Infinitary Intersection Types as Sequences: a New Answer to Klop's Question

Pierre Vial

IRIF, Université Paris-Diderot pvial@pps.univ-paris-diderot.fr

Abstract

We provide a type-theoretical characterization of weakly-normalizing terms in an infinitary lambda-calculus. We adapt for this purpose the standard quantitative (with non-idempotent intersections) type assignment system of the lambda-calculus to our infinite calculus.

Our work provides a new answer to Klop's HHN-problem, namely, finding out if there is a type system characterizing the hereditary head-normalizing (HHN) lambda-terms. Tatsuta showed that HHN could not be characterized by a finite type system. We prove that an infinitary type system endowed with a validity condition called approximability can achieve it.

1 Introduction

The **head-normalizing (HN) terms** can be characterized by various *intersection* type systems. Recall that a term is HN if it can be reduced to a **head-normal form (HNF)**, *i.e.* a term t of the form $\lambda x_1 \ldots x_p.(x\, t_1) \ldots t_q$ ($p \geq 0, q \geq 0$), where x is referred as the **head-variable** of t and the terms $t_1, \ldots, t_q$ as the **arguments** of the head-variable x.

In general, intersection type frameworks, introduced by Coppo and Dezani [4], allow to characterize many classes of normalizing terms, such as the **weakly normalizing (WN) terms** (see [12] for an extensive survey). A term is WN if it can be reduced to a **normal form (NF)**, *i.e.* a term without *redexes*. *Inductively*, a term is WN if it is HN and all the arguments of its head-variable are WN (it is meant that the base cases of this induction are the terms whose HNF is $\lambda x_1 \ldots x_p.x$).

According to Tatsuta [11], the question of finding out a type system characterizing **hereditary head-normalizing (HHN) terms** was raised by Klop in a private exchange with Dezani in the late 90s. The definition of HHN term is given by the *coinductive* version of the above inductive definition: *coinductively*, a term is HHN if it is HN and all the arguments of its head variable are themselves HHN. It is equivalent to say that the Böhm tree of the term does not hold any occurrence of $\perp$. Tatsuta focused his study on *finitary* type systems and showed Klop's problem's answer was negative for them, by noticing that the set of HHN terms was not recursively enumerable.

Parallelly, the Böhm trees without $\perp$ can be seen as the set of normal forms of an infinitary calculus, referred as Λ^{001} in [7], which has been reformulated very elegantly in coinductive frameworks [6, 5]. In this calculus, the HHN terms correspond to the infinitary variant of the WN terms. An infinite term is WN if it can be reduced to a NF by at least one **strongly converging reduction sequence (s.c.r.s.)**, which constitute a special kind of reduction sequence of (possibly) infinite length, regarded as *sound*. This motivates to check whether an *infinitary* type system is able to characterize HHN terms in the infinite calculus Λ^{001}.

We use a quantitative, resource-aware type system to help us achieve this goal. In those type systems, typability is known to imply normalizability by a very simple (variant of the same) argument. Namely, reducing a typed redex inside a derivation decrease some non-negative integer measure, which entails that the reduction must stop at some point (see for instance [1] or Lemma 6). This is unlikely to be adapted in an infinitary framework.

However, quantitative type derivations do have very simple and readable combinatorial features that will turn out to be useful to build an infinitary type system. In particular, reduction inside a derivation almost comes down to *moving* parts of the original derivation, without adding new rules (a figure is given in § 4.1).

Contributions

We define an infinitary quantitative type system, inspired by the finitary de Carvalho's system $\mathcal{M}_0$ [3]. However, we show that a direct coinductive adaptation of system $\mathcal{M}_0$ cannot work for two reasons (*c.f.* Section 2):

- It would lead to the possibility of typing some non-HN terms, like $\Delta\Delta$. That is why a validity criterion is needed to discard irrelevant derivations, as in other infinitary frameworks [10].
- This validity criterion relies on the idea of **approximability**. It can be seen that multisets are not fit to formally express such a notion, which motivates the need for rigid constructions: multisets of types are replaced (coinductively) by families of types indexed by integers called **tracks**.

Tracks constitute the main feature of the type system presented here. They act as *identifiers* and allow us to bring out a combinatorics that existed implicitly – but could not be formulated – in regular quantitative type systems, where multiset constructions made it impossible to distinguish two copies of the same type. For instance, we will be able to trace any type through the rules of a whole typing derivation. Our framework is deterministic, *e.g.* there is a *unique* way to produce a derivation from another one while reducing a redex.

Outline

We informally discuss the necessity of the notion of approximability and rigid constructions in § 2. In § 3, we formally define our terms, type system and tracks. In § 4, we define reduction and expansion of a typing derivation, as well as *residuals*. In § 5, we formulate the approximability condition and the WN characterization criterion (called *unforgetfulness* here), and next, we prove an infinitary subject reduction property. In § 6, we describe all the sound derivations typing a normal form and prove an infinitary subject expansion property. It concludes the proof of our type-theoretic characterization of WN.

2 Informal Discussion

In this section, we **informally** introduce, through a few examples, the key concepts of our work, namely *rigidity* and *approximability*.

2.1 The Finitary Type System $\mathcal{M}_0$ and Unforgetfulness

Let us first recall the typing system $\mathcal{M}_0$ with *non-idempotent* intersection types [3], given by the following *inductive* grammar $\sigma,\ \tau ::= \alpha \mid [\sigma_i]_{i\in I} \to \tau$, where the constructor $[\,]$ is used for finite multisets, and the type variable α ranges over a countable set $\mathcal{X}$ of type variables. We write $[\sigma]_n$ to denote the multiset containing σ with multiplicity n. The multiset $[\sigma_i]_{i\in I}$ is meant to be the intersection of the types σ_i, taking into account their *multiplicity*. In idempotent intersection type systems, the type intersections $A \wedge B \wedge A$ and $A \wedge B$ are *de facto* equal, whereas in $\mathcal{M}_0$, the multiset types $[\sigma,\ \tau,\ \sigma]$ and $[\sigma,\ \tau]$ are not. No weakening is allowed either, *e.g.* $\lambda x.x$ can be typed with $[\tau] \to \tau$, but *not* with $[\tau, \sigma] \to \tau$.

In system $\mathcal{M}_0$, a *judgment* is a triple $\Gamma \vdash t : \sigma$, where Γ is a context, *i.e.* a function from the set $\mathcal{V}$ of term variables to the sets of multiset types $[\sigma_i]_{i \in I}$, t is a term and σ is a type. The multiset union $+$ is extended point-wise on contexts. Let us consider the rules below:

$$\frac{}{x : [\tau] \vdash x : \tau}\ \text{ax} \qquad \frac{\Gamma, x : [\sigma_i]_{i\in I} \vdash t : \tau}{\Gamma \vdash \lambda x.t : [\sigma_i]_{i \in I} \rightarrow \tau}\ \text{abs}$$

$$\frac{\Gamma \vdash t : [\sigma_i]_{i\in I} \rightarrow \tau \qquad (\Delta_i \vdash u : \sigma_i)^{i \in I}}{\Gamma + \sum\limits_{i \in I} \Delta_i \vdash t(u) : \tau}\ \text{app} \qquad \frac{\Pi_k \qquad (\Pi_k)_{i\in I}}{\Delta_i \vdash t(u)}\ \text{app}$$

The set of derivations is defined *inductively* by the above rules. We write $\Pi \rhd \Gamma \vdash t : \tau$ to mean that the (finite) derivation Π concludes with the judgment $\Gamma \vdash t : \tau$. A term is HN iff it is typable in system $\mathcal{M}_0$.

▶ **Remark.** The rule `app` relies on the equality between two multisets: the multisets of the types typing u and the negative part of the arrow type typing t must be equal to grant that tu is typable. In constrast to equality between two sequences, the multiset equality $[\sigma_i]_{i\in I} = [\sigma'_i]_{i \in I'}$ can be seen as *commutative* since the order we use to list the elements of the involved m.s. is of no matter (it is intuitively *collapsed* for m.s.).

Notice that if x is assigned $[\,] \rightarrow \tau$, then $x\,t$ is typable with type τ for any term t – which is left untyped – even if t is not HN. In order to characterize WN, we must grant somehow that every subterm (that cannot be erased during a reduction sequence) is typed : $[\,]$ should not occur at bad positions in a derivation Π. Actually, it is enough to only look at the judgment concluding Π : a term t is WN iff it is typable in $\mathcal{M}_0$ inside an **unforgetful** judgment. We say here that judgment $\Gamma \vdash t : \tau$ is unforgetful when Γ (resp. τ) does not hold negative (resp. positive) occurrences of $[\,]$. The proper definitions are to be found in § 5.2, but, for now, it is enough for now to notice that a sufficient condition of unforgetulness is to be $[\,]$**-free**: t is WN as soon as Γ and τ do not hold $[\,]$.

2.2 Infinitary Subject Expansion by Means of Truncation

Let us just admit that there is an infinitary version of $\mathcal{M}_0$, that we call $\mathcal{M}$. System $\mathcal{M}$ allows infinite multiset (*e.g.* $[\alpha]_\omega$ is the multiset in which α occurs with an infinite multiplicty, s.t. $[\alpha]_\omega = [\alpha] + [\alpha]_\omega$) and proofs of infinite depth.

