathtt{S}_{\mathtt{hp}}$)**.** *If t is approximably typable in system $\mathtt{S}_{\mathtt{hp}}$, then t is head normalizing.*

Proof. If t is approximably typable, there is a *finite* $\mathtt{S}_{\mathtt{hp}}$-derivation $P \rhd C \vdash t : T$. If t is a HNF, we are done. In the other case, $t \to_{\mathtt{h}} t'$ for some t'. It is routine work in *non-idempotent* intersection type theory (see [4]) to prove a *weighted* subject reduction property, i.e., that there is $P' \rhd C \vdash t' : T$ such that $|\mathtt{supp}(P')| < |\mathtt{supp}(P)|$, i.e., P' contains strictly less judgments than P does. The only unusual rules are $\mathtt{hp}_d$ and $\mathtt{hp}$, which are easily handled.

Since $|\mathtt{supp}(P)| \in \mathbb{N}$ and $\mathbb{N}$ is well-founded, weighted subject reduction entails that head reduction terminates on t. ◀

▶ **Corollary 31.** *If $\vdash t : \mathtt{ptyp}$ is approximably derivable, then t is WN_∞.*

Proof. By Lemma 30, t reduces to a HNF t'. By subject reduction, $\vdash t' : \mathtt{ptyp}$ is also approximably derivable. Then, Lemma 28 entails that $t = \lambda x.t_0$ and $x : (2\cdot S) \vdash t_0 : T$ is approximably derivable in system $\mathtt{S}$ for some t_0 and permutation pair (S,T). Since this latter judgment is $(\,)$-free, Proposition 15 entails that t_0 is WN_∞. Thus, t also is WN_∞. ◀

More generally, the dynamic properties of system $\mathtt{S}$ are preserved in system $\mathtt{S}_{\mathtt{hp}}$.

▶ **Proposition 32** (Infinitary subject reduction)**.** *If $t \to_\beta^\infty t'$ and $P \rhd C \vdash t : T$ is an approximable $\mathtt{S}_{\mathtt{hp}}$-derivation, then there exists an approximable derivation $P' \rhd C \vdash t' : T$.*

▶ **Proposition 33** (Infinitary subject expansion)**.** *If $t \to_\beta^\infty t'$ and $P' \rhd C \vdash t' : T$ is an approximable $\mathtt{S}_{\mathtt{hp}}$-derivation, then there exists an approximable derivation $P \rhd C \vdash t : T$.*

Proof. The proofs of infinitary subject reduction and expansion in system $\mathtt{S}_{\mathtt{hp}}$ do not differ of those for system $\mathtt{S}$, which can be found in [17] (in particular, Section II.D. and VI.D.) or in Chapter 10 of [18], so we do not give the details. Once again, the only new rules are $\mathtt{hp}$ and $\mathtt{hp}_d$, which are easily handled in the one step case, then in the asymptotic case.

Infinitary subject reduction is easy to prove, but infinitary subject expansion holds because we can expand *finite* approximations of a derivation P' concluding a productive reduction path. Why? Because, if $t \to_\beta^\infty t'$ and P'_f is finite and types t', then there is a term $t \to_\beta^k t_k$ obtained from t after a finite number k of β-reduction steps such that, on $\mathtt{supp}(P')$, t_k and t' induce the same subtree (this is a consequence of Definition 5). Thus, we can *substitute* t' with t_k in P'_f: we then obtain P'^k_f typing t_k. After k expansion steps, we obtain from P'^k_f a derivation P_f typing t. To conclude this dense summary, let us just say that the infinitary subject expansion is not so about the rules than about the possibility to replace a derivation by its finite approximations, which is precisely what Definition 27 captures for system $\mathtt{S}_{\mathtt{hp}}$. ◀

[1] The case $P'_0 \rhd \vdash t : \mathtt{ptyp}$ is possible: there are finite HP and PPP, e.g., $\lambda x.x$ and (o,o).

We now give a positive answer to TLCA Problem # 20:

▶ **Theorem 34** (Characterizing hereditary permutators with a unique type)**.** *Let* $t \in \Lambda^{001}$*. Then* t *is a hereditary permutator iff* $\vdash t : \mathtt{ptyp}$ *is approximably derivable in system* $\mathtt{S}_{\mathtt{hp}}$*.*

Proof.

⇒ If t is a HP, let t' be its 001-NF. By Lemma 29, there is an approximable derivation $P'\triangleright \vdash t' : \mathtt{ptyp}$. By Proposition 33, there is $P\triangleright \vdash t' : \mathtt{ptyp}$ approximable.

⇐ Given by Corollary 31. ◀

Future work

We plan to adapt system S to characterize other sets of Böhm trees and other notions of infinitary normalization, including weak normalization in the calculi Λ^{101} and Λ^{111} of [11].

References

1 Patrick Bahr. Strict Ideal Completions of the Lambda Calculus. In *FSCD 2018, July 9-12, Oxford*, pages 8:1–8:16, 2018.

2 Henk Barendregt. *The Lambda-Calculus: Its Syntax and Sematics*. Ellis Horwood series in computers and their applications. Elsevier, 1985.

3 Jan A. Bergstra and Jan Willem Klop. Invertible Terms in the Lambda Calculus. *Theor. Comput. Sci.*, 11:19–37, 1980.

4 Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-Idempotent Intersection Types for the Lambda-Calculus. *Mathematical Structures in Computer Science.*, 2017.

5 Daniel De Carvalho. *Sémantique de la logique linéaire et temps de calcul*. PhD thesis, Université Aix-Marseille, November 2007.

6 Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 4:685–693, 1980.

7 Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume I. North-Holland Co., Amsterdam, 1958. (3rd edn. 1974).

8 Lukasz Czajka. A Coinductive Confluence Proof for Infinitary Lambda-Calculus. In *Rewriting and Typed Lambda Calculi - Joint International Conference, RTA-TLCA, Vienna, Austria, July 14-17*, pages 164–178, 2014.

9 Mariangiola Dezani-Ciancaglini. Characterization of Normal Forms Possessing Inverse in the *lambda-beta-eta*-Calculus. *Theor. Comput. Sci.*, 2(3):323–337, 1976.

10 Philippa Gardner. Discovering Needed Reductions Using Type Theory. In *TACS, Sendai*, 1994.

11 Richard Kennaway, Jan Willem Klop, M. Ronan Sleep, and Fer-Jan de Vries. Infinitary Lambda Calculus. *Theor. Comput. Sci.*, 175(1):93–125, 1997.

12 Betti Venneri Mario Coppo, Mariangiola Dezani-Ciancaglini. Functional Characters of Solvable Terms. *Mathematical Logic Quarterly*, 27:45–58, 1981.

13 Makoto Tatsuta. Types for Hereditary Head Normalizing Terms. In *FLOPS, Ise, Japan, April 14-16*, pages 195–209, 2008.

14 Makoto Tatsuta. Types for Hereditary Permutators. In *LICS, 24-27 June, Pittsburgh*, pages 83–92, 2008.

15 Steffen van Bakel. Complete Restrictions of the Intersection Type Discipline. *Theor. Comput. Sci.*, 102(1):135–163, 1992.

16 Steffen van Bakel. Intersection Type Assignment Systems. *Theor. Comput. Sci.*, 151(2):385–435, 1995.

17 Pierre Vial. Infinitary intersection types as sequences: a new answer to Klop's problem. In *LICS, Reykjavik*, 2017.

18 Pierre Vial. *Non-Idempotent Typing Operator, beyond the Lambda-Calculus*. Phd thesis, Université Sorbonne Paris-Cité, 2017, available on `http://www.irif.fr/~pvial`.