Let $\Delta_f = \lambda x.f(xx)$ and $\mathtt{Y} = \Delta_f \Delta_f$. Notice $\mathtt{Y} \rightarrow f(\mathtt{Y})$, so $\mathtt{Y} \rightarrow^k f^k(\mathtt{Y})$. Intuitively, if $k \rightarrow \infty$, the redex disapper and we get $\mathtt{Y} \rightarrow^\infty f^\omega$ where f^ω is the (infinite) term $f(f(f(\ldots)))$, satisfying $f^\omega = f(f^\omega)$ and containing a rightward infinite branch. Since f^ω does not hold any redex, f^ω can be seen as the NF of $\mathtt{Y}$.

In order to adapt the previous criterion, the idea is to type NF (here, f^ω) in unforgetful judgment, and then proceed by (possibly infinite) expansion to get a typing derivation of the expanded term (here, $\mathtt{Y}$). Let us consider the following $\mathcal{M}$-derivation Π' (presented as fixpoint):

$$\Pi' = \frac{\dfrac{}{f : [[\alpha] \rightarrow \alpha] \vdash f : [\alpha] \rightarrow \alpha}\ \text{ax} \qquad \dfrac{\Pi'}{f : [[\alpha] \rightarrow \alpha]_\omega \vdash f^\omega : \alpha}}{f : [[\alpha] \rightarrow \alpha]_\omega \vdash f^\omega : \alpha}\ \text{app}$$

Now, Π' yields an unforgetful typing of f^ω (no occurrence of $[\,]$). This the kind of derivation we want to expand in order to get a derivation Π typing $\mathtt{Y}$. Since $\mathtt{Y} \rightarrow^\infty f^\omega$ (infinite number of reduction steps), we are stuck. But notice that Π' can be truncated into the derivation Π'_n below for any $n \geqslant 1$ (we write Γ_n for $f : [[\alpha] \rightarrow \alpha]_{n-1} + [[\,] \rightarrow \alpha]$):

$$\dfrac{\dfrac{}{f : [[\alpha] \to \alpha] \vdash f : [\alpha] \to \alpha}\,\texttt{ax} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{f : [[\alpha] \to \alpha] \vdash f : [\alpha] \to \alpha}\,\texttt{ax} \qquad \dfrac{\dfrac{}{\Gamma_1 \vdash f : [\,] \to \alpha}\,\texttt{ax}}{\Gamma_1 \vdash f^\omega : \alpha}\,\texttt{app}}{\Gamma_2 \vdash f^\omega : \alpha}\,\texttt{app}}{\vdots}}{\Gamma_{n-1} \vdash f^\omega : \alpha}}{\Gamma_n \vdash f^\omega : \alpha}\,\texttt{app}$$

By **truncation**, we mean that the finite derivation Π'_n can be (informally) obtained from the infinite one Π' by erasing some elements from the infinite multisets appearing in the derivation. Conversely, we see that Π' is the graphical **join** of the Π'_n: Π' is obtained by superposing all the derivations Π'_n on the same (infinite) sheet of paper.

However, we are still stuck: we do not know how to expand Π'_n, because although finite, it still types the ∞-reduced term f^ω. But notice that we can replace f^ω by $f^k(\mathtt{Y})$ inside Π'_k whenever $k \geqslant n$, because those two terms do not differ in the typed parts of Π' (**subject subsitution**). It yields a derivation $\Pi^k_n \rhd \Gamma_n \vdash f^k(\mathtt{Y}) : \alpha$. This time, Π^k_n is a derivation typing the k-th reduced of $\mathtt{Y}$, so we can expand it k times, yielding a derivation Π_n (Π_n does not depend on k). Then, we can rebuild a derivation Π such that each Π_n is a truncation of Π the same way Π'_n is of Π' (Π can be seen as the "graphical" join of the Π_n).

Thus, the ideas of truncation, subject subsitution and join indicate us how to perform ∞-subject expansion (*cf.* **??**). The particular form of Π_n and Π does not matter. Let us just say here that the Π_n involve a family of types $(\gamma)_{n \geqslant 1}$ inductively defined by $\gamma_1 = [\,] \to \alpha$ and $\gamma_{n+1} = [\gamma_i]_{1 \leqslant i \leqslant n} \to \alpha$ and Π involves an infinite type γ satisfying $\gamma = [\gamma]_\omega \to \alpha$.

Unfortunately, it is not difficult to see that the type γ also allows to type the non-HN term $\Delta\Delta$. Indeed, $x : [\gamma]_{n \in \omega} \vdash xx : \alpha$ is derivable, so $\vdash \Delta : \gamma$ and $\vdash \Delta\Delta : \alpha$ also are.

This last observation shows that the naive extension of the standard non-idempotent type system to infinite terms is unsound as non-HN terms can be typed. Therefore, we need to discriminate between sound derivations (like Π typing $\mathtt{Y}$) and unsound ones. For that, we define an infinitary derivation Π to be **valid** or **approximable** when Π admits finite truncations, generally denoted by ${}^{\mathrm{f}}\Pi$ – that are finite derivations of $\mathcal{M}_0$ –, so that any fixed finite part of Π is contained in some truncation ${}^{\mathrm{f}}\Pi$ (for now, a finite part of Π informally denotes a finite selection of graphical symbols of Π, a formal definition is given in Sec. 3.4).

2.3 Safe Truncations of Typing Derivations

Truncating derivations can obliterate different possible *reduction choices* in system $\mathcal{M}$. This problem suggests the need for rigid constructions.

Let us consider a redex $t = (\lambda x.r)s$ and $t' = r[s/x]$. If a derivation Π types t then r has been given some type τ in some context $\Gamma, x : [\sigma_i]_{i \in I}$ through a subderivation Π_0 (see Appendix A for a figure). Also, for each $i \in I$, s has been given the type σ_i through some subderivation Π_i. We can obtain a derivation Π' typing the term t' by replacing the axiom rule yielding $x : [\sigma_i] \vdash x : \sigma_i$ by the derivation Π_i. The construction of such a Π' from Π relies generally on a result referred as the "substitution lemma".

If a type σ occurs several times in $[\sigma_i]_{i \in I}$ – say n times –, there must be n axiom leaves in Π typing x with type σ, but also n argument derivations Π_i proving $s : \sigma$. When an axiom rule typing x and an argument derivation Π_i are concluded with the same type σ, we shall informally say that we can **associate** them. It means that this axiom rule can be substituted by that argument derivation Π_i when we reduce t to produce a derivation Π' typing t'. There is not only one way to associate the Π_i to the axiom leaves typing x (there can be as many as $n!$). Observe the following independent situations:

- Assume Π_1 and Π_2 (typing s), both concluded with the same type $\sigma = \sigma_1 = \sigma_2$. Thus, we also have two axiom leaves #1 and #2 concluded by $x : [\sigma] \vdash x : \sigma$, where #1 can be associated with Π_1 or Π_2. When we truncate Π into a finite ${}^{\mathrm{f}}\Pi$, the subderivation Π_1 and Π_2 are also cut into two derivations ${}^{\mathrm{f}}\Pi_1$ and ${}^{\mathrm{f}}\Pi_2$. In each ${}^{\mathrm{f}}\Pi_i$, σ can be cut into a type ${}^{\mathrm{f}}\sigma_i$. When Π_1 and Π_2 are different, it is possible that ${}^{\mathrm{f}}\sigma_2 \neq {}^{\mathrm{f}}\sigma_1$ for every finite truncation of Π. Thus, it is possible that, for every truncation ${}^{\mathrm{f}}\Pi$, the axiom leaf #1 *cannot* be associated to ${}^{\mathrm{f}}\Pi_2$: indeed, an association that is possible in Π could be impossible for any of its truncations.
- Assume this time $\sigma_1 \neq \sigma_2$. When we truncate Π into a finite ${}^{\mathrm{f}}\Pi$, both σ_1 and σ_2 can be truncated into the same finite type ${}^{\mathrm{f}}\sigma$. We can then associate ${}^{\mathrm{f}}\Pi_1$ with axiom #2 and ${}^{\mathrm{f}}\Pi_2$ with axiom #1 inside ${}^{\mathrm{f}}\Pi$, thus producing a derivation ${}^{\mathrm{f}}\Pi'$ typing t' that has no meaning w.r.t. the possible associations in the original derivation Π.

That is why we will need a *deterministic* association between the argument derivations and the axiom leaves typing x, so that the associations between them are preserved even when we truncate derivations. System $\mathcal{M}$ does not allow to formulate a well-fit notion of approximability for derivations that would be stable under (anti)reduction and hereditary for subterms. This leads us to formulate a rigid typing system in next section.

3 A Rigid Type System

A non-negative integer is called here a **track**, an **argument track** is a integer $\geqslant 2$. Let $\mathbb{N}^*$ the set of finite sequences of non-negative integers. If $a, b \in \mathbb{N}^*$, $a \cdot b$ is the concatenation of a and b, ε is the empty-sequence and $a \leqslant b$ if there is $c \in \mathbb{N}^*$ s.t. $b = a \cdot c$. The length of a is written $|a|$. The **applicative depth** $\mathrm{ad}(a)$ of $a \in \mathbb{N}^*$ is the number of argument tracks it a holds (*e.g.* $\mathrm{ad}(0 \cdot 3 \cdot 2 \cdot 1 \cdot 1) = 2$). If $a \in \mathbb{N}^*$, the **collapse** of a, written $\overline{a}$, is obtained by replacing in a very track > 3 by 2, *e.g.* $\overline{0 \cdot 5 \cdot 1 \cdot 3 \cdot 2} = 0 \cdot 2 \cdot 1 \cdot 2 \cdot 2$.

A **tree** A of $\mathbb{N}^*$ is a non-empty subset of $\mathbb{N}^*$ that is downward-closed for the prefix order ($a \leqslant a' \in A$ implies $a \in A$).

A subset $F \subset \mathbb{N}^*$ is a **forest** if $F = A - \{\varepsilon\}$ for some tree A such that $0, 1 \notin F$.

3.1 Infinitary Terms

Let $\mathcal{V}$ be a countable set of term variables. The set of terms Λ^{111} is defined *coinductively*:

$$t, u \ ::= \ x \parallel \lambda x.t \parallel tu$$

The parsing tree of $t \in \Lambda^{111}$, also written t, is the labelled tree on $\Sigma_t := \mathcal{V} \cup \{\lambda x \mid x \in \mathcal{V}\} \cup \{@\}$ defined coinductively by: $\mathtt{supp}(x) = \{\varepsilon\}$ and $x(\varepsilon) = x$, $\mathtt{supp}(\lambda x.t) = \{\varepsilon\} \cup 0 \cdot \mathtt{supp}(t)$, $(\lambda x.t)(\varepsilon) = \lambda x$ and $(\lambda x.t)(0 \cdot b) = t(b)$, $\mathtt{supp}(tu) = \{\varepsilon\} \cup 1 \cdot \mathtt{supp}(t) \cup 2 \cdot \mathtt{supp}(u)$, $(tu)(\varepsilon) = @$, $(tu)(1 \cdot b) = t(b)$ and $(tu)(2 \cdot b) = u(b)$.

The abstraction λx binds x in t and α-equivalence can be defined properly [7].

The relation $t \xrightarrow{b} t'$ is defined by induction on $b \in \{0, 1, 2\}^*$: $(\lambda x.r)s \xrightarrow{\varepsilon} r[s/x]$, $\lambda x.t \xrightarrow{0 \cdot b} \lambda x.t'$ if $t \xrightarrow{b} t'$, $t_1 t_2 \xrightarrow{1 \cdot b} t'_1 t_2$ if $t_1 \xrightarrow{b} t_1$, $t_1 t_2 \xrightarrow{2 \cdot b} t_1 t'_2$ if $t_2 \xrightarrow{b} t'_2$. We define β**-reduction** by $\rightarrow = \bigcup\limits_{b \in \{0, 1, 2\}^*} \xrightarrow{b}$.

If $b = (b_i)_{i \in \mathbb{N}}$ is an *infinite* sequence of integers, we extend $\mathrm{ad}(b)$ as $|\{i \in \mathbb{N} \mid b_i \geqslant 2\}|$ and we say that b is an infinite branch of $t \in \Lambda^{111}$ if, for all $n \in \mathbb{N}$, $b_0 \cdot b_1 \cdot \ldots \cdot b_n \in \mathtt{supp}(t)$. The calculus Λ^{001} is the set of terms $t \in \Lambda^{111}$ such that, for every infinite branch b of t, $\mathrm{ad}(b) = \infty$. Thus, for 001-terms, infinity is allowed, provided we descend infinitely many times in application arguments.

We define a reduction sequence of length $\leqslant \omega$ of 001-terms $t_0 \overset{b_0}{\to} t_1 \overset{b_1}{\to} t_2 \ldots$ to be **strongly converging** if it is finite or if $\lim \mathrm{ad}(b_n) = +\infty$. See [7] for an in-depth study of **strongly converging reduction sequences (s.c.r.s)**. A compression property allows us to consider only sequences of length $\leqslant \omega$ without loss of generality. Assuming strong convergence, let $b \in \mathbb{N}^*$ and $N \in \mathbb{N}$ s.t. $\forall n \geqslant N,\ \mathrm{ad}(b_n) > \mathrm{ad}(b)$. Then, either $\forall n \geqslant N,\ b \notin \mathtt{supp}(t_n)$ or $\forall n \geqslant N,\ b \in \mathtt{supp}(t_n)$ and $t_n(b) = t_N(b)$. Let B' be the set of all the $b \in \mathbb{N}^*$ in the latter case and t' the labelled tree define by $\mathtt{supp}(t') = B'$ and $t'(b) = t_N(b)$ for any large enough N. We notice that $t' \in \Lambda^{111}$. Actually, t' is a 001-term (because at fixed applicative depth, t' must be identical to a t_N, for some large enough N) and we call t' the **limit** of the s.c.r.s. The notation $t \to^{\infty} t'$ means that there is a s.c.r.s. starting from t, whose limit is t'.

3.2 Rigid Types

If X is a set, a **(partial) sequence** of X is a familly $x = (x_k)_{k\in K}$ s.t. $K \subseteq \mathbb{N} - \{0, 1\}$ and $x_k \in X$ for all K. We say x_k is placed on **track** k inside $(x_k)_{k\in K}$ and K is the set of **roots** of x: we write $K = \mathtt{Rt}(x)$.

Let $\mathcal{X}$ be a countable set of types variables (metavariable α). The sets of (rigid) types $\mathtt{Types}^+$ (metavariables T, T_i, $\ldots$) and rigid (sequence) types $\mathtt{FTypes}^+$ (metavariables F, $\ldots$) are coinductively defined by:

$$\begin{array}{rcl} T & ::= & \alpha \parallel F \to T \\ F & ::= & (T_k)_{k\in K} \end{array}$$

▶ **Remark.**
- The **sequence type (seq.t.)** $F = (T_k)_{k\in K}$ is a sequence of types in the above acception and is seen as an intersection of the types T_k it holds.
- If $U = F \to T$, we set $\mathtt{Tl}(U) = F$ and $\mathtt{Hd}(U) = T$ (**tail** and **head**).

The equality between two types (resp. sequence types) is defined by mutual coinduction: $F \to T = F' \to T'$ if $F = F'$ and $T = T'$ and $(T_k)_{k\in K} = (T'_k)_{k\in K'}$ if $K^1 = K^2$ and for all $k \in K,\ T_k = T'_k$. It is a **syntactic equality** (unlike multiset equality). A $\mathtt{S}$-type can only be written one way.

The support of a type (resp. a sequence type), which is a tree of $\mathbb{N}^*$ (resp. a forest), is defined by mutual coinduction: $\mathtt{supp}(\alpha) = \varepsilon$, $\mathtt{supp}(F \to T) = \{\varepsilon\} \cup \mathtt{supp}(F) \cup 1 \cdot \mathtt{supp}(T)$ and $\mathtt{supp}((T_k)_{k\in K}) = \bigcup\limits_{k\in K} k \cdot \mathtt{supp}(T_k)$

A type of $\mathtt{Types}^+$ is in the set $\mathtt{Types}$ if its support does not hold an infinite branch ending by 1^ω. A sequence type $\mathtt{FTypes}^+$ is in $\mathtt{FTypes}$ if it holds only types of $\mathtt{Types}$. A (sequence) type is said to be finite when its support is. We write $(\ )$ for the forest type whose support is empty and $(T)_{i\in\{k\}}$ (only one type T, on track k) will simply be written $k \cdot T$.

When we quotient the sets $\mathtt{Types}$ and $\mathtt{FTypes}$ by a suitable congruence (collapsing the order in nested sequences), we get the set of types and multiset types of system $\mathcal{M}$ (*cf.* Appendix F).

We say that a family of seq.t. $(F^i)_{i\in I}$ is **disjoint** if the $\mathtt{Rt}(F^i)$ (i ranging over i) are pairwise disjoint. This means that there is no overlapping of typing information between the F^i. In that case, we define the **join** of $(F^i)_{i\in I}$ as the seq.t. F s.t. $\mathtt{Rt}(F) = \bigcup\limits_{i\in I} \mathtt{Rt}(F^i)$ and, for all $k \in \mathtt{Rt}(F)$, $F_k = F^i_k$ where i the unique index s.t. $k \in \mathtt{Rt}(F^i)$.

3.3 Rigid Derivations

A **(rigid) context** C is a function from $\mathcal{V}$ to $\mathtt{FTypes}$. The context $C - x$ is defined by $(C - x)(y) = C(y)$ for any $y \neq x$ and $(C - x)(x) = (\ )$. We define the join of contexts point-wise. A **judgment** is a sequent of the form $C \vdash t : T$, where C is a context, t a 001-term and $T \in \texttt{Types}$. The set $\texttt{Deriv}$ of **(rigid) derivations** (denoted P) is defined coinductively by the following rules:

$$\frac{}{x : k \cdot T \vdash x : T}\ \texttt{ax} \qquad \frac{C \vdash t : T\ (\text{at}\,0)}{C - x \vdash \lambda x.t :\ C(x) \to T}\ \texttt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S'_k)_{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash tu : T}\ \texttt{app}$$

▶ **Additional Constraints.**
- In the $\texttt{app}$-rule, the right part of the application is a sequence of judgments and we must have $(S_k)_{k \in K} = (S'_k)_{k \in K'}$ (syntactic equality).
- Still in the $\texttt{app}$-rule, the contexts must be disjoint, so that no track conflict occurs.

In the axiom rule, k is called an **axiom track**. In the $\texttt{app}$-rule Once again, the judgment $(\Delta_k \vdash u : S_k)$ is called the **track** k **premise** of the rule.

Once again, this definition is very low-level, since the involved sequence types must be *syntactically equal* to grant that the application is typable. The $\texttt{app}$-rule may also be incorrect because two sequence types $C(x)$ or $D_k(x)$ (for a $x \in \mathcal{V}$) are not disjoint (**track conflict**). However, if we change wisely the values given to the axioms tracks, we can always assume that no conflict occurs for a specific axiom rule (for instance, using a bijection between $\mathbb{N}$ and a countable disjoint union of $\mathbb{N}$).

We can define **isomorphisms of derivations**. It is formally done in Appendix E. Concretely, P_1 and P_2 are isomorphic, written $P_1 \equiv P_2$, if they type the same term, there is well-behaved labelled tree isomorphism between their support and use isomorphic types and contexts. In that case, we can define type isomorphisms that are compatible in some sense with the typing rules in the two derivations P_1 and P_2.

3.4 Components of a Rigid Derivations and Quantitativity

Thanks to rigidity, we can designate, identify and name every part of a derivation, thus allowing to formulate many associate, useful notions.

We can define the **support** of a derivation $P \rhd C \vdash t : T$: $\texttt{supp}(P) = \varepsilon$ if P is an axiom rule, $\texttt{supp}(P) = \{\varepsilon\} \cup 0 \cdot \texttt{supp}(P_0)$ if $t = \lambda x.t_0$ and P_0 is the subderivation typing t_0, $\texttt{supp}(P) = \{\varepsilon\} \cup 1 \cdot \texttt{supp}(P_1) \cup \bigcup_{k \in K} \texttt{supp}(P_k)$ if $t = t_1\, t_2$, P_1 is the left subderivation typing t_1 and P_k the right subderivation proving the track k premise. The P_k ($k \in K$) are called **argument derivations**.

If $a \in \texttt{supp}(P)$, then a points to a judgment inside P – say this judgment is $C(a) \vdash t|_{\overline{a}} : T(a)$: we say a is a **position** of P. Now, let us locate ourselves at position a: if $c \in \texttt{supp}(T(a))$, then c is a pointer to a type symbol (α or $\to$) in the type on the right side of the sequent nested a position a. We call the pair (a, c) a **right biposition**: it points to a position in a type of a judgment nested in a judgment. Likewise, if $x \in \mathcal{V}$ and $k \cdot c \in C(a)(x)$, the pair (c, x) points to a type symbol inside seq.t. $C(a)(x)$ (on the left side of the sequent) and we call the triple (a, x, c) a **left biposition**. The **bisupport** of P, written $\texttt{bisupp}(P)$ is the set of bipositions inside P and if $\texttt{b} \in \texttt{bisupp}(P)$, $P(\texttt{b})$ is the nested type symbol that $\texttt{b}$ points at.

If $a \in A := \texttt{supp}(P)$ and $x \in \mathcal{V}$, we set $\texttt{Ax}(a)(x) = \{a' \in A\ |a' \geqslant a \text{ and } t(\overline{a'}) = x\}$ if x is free at pos. $\overline{a}$ and $\texttt{Ax}(a)(x) = \emptyset$ if not: it is the set of (positions of) axiom leaves typing x above a. If $a \in A$ is an axiom, we write $\texttt{tr}(a)$ for its associated axiom track. The presence of an infinite branch inside a derivation makes it possible that a type in a context is not created in an axiom rule. This yields the notion of **quantitative derivation**, in which this does not happen:

▶ **Definition 1.** A derivation P is **quantitative** when, for all $a \in A$ and $x \in \mathcal{V}$, $C(a)(x) = \bigcup_{a' \in \mathtt{Ax}(a)(x)} \mathtt{tr}(a') \cdot T(a')$.

Now, assume P is quantitative. For all $a \in A$ and $x \in \mathcal{V}$, we set $\mathtt{AxTr}(a)(x) = \mathtt{Rt}(C(a)(x))$. For all $a \in A$, $x \in \mathcal{V}$ and $k \in \mathtt{AxTr}(a)(x)$, we write $\mathtt{pos}(a, x, k)$ for the unique position $a' \in \mathtt{Ax}(a)(x)$ such that $\mathtt{tr}(a') = k$.

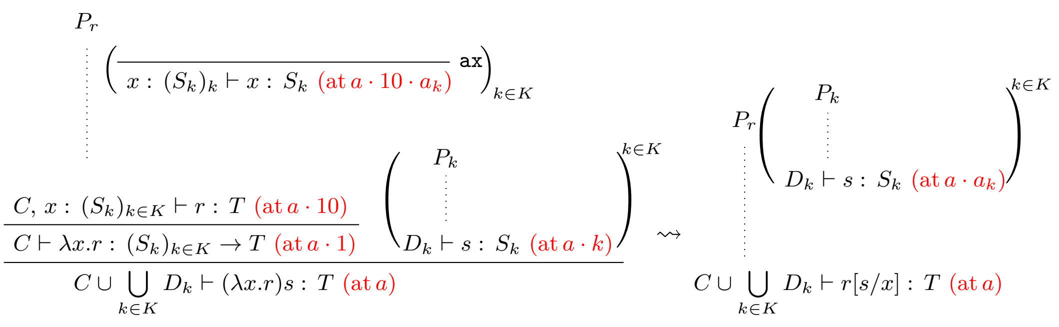
4 Dynamics

In this section, we explore the way reduction is performed inside a derivation and introduce the notion of approximations and approximable derivations. We assume $t|_b = (\lambda x.r)s$ and $t \xrightarrow{b} t'$ and we consider a derivation P s.t. $P \rhd C \vdash t : T$. The letter a will stand for a representative of b and the letter α for positions inside $A := \mathtt{supp}(P)$ (and *not* for type variables).

4.1 Residual (bi)positions

When $a \in \mathtt{Rep}_A(b)$, we set $\mathtt{RedTr}(a) = \mathtt{Rt}(C(a \cdot 10)(x))$. For $k \in \mathtt{RedTr}(a \cdot 10)(x)$, we write a_k for the unique $a_k \in \mathbb{N}^*$ such that $\mathtt{pos}(a \cdot 10, x, k) = a \cdot 10 \cdot a_k$.

Assume $t|_{\overline{a}}$ is a redex $(\lambda x.r)s$. We want to grant subject reduction according to the picture below:


P_r
$\mathtt{ax}$
$x : (S_k)_k \vdash x : S_k$ (at $a \cdot 10 \cdot a_k$)
$k \in K$
$C, x : (S_k)_{k \in K} \vdash r : T$ (at $a \cdot 10$)
$C \vdash \lambda x.r : (S_k)_{k \in K} \to T$ (at $a \cdot 1$)
P_k
$D_k \vdash s : S_k$ (at $a \cdot k$)
$k \in K$
$C \cup \bigcup_{k \in K} D_k \vdash (\lambda x.r)s : T$ (at a)
$\rightsquigarrow$
P_r
P_k
$k \in K$
$D_k \vdash s : S_k$ (at $a \cdot a_k$)
$C \cup \bigcup_{k \in K} D_k \vdash r[s/x] : T$ (at a)

Derivation typing $(\lambda x.r)s$

- P_r is the subderiv. (above a) typing r.
- In P_r, the axiom rule (typing x) using track k is at position $a \cdot 10 \cdot a_k$.
- The arg. deriv. P_k yields the track k premise.

Derivation typing $r[s/x]$

- The application and abstraction of the redex have been destroyed.
- In P_r, the arg. deriv. P_k has replaced x-axiom using track k.

Notice how this transformation is deterministic: for instance, assume $7 \in K$. There must be an axiom rule typing x using axiom track 7 *e.g.* $x : 7 \cdot S_7 \vdash x : S_7$ at position $a \cdot 10 \cdot a_7$ and also a subderivation at argument track 7, namely, P_7 concluded by $s : S_7$ at position $a \cdot 7$. Then, when we fire the redex at position b, the subderivation P_7 *must* replace the axiom rule on track 7, even if there may be several $k \neq 7$ such that $S_k = S_7$ (compare with § 2.3).

The figure above represents the quantitative case but the following construction does not assume P to be quantitative (although motivated by it). The notion of *residual* (right bi)positions tells us where a (right bi)position inside P will be placed in derivation P'. Assume $\alpha \in A$, $\overline{\alpha} \neq a$, $a \cdot 1$, $a \cdot 10 \cdot a_k$ for no $a \in \mathtt{Rep}_A(b)$ and $k \in \mathtt{RedTr}(a)$. The **residual position** of α, written $\mathtt{Res}_b(\alpha)$, is defined as follows *i.e.* (1) if $\alpha \geqslant a \cdot k \cdot \alpha_0$ for some $a \in \mathtt{Rep}(b)$ and $k \geqslant 2$, then $\mathtt{Res}_b(\alpha) = a \cdot a_k \cdot \alpha_0$ (2) if $\alpha = a \cdot 10 \cdot \alpha_0$ for some $a \in \mathtt{Rep}(b)$, then $\mathtt{Res}_b(\alpha) = a \cdot \alpha_0$ and (3) if $\overline{a} \not\geqslant b$, $\mathtt{Res}_b(\alpha) = a$.

We set $A' = \mathtt{codom}(\mathtt{Res}_b)$ (**residual support**). Now, whenever $\alpha' := \mathtt{Res}_b(\alpha)$ is defined, the **residual biposition** of $\mathtt{b} := (\alpha, \gamma) \in \mathtt{bisupp}(P)$ is $\mathtt{Res}_b(\mathtt{b}) = (\alpha', \gamma)$. We notice that $\mathtt{Res}_b$ is an *injective*, *partial* function, both for positions and right bipositions. In particular, $\mathtt{Res}_b$ is a bijection from $\mathtt{dom}(\mathtt{Res}_b)$ to A' and we write $\mathtt{Res}_b^{-1}$ for its inverse. For any $\alpha' \in A'$, let $C'(\alpha')$ be the context defined by $C'(\alpha') = (C(\alpha) - x) \cup \bigcup_{k \in K(\alpha)} C(\alpha \cdot k)$, where $\alpha = \mathtt{Res}_b^{-1}(\alpha')$ and $K(a) = \mathtt{Rt}(C(a)(x))$. Notice that $C'(\alpha) = C(\alpha)$ for any $\alpha \in A$ s.t. $\overline{\alpha} \not\geqslant b$, *e.g.* $C'(\varepsilon) = C(\varepsilon)$.

4.2 Deterministic subject reduction and expansion

Let P' be the labelled tree such that $\mathtt{supp}(P') = A'$ and $P'(\alpha')$ is $C'(\alpha') \vdash t'|_{\alpha'} : T(\alpha)$ with $\alpha' = \mathtt{Res}_b(\alpha)$. We claim that P' is a correct derivation concluded by $C \vdash t' : T$: indeed, $\overline{A'} \subset \mathtt{supp}(t')$ stems from $\overline{A} \subset \mathtt{supp}(t)$. Next, for any α' and $\alpha = \mathtt{Res}_b^{-1}(\alpha)$, $t'(\overline{\alpha'}) = t(\overline{\alpha})$ and the rule at position α' is correct in P' because the rule at position α in P is correct (for the abstraction case, we notice that $t'(\overline{\alpha'}) = \lambda y$ implies $C'(\alpha')(y) = C(\alpha)(y)$).

▶ **Proposition 1** (Subject Reduction). If $t \xrightarrow{b} t'$ and $C \vdash t : T$ is derivable, then so is $C \vdash t' : T$.

With the above notations, we also write $P \xrightarrow{b} P'$. The subject-expansion property hold for quantitative derivations. Namely, we build a derivation $P \rhd C \vdash t : T$ from a derivation $P' \rhd C \vdash t' : T$, so that $P \xrightarrow{b} P'$ by using a converse method. There are several possibilities to build such a P, because we have to choose an axiom track k for each occurrence of x inside P (in that case, x is *quantitatively* typed). For instance, we can fix an injection $\lfloor \cdot \rfloor$ from $\mathbb{N}^*$ to $\mathbb{N} - \{0, 1\}$ and to choose the track $\lfloor \alpha \rfloor$ for any axiom rule created at position α.

▶ **Proposition 2** (Subject Expansion). If $t \xrightarrow{b} t'$ and $C \vdash t' : T$ is derivable, then so is $C \vdash t : T$.

Determinism make subject reduction/expansion well-behaved w.r.t. isomoprhism:

▶ **Lemma 1.** ▬ *If $P_1 \equiv P_2$, $P_1 \xrightarrow{b} P_1'$ and $P_2 \xrightarrow{b} P_2'$, then $P_1' \equiv P_1'$.*

▬ *Assume P_1 and P_2 quantitative: if $P_1 \xrightarrow{b} P'$, $P_2 \xrightarrow{b} P'$, then $P_1 \equiv P_2$.*

5 Approximable Derivations and Unforgetfulness

5.1 Approximability and Monotonicity

We define here our validity condition *i.e.* approximability. Morally, a derivation is approximable if all its bipositions are relevant.

▶ **Definition 2.** ▬ Let P and P_* be two derivations. We say P_* is an **approximation** of P, and we write $P_* \leqslant_\infty P$, if $\mathtt{bisupp}(P_*) \subset \mathtt{bisupp}\, P$ and for all $\mathtt{b} \in \mathtt{bisupp}(P_*)$, $P_*(\mathtt{b}) = P(\mathtt{b})$.

- We write $\mathtt{Approx}_\infty(P)$ for the set of approximations of a derivation P and $\mathtt{Approx}(P)$ for the set of *finite* approximations of P.

Thus, $P_* \leqslant_\infty P$ if P_* is a *sound* restriction of P of a subset of $\mathtt{bisupp}(P)$. We usually usually write ${}^{\mathrm{f}}P$ for a *finite* approximation of P and in that case only, write ${}^{\mathrm{f}}P \leqslant P$ instead of ${}^{\mathrm{f}}P \leqslant_\infty P$. Actually, $\leqslant_\infty$ and $\leqslant$ are associated to lattice structures induced by the set-theoretic union and intersection on bisupports :

▶ **Theorem 2.** *The set of derivations typing a same term t endowed with $\leqslant_\infty$ is a directed complete partial order (dcpo) .*
- *If D is a directed set of derivations typing t:*
  - *The* **join** *$\vee D$ of D is the labelled tree P defined by* $\mathtt{bisupp}(P) = \cup_{P_* \in D} \mathtt{bisupp}(P_*)$ *and* $P(\mathtt{b}) = P_*(\mathtt{b})$ *(for any $P_* \in D$ s.t. $\mathtt{b} \in \mathtt{bisupp}(P_*)$), which is a derivation.*
  - *The* **meet** *$\wedge D$ of D is the labelled tree P defined by* $\mathtt{bisupp}(P) = \cap_{P_* \in D} \mathtt{bisupp}(P_*)$ *and* $P(\mathtt{b}) = P_*(\mathtt{b})$*, which also is a derivation.*
- *If P is a derivation typing t,* $\mathtt{Approx}_\infty(P)$ *is a complete lattice and* $\mathtt{Approx}(P)$ *is a finite lattice.*

This allows us to define now the notion of approximability, related to the *finite approximations* :

▶ **Lemma 3.** - *Reduction is monotonic: if ${}^{\mathrm{f}}P \leqslant P$, ${}^{\mathrm{f}}P \xrightarrow{b} {}^{\mathrm{f}}P'$ and $P \xrightarrow{b} P'$, then ${}^{\mathrm{f}}P' \leqslant P'$.*
- *Moreover, if $P \xrightarrow{b} P'$, for any ${}^{\mathrm{f}}P' \leqslant P$, there is a unique ${}^{\mathrm{f}}P \leqslant P$ s.t. ${}^{\mathrm{f}}P \xrightarrow{b} P'$.*

▶ **Definition 3.** A derivation P is **approximable** if, for all finite ${}^0B \subset \mathtt{bisupp}(P)$, there is a ${}^{\mathrm{f}}P \leqslant P$ s.t. ${}^0B \subset \mathtt{bisupp}({}^{\mathrm{f}}P)$.

▶ **Lemma 4.** - *If P is not quantitative, then P is not approximable.*
- *If P is quantitative and $P \xrightarrow{b} P'$, then P is approximable iff P' is approximable.*

Proof. - Assume $\mathtt{b} = (a, x, k) \in \mathtt{bisupp}(P)$ is such that there is no $a_0 \in \mathtt{Ax}(a)(x)$ s.t. $\mathtt{tr}(a_0) = k$. No finite ${}^{\mathrm{f}}P \leqslant$ could contain $\mathtt{b}$, because, by typing constraints, it would also contain all the $(a', x, k) \in \mathtt{bisupp}(P)$ (there must be infinitely many).
- Assume P approximable. Let ${}^0B' \subset \mathtt{bisupp}(P')$ be finite. We set ${}^0B = \mathtt{Res}_b^{-1}({}^0B')$. There is ${}^{\mathrm{f}}P \leqslant P$ s.t. $B \subset \mathtt{bisupp}({}^{\mathrm{f}}P)$. Let ${}^{\mathrm{f}}P'$ be the reduced of ${}^{\mathrm{f}}P$ at position b. Then ${}^0B' \subset \mathtt{bisupp}({}^{\mathrm{f}}P')$.
  The proof is the same for the converse. However, $\mathtt{Res}_b$ and $\mathtt{Res}_b^{-1}$ are not defined for every biposition (*e.g.* left ones) and our argument is faulty. It is not hard to avoid this problem (it is done in Appendix B), using *interdependencies* between bipositions.

◀

5.2 Unforgetfulness

The left side of an arrow is regarded as having a negative **polarity** and its right side as having a positive polarity. The type characterization of the WN terms in the finitary calculus (th. 4, ch.3, [8]) relies on the notion of positive and negative occurrences of the "meaningless" type Ω. We adapt this criterion, which motivates:

▶ **Definition 4.** - The sets $\mathtt{EFO}^+(\mathtt{U})$ and $\mathtt{EFO}^-(\mathtt{U})$ are defined by mutual coinduction for U, a type T or a forest type F. $\mathtt{EFO}$ stands for **empty forest occurrences** and $\pm$ indicates their polarity. The symbol $\mp$ is $-/+$ when $\pm$ is $+/-$.
  - $\mathtt{EFO}^\pm(\alpha) = \emptyset$ for $\alpha \in \mathcal{X}$.

- $\mathtt{EFO}^{\pm}((\,)\to\mathtt{T}) = \{\varepsilon\}\cup 1\cdot\mathtt{EFO}^{\pm}(\mathtt{T})$
- If $F\neq(\,)$, $\mathtt{EFO}^{\pm}(\mathtt{F}\to\mathtt{T}) = \mathtt{EFO}^{\mp}(\mathtt{F})\cup 1\cdot\mathtt{EFO}^{\pm}(\mathtt{T})$.
- $\mathtt{EFO}^{\pm}(\mathtt{F}) = \bigcup_{\mathtt{k}\in\mathtt{K}} \mathtt{k}\cdot\mathtt{EFO}^{\pm}(\mathtt{T_k})$ with $F=(T_k)_{k\in K}$.

- We say a judgment $C\vdash t:T$ is **unforgetful**, when, for all $x\in\mathcal{V}$, $\mathtt{EFO}^{-}(\mathtt{C(x)})=\emptyset$ and $\mathtt{EFO}^{+}(\mathtt{T})=\emptyset$. A derivation proving such a judgment is also said to be unforgetful.

▶ **Lemma 5.** *If $P\rhd C\vdash t:T$ is an unforgetful derivation typing a HNF $t=\lambda x_1\ldots x_p.(x\,t_1)\ldots t_q$, then, there are unforgetful subderivations of P typing t_1, t_2,..., t_q.*
Moreover, if P is approximable, so are they.

Proof. Whether $x=x_i$ for some i or not, the unforgetfulness condition grants that every argument of the head variable x is typed, since $(\,)$ cannot occur negatively in its unique given type. ◀

▶ **Lemma 6.** *If $P\rhd C\vdash t:T$ is a finite derivation, then t is head normalizable.*

Proof.
- By typing constraints, the head redex (if t is not already in HNF) must be typed.
- When we reduce a typed redex, the number of rules of the derivation must strictly decrease (at least one @-rule and one λx-rule disappear). See Appendix A.3.
- Thus, the head-reduction strategy must halt at some point.

◀

▶ **Proposition 3.** If a term t is typable by a unforgetful approximable derivation, then it is WN (in other words, it is HHN).

Proof. Consequence of the two former lemmas. ◀

5.3 The infinitary subject reduction property

In this section, we show how to define a derivation P' typing t' from a derivation typing a term t that strongly converges towards t'. Actually, when a reduction is performed at depth n, the contexts and types are not affected below n. Thus, a s.c.r.s. stabilizes contexts and types at any fixed depth. It allows to define a derivation typing the limit t'.

The following **subject substitution lemmas** are very useful while working with s.c.r.s.:

▶ **Lemma 1.** Assume $P\rhd C\vdash t:T$ and for all $a\in A:=\mathtt{supp}(P)$, $t(\overline{a})=t'(\overline{a'})$ (no approximability condition).
Let P' be the derivation obtained from P by substituting t with t' *i.e.* $\mathtt{supp}(P')=\mathtt{supp}(P)$ and for all $a\in A$, $P'(a)=C(a)\vdash t'|_{\overline{a}}:T(a)$.
Then P' is a correct derivation.

▶ **Lemma 2.** Assume ${}^{\mathrm{f}}P\leqslant P$, $P\rhd C\vdash t:T, P'\rhd C'\vdash t':T'$ and for all $a\in{}^{\mathrm{f}}A:=\mathtt{supp}({}^{\mathrm{f}}P)$, $t(\overline{a})=t'(\overline{a})$, $C(a)=C'(a)$ and $T(a)=T'(a)$.
Let ${}^{\mathrm{f}}P'$ be the derivation obtained by replacing t with t'. Then ${}^{\mathrm{f}}P'\leqslant P'$.

Now, assume that:

- $t\to^{\infty}t'$ is a s.c.r.s. Say that this sequence is $t=t_0\overset{b_0}{\to}t_1\overset{b_1}{\to}\ldots\overset{b_{n-1}}{\to}t_n\overset{b_n}{\to}t_{n+1}\overset{b_{n+1}}{\to}\ldots$ with $b_n\in\{0,\ 1,\ 2\}^*$ and $\mathrm{ad}(b_n)\longrightarrow+\infty$.
- There is a quantitative derivation $P\rhd C\vdash t:T$ and $A=\mathtt{supp}(P)$.

By performing step by step the s.c.r.s. $b_0,\ b_1,\ldots$, we get a sequence of derivations $P_n \rhd \Gamma_n \vdash t_n : T_n$ of support A_n (satisfying $C_n(\varepsilon) = C(\varepsilon)$ and $T_n(\varepsilon) = T(\varepsilon)$). When performing $t_n \overset{b_n}{\to} t_{n+1}$, notice that $C_n(c)$ and $T_n(c)$ are not modified for any c such that $b_n \not\leqslant \overline{c}$.

Let $a \in \mathbb{N}^*$ and $N \in \mathbb{N}$ be such that, for all $n \geqslant N$, $|b_n| > |a|$. There are two cases:

- $a \in A_n$ for all $n \geqslant N$. Moreover, $C_n(a) = C_N(a)$, $T_n(a) = T_N(a)$ for all $n \geqslant N$, and $t'(\overline{a} = t_n(\overline{a}) = t_N(\overline{a})$.
- $a \notin A_n$ for all $n \geqslant N$.

We set $A' = \{a \in \mathbb{N}^* \mid \exists N,\ \forall n \geqslant N,\ a \in A_n\}$. We define a labelled tree P' whose support is A' by $P'(a) = C_n(a) \vdash t'|_{\overline{a}} : T_n(a)$ for any $n \geqslant N(|a|)$ (where $N(\ell)$ is the smallest rank N such that $\forall n \geqslant N,\ |a_n| > \ell$).

▶ **Proposition 4.** The labelled tree P' is a derivation (the subject-reduction property holds for s.c.r.s. without considering approximability).

Proof. Let $a \in A'$ and $n \geqslant N(|a|+1)$. Thus, $t'(a) = t_n(a)$ and the types and contexts involved at node a and its premises are the same in P' and P_n. So the node a of P' is correct, because it is correct for P_n. ◀

▶ **Proposition 5.** If P is approximable, so is P'.

Proof. Assume ${}^{\mathrm{f}}P \leqslant P$. Let $N = |\,\mathtt{bisupp}({}^{\mathrm{f}}P)|$. Notice that ${}^{\mathrm{f}}P$ cannot type any position whose length is greater than N.

Then, t can be reduced (in a finite number ℓ of steps) into a term t_ℓ such that $t_\ell(b) = t'(b)$ for all $b \in \{0,\ 1,\ 2\}^*$ such that $|b| \leqslant N$.

We have ${}^{\mathrm{f}}P_\ell \leqslant P_\ell$ (monotonicity). Let ${}^{\mathrm{f}}P'$ be the derivation obtained by replacing t_ℓ by t' in ${}^{\mathrm{f}}P_\ell$. The Substitution Lemmas entail that ${}^{\mathrm{f}}P'$ is a correct derivation and ${}^{\mathrm{f}}P' \leqslant P'$. ◀

6 Typing Normal Forms and Subject Expansion

In this section, we characterize all the possible quantitative derivations typing a normal form t, and show all of them to be approximable.

6.1 Positions in a Normal Form

We write $a \prec a'$ when there is a_0 such that $a_0 \leqslant a$, $a_0 \leqslant a'$, $\mathrm{ad}(a) = \mathrm{ad}(b)$ and $\mathrm{ad}(a') \geqslant \mathrm{ad}(b)$. The relation $a \prec a'$ is a *preorder* and represents an "applicative priority" w.r.t. typing. Namely, assume $\overline{a}$, $\overline{a'}$ are in $\mathtt{supp}(t)$, $a \prec a'$ and P types t. Then, $a' \in \mathtt{supp}(P)$ implies $a \in \mathtt{supp}(P)$. For instance, if $021037 \in \mathtt{supp}(P)$, then $t(\overline{02103}) = @$ and 02031, which is this application left-hand side, should also be in $\mathtt{supp}(P)$, as well as every prefix of 021037.

This motivates to say that a $A \subset \mathbb{N}^*$ is a **derivation support (d-support)** of t if $\overline{A} \subset \mathtt{supp}(t)$ and A is downward closed for $\prec$. We will show that, in that case, there is actually a P typing t s.t. $A = \mathtt{supp}(P)$ (this holds only because t is a NF).

If t is a NF and $\overline{a} \in \mathtt{supp}(t)$ (a is a d-position in t), we may have:

- $t|_{\overline{a}} = \lambda x_1 \ldots x_n.u$ where u is not an abstraction. We set then $\mathring{a} = a \cdot 0^n$. When $n \neq 0$, we say that a is an **abstraction position**.
- $t_{\overline{a}}$ is not an abstraction. Then there is a smallest prefix a' of a such that $a = a' \cdot 1^n$. We set then $\mathring{a} = a'$. When $n \neq 0$, we say a is a **partial position**.

When $a = \mathring{a}$, a is called a **full position**. The set of full positions inside a d-support A of t is also written $\mathring{A}$. In the 3 cases, $\texttt{rdeg}(a) := ||a| - |\mathring{a}||$ is the **relative degree** of position a.

We identify graphically $\mathring{a}$ and its collapse. We have 3 kinds of positions : a position is full when we can choose freely the type it makes appear. An abstraction position is a position that prefixes a full position by means of a sequence of abstraction and a partial position is a position that postfixes a full position by ....

6.2 Building a Derivation typing a Normal Form

We build here, from any given d-support A of T and function T from $\mathring{A}$ to $\texttt{Types}$, a quantitative derivation P such that $\texttt{supp}(P) = A$, giving the type $T(a)$ to $t|_{\overline{a}}$ for any $a \in \mathring{A}$. Disregarding indexation problems, we must have, by typing constraints:

- If a is an abstraction position - say $\texttt{rdeg}(a) = n$ and $t|_{\overline{a}} = \lambda x_1 \dots x_n.t_{\mathring{a}}$ -, then $T(a) = C(a \cdot 0)(x_1) \to \dots \to C(a \cdot 0^n)(x_n) \to T(\mathring{a})$, where $C(a)(x)$ is a s.t. containing every type given in $\texttt{Ax}(a)(x)$.
- If a is partial - say $a = \mathring{a} \cdot 1^n$ and $t|_{\mathring{a}} = t_{\overline{a}} t_1 \dots t_n$ -, then $T(a) = \mathrm{R}_i(a) \to \dots \to \mathrm{R}_n(a) \to T(\mathring{a})$, where $\mathrm{R}_i(a)$ is the s.t. holding all the types given to t_i (below $\mathring{a}$).

If $(T_i)_{i \in I}$ is a family of types and $(k_i)_{\in I}$ a family of pairwise distinct integers $\geqslant 2$, the notation $(k_i \cdot T_i)_{i \in I}$ will denote the forest type F s.t. $\texttt{Rt}(F) = \{k_i \mid i \in I\}$ and $F|_k = T_i$ where i is the unique index s.t. $k = k_i$.

We consider from now on an injection $a \mapsto \lfloor a \rfloor$ from $\mathbb{N}^*$ to $\mathbb{N} - \{0,\ 1\}$. To each $a \in \mathbb{N}^*$, we attribute a fresh type variable X_a.

When a is partial and $\texttt{rdeg}(a) = n$ (and thus, $a = \mathring{a} \cdot 1^n$), we set, for $1 \leqslant k$, $AP_k(a) = \{\mathring{a} \cdot 1^{n-k} \cdot \ell \in A \mid \ell \geqslant 2\}$ (AP stands for "argument positions").

- If $a \in A$ and $x \in \mathcal{V}$ is free at position a, we define the forest type $E(a)(x)$ by $E(a)(x) = (\lfloor a' \rfloor \cdot X_{a'})_{a' \in \texttt{Ax}(a)(x)}$ ($\texttt{Ax}(a)(x)$ is defined w.r.t. A). If $a \in A$ is partial and $a = \mathring{a} \cdot 1^n$ and $1 \leqslant k \leqslant n$, we define the forest type $F_k(a)$ by $F_k(a) = (\lfloor a' \rfloor \cdot X_{a'})_{a' \in AP_k(a)}$. If $a \in A$ is full, we set $S(a) = T(a)$ (in that case, $S(a)$ does not hold any X_k).
- If $a \in A$ is an abstraction position – say $t|_{\overline{a}} = \lambda x_1 \dots x_n.u$ where $t_{\mathring{a}} = u$), we set $S(a) = E(a \cdot 0)(x_1) \to \dots \to E(\mathring{a})(x_n) \to T(\mathring{a})$. If $a \in A$ is partial, we set $S(a) = F_1(a) \to F_2(a) \to \dots \to F_n(a) \to T(\mathring{a})$. We we extend T (defined on full positions) to A by the following coinductive definition: for all $a \in A$, $T(a) = S(a)[T(a')/X_{a'}]_{a' \in \mathbb{N}^*}$. For all $a \in A$, we define the contexts $C(a)$ by $C(a)(x) = E(a)(x)[T(a')/X_{\lfloor a' \rfloor}]_{a' \in \texttt{Ax}(a)(x)}$.

Those definitions are well-founded, because whether a is λx-position or a partial one, every occurrence of an X_k is at depth $\geqslant 1$ and the coinduction is *productive*. Eventually, let P be the labelled tree whose support is A and s.t., for $a \in A$, $P(a)$ is $C(a) \vdash t|_{\overline{a}} : T(a)$.

▶ **Proposition 6.** The labelled tree P is a derivation proving $C(\varepsilon) \vdash t : T(\varepsilon)$.

Proof. Let $a \in A$. Whether $t(\overline{a})$ is x, λx or @, we check the associated rule has been correctly applied. Roughly, this comes from the fact that the variable $X_{a'}$ is "on the good track" (*i.e.* $\lfloor a' \rfloor$), as well as in $F_i(a)$, thus allowing to retrieve correct typing rules. ◀

▶ **Definition 5.** The above method of building of a derivation P typing a normal form t, from a d-support A of t and a function T from full positions of A to $\texttt{Types}$, will be referred as the **trivial construction**.

▶ **Proposition 7.** A normal form $t \in \Lambda^{001}$ admits an unforgetful derivation.

Proof. We set $A = \texttt{supp}(t)$ and $T(a) = \alpha$ for each full position (where α is a type variable). In that case, the trivial construction yields an unforgetful derivation of t. ◀

It is easy to check that the above derivations yield representatives of every possible quantitative derivation of a NF (notice there is no approximability condition):

▶ **Proposition 8.** If P is a quantitative derivation typing t, then the trivial construction w.r.t. $A := \texttt{supp}(P)$ and the restriction of T on $\mathring{A}$ yields P itself.

6.3 Approximability

We explain here why every quantitative derivation P typing a normal form is approximable (see Appendix D for a complete proof). This means that we can build a finite derivation ${}^{\mathrm{f}}P \leqslant P$ containing any finite part ${}^{0}B$ of $\texttt{bisupp}(P)$. We will proceed by:

- Choosing a finite d-support ${}^{\mathrm{f}}A$ of A *i.e.* we will discard all positions in A but finitely many.
- Choosing, for each $T(a)$ s.t. a is full, a finite part of ${}^{\mathrm{f}}T(a)$ of $T(a)$.

The trivial construction using ${}^{\mathrm{f}}A$ and ${}^{\mathrm{f}}T$ will yield a derivation ${}^{\mathrm{f}}P \leqslant P$ typing t.

Namely, we fix an integer n and discard every position $a \in A$ such that $\mathrm{ad}(a) > n$ or a holds a track $\geqslant n$. It yields a finite d-support $A_n \subset A$. Then, for each $\mathring{a}$, we discard every inner position inside $T(\mathring{a})$ according to the same criterion. It yields finite types $T_n(\mathring{a})$. The trivial construction starting from A_n and T_n yields a (finite) derivation $P_n \leqslant P$. We prove then that n can always be chosen big enough to ensure that ${}^{0}B \subset \texttt{bisupp}(P_n)$.

6.4 The Infinitary Subject Expansion Property

In Section 5.3, we defined the derivation P' resulting from a s.c.r.s. from any (approximable or not) derivation P. Things do not work so smoothly for subject expansion when we try to define a good derivation P which results from a derivation P' typing the limit of a s.c.r.s. Indeed, approximability play a central role w.r.t. expansion. Assume that:

- $t \rightarrow^{\infty} t'$. Say through the s.c.r.s. $t = t_0 \overset{b_0}{\rightarrow} t_1 \overset{b_1}{\rightarrow} \ldots t_n \overset{b_n}{\rightarrow} t_{n+1} \rightarrow \ldots$ with $b_n \in \{0,\ 1,\ 2\}^*$ and $\mathrm{ad}(b_n) \longrightarrow +\infty$.
- P' is an approximable derivation of $C' \vdash t' : T'$.

The main point is to understand how subject expansion works with a finite derivation ${}^{\mathrm{f}}P' \leqslant P'$. The technique of § 2.2 can now be formally performed. Mainly, since ${}^{\mathrm{f}}P'$ is finite, for a large enough n, t' can be replaced by t_n inside ${}^{\mathrm{f}}P'$, due to the subject substitution lemmas § 5.3, which yields a finite derivation ${}^{\mathrm{f}}P_n$ typing t_n. But when t_n is typed instead of t', we can perform n steps of expansion (starting from ${}^{\mathrm{f}}P_n$) to obtain a finite derivation ${}^{\mathrm{f}}P$ typing t. Then, we define P as the join of the ${}^{\mathrm{f}}P$ when ${}^{\mathrm{f}}P'$ ranges over $\texttt{Approx}(P')$. Complete proofs and details are to be found in Appendix C.5

▶ **Proposition 9.** The subject expansion property holds for approximable derivations and strongly convergent sequences of reductions.

Since subject-reduction and expansion of infinite length (in s.c.r.s) preserve unforgetful derivation, it yields our main characterization theorem :

▶ **Theorem 7.** *A term t is weakly-normalizing in Λ^{001} if and only if t is typable by means of an approximable unforgetful derivation.*

Proof. The $\Leftarrow$ implication is given by proposition 3. For the direct one: assume t to be WN and the considered s.c.r.s. to yield the NF t' of t. Let P' be an unforgetful derivation typing t' (granted by proposition 7). Then, the derivation P obtained by the proposition below is unforgetful and types t. ◀

7 Conclusion

We have provided an intersection type system characterizing weak-normalizability in the infinitary calculus Λ^{001}. The use of functions from the set of integers to the set of types to represent intersection – instead of multisets or conjunctions – allows to express a validity condition that could only be suggested in De Carvalho's type assignment system. Our type system system is relatively simple and offers many ways to describe proofs (*e.g.* tracking, residuals).

It is then natural to seek out whether this kind of framework could be adapted to other infinitary calculi and if we could also characterize strong normalization in Λ^{001}, using for instance a memory operator [2]. Although our derivations are very low-level objects, it can be shown they allow to represent any infinitary derivation of system $\mathcal{M}$ [13]. We would like to find alternatives to the approximability condition, *e.g.* formulating it only in term of tracks. It is to be noticed that derivation approximations provide *affine* approximations that behave *linearly* in Mazza's polyadic calculus [9]. Last, the type system presented here can very easily adapted to the terms whose Böhm tree may contain $\perp$, with the same properties. In particular, two terms having the same Böhm tree can be assigned the same types in the same contexts. We would like to investigate ways to get (partial forms of) the converse.

References

1 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, pages 341–354, 2014.

2 Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Strong normalization through intersection types and memory. In *Proc. of the 10th Int. Workshop on Logical and Semantical Frameworks, with Applications (LSFA), ENTCS, Natal, Brazil, August-September 2015.*

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

4 Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.

5 Lukasz Czajka. A coinductive confluence proof for infinitary lambda-calculus. In *Rewriting and Typed Lambda Calculi - Joint International Conference, RTA-TLCA 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 164–178, 2014.

6 Jörg Endrullis, Helle Hvid Hansen, Dimitri Hendriks, Andrew Polonsky, and Alexandra Silva. A coinductive framework for infinitary rewriting and equational reasoning. In *26th International Conference on Rewriting Techniques and Applications, RTA 2015, June 29 to July 1, 2015, Warsaw, Poland*, pages 143–159, 2015.

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

8 Jean-Louis Krivine. *Lambda-calculus, types and models.* Ellis Horwood series in computers and their applications. Masson, 1993.

9 Damiano Mazza. An infinitary affine lambda-calculus isomorphic to the full lambda-calculus. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 471–480, 2012.

10 Luigi Santocanale. A calculus of circular proofs and its categorical semantics. Technical Report RS-01-15, BRICS, Dept. of Computer Science, University of Aarhus, May 2001.

11 Makoto Tatsuta. Types for hereditary head normalizing terms. In *Functional and Logic Programming, 9th International Symposium, FLOPS 2008, Ise, Japan, April 14-16, 2008. Proceedings*, pages 195–209, 2008.

12 Steffen van Bakel. Intersection type assignment systems. *Theor. Comput. Sci.*, 151(2):385–435, 1995.

13 Pierre Vial. Unpublished work.

Contents

A Performing reduction in a derivation

We assume here that $t|_b = (\lambda x.r)s\ t \xrightarrow{b} t'$.

A.1 In De Carvalho's system $\mathcal{M}_0$

We assume assume $\Pi \rhd \Gamma_\varepsilon \vdash t : \tau_\varepsilon$. We consider the (non-deterministic) transformation below, that we perform on any subderivation of Π corresponding to the position b. For such a subderivation, it is meant that the Π are the argument subderivations typing s and that Π_0 is the subderivation typing r. We have indicated between brackets the positions of the axiom leaves typing x, the bound variable to be substituted during the reduction.

$$\dfrac{\dfrac{\begin{array}{c}\Pi_0 \quad \left(\dfrac{}{x : [\sigma_i] \vdash x : \sigma_i}\ \text{ax}\right)_{i \in I} \\ \vdots \\ \Gamma, x : [\sigma_i]_{i\in I} \vdash r : \tau\end{array}}{\Gamma \vdash \lambda x.r : [\sigma]_{i\in I} \to \tau} \qquad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i\end{array}\right)_{i\in I}}{\Gamma + \sum_{i\in I} \Delta_i \vdash (\lambda x.r)s : \tau} \quad \rightsquigarrow \quad \begin{array}{c}\Pi_0 \quad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i\end{array}\right)_{i\in I} \\ \vdots \\ \Gamma + \sum_{i\in I} \Delta_i \vdash r[s/x] : \tau\end{array}$$

Thus, from the type-theoretical point of view, in De Carvalho's type assignment system, reduction consists in:

- Destroying the application and abstraction rules related to to fired redex and the axiom rules of the variable to be substituted.
- Moving argument parts of the redex, without adding any rule (contrary to the idempotent intersection frameworks).

A.2 In rigid derivations

We assume here that $P \rhd C_\varepsilon \vdash t : \varepsilon$.
The derivation P' defined in §4.2 can obtained by performing the following transformation at position a, when a ranges over $\mathtt{Rep}_A(b)$.

The subderivation $P|_a$ must look like:

$$\dfrac{\dfrac{\begin{array}{c}P_0 \quad \left(\dfrac{}{x : (S_k)_k \vdash x : S_k\ \text{(at } a \cdot 10 \cdot a_k)}\ \text{ax}\right)_{k\in K} \\ \vdots \\ \hline C, x : (S_k)_{k\in K} \vdash r : T\ \text{(at } a \cdot 10)\end{array}}{C \vdash \lambda x.r : (S_k)_{k\in K} \to T\ \text{(at } a \cdot 1)} \qquad \left(\begin{array}{c}P_k \\ \vdots \\ D_k \vdash s : S_k\ \text{(at } a \cdot k)\end{array}\right)_{k\in K}}{C \cup \bigcup_{k\in K} D_k \vdash (\lambda x.r)s : T\ \text{(at } a)}$$

The subderivation $P|_a$ must look like:

$$\dfrac{\dfrac{\begin{array}{c} P_0 \quad \left(\dfrac{}{x : (S_k)_k \vdash x : S_k}\,\texttt{ax}\right)_{k\in K} \\ \vdots \end{array}}{C,\, x : (S_k)_{k\in K} \vdash r : T\ \text{(at } a\cdot 10\text{)}}}{C \vdash \lambda x.r : (S_k)_{k\in K} \to T\ \text{(at } a\cdot 1\text{)}} \quad \left(\begin{array}{c} P_k \\ \vdots \\ D_k \vdash s : S_k\ \text{(at } a\cdot k\text{)} \end{array}\right)_{k\in K}}{C \cup \bigcup_{k\in K} D_k \vdash (\lambda x.r)s : T\ \text{(at } a\text{)}}$$

We replace $P|_a$ by the derivation below:

$$\dfrac{\begin{array}{c} P_0 \quad \left(\begin{array}{c} P_k \\ \vdots \\ D_k \vdash s : S_k\ \text{(at } a\cdot a_k\text{)} \end{array}\right)_{k\in K} \\ \vdots \end{array}}{C \cup \bigcup_{k\in K} D_k \vdash r[s/x] : T\ \text{(at } a\text{)}}$$

Notice how this transformation is deterministic: for instance, assume $7 \in K$. There must be an axiom rule typing x using axiom track 7 : it yields $x : (S_7)_7 \vdash x : S_7$ at position $a \cdot 10 \cdot a_7$. There must also a subderivation at argument track 7: it is P_7 at position $a \cdot 7$. Then, when we fire the redex at position b, the subderivation P_7 *must* replace the axiom rule on track 7, even if there may be several $i \neq 7$ such that $S_i = S_7$.

A.3 The quantitative argument

We expose De Carvalho's original argument in system $\mathcal{M}_0$. We formulate it here w.r.t. rigid derivations.

We work here with *finite* derivations. If P is a finite derivation, we write $\texttt{nr}(\texttt{P})$ ("number of rules") for $|\,\texttt{supp}(P)|$.

We want to show that if the redex at position b is typed – *i.e.* if $b \in \overline{\texttt{supp}(P)}$–, then the reduced derivation P' verifies $\texttt{nr}(\texttt{P}') < \texttt{nr}(\texttt{P})$.

With the same notations as previously, we have $\texttt{nr}(\texttt{P}'|_\texttt{a}) = \texttt{nr}(\texttt{P}|_\texttt{a}) - 2 - |\texttt{K}|$ since the @-rule at position a disappear, as well as the λ-rule at position $a \cdot 1$ and the $|K|$ axiom rules typing X.

Thus, $\texttt{nr}(\texttt{P}') \leqslant \texttt{nr}(\texttt{P})$ as soon as there is a $a \in \texttt{supp}(P)$ s.t. $\overline{a} = b$.

B Equinecessity, Reduction and Approximability

The rigid construction presented here ensure "trackability", contrary to multiset construction of system $\mathcal{M}_0$. We show here a few applications useful to prove that approximability is stable under reduction or expansion (Lemma 4) . We consider a *quantitative* derivation P, with the usual associated notations (including $A = \texttt{supp}(P)$).

B.1 Equinecessary bipositions

▶ **Definition 6.** Let $\texttt{b}_1$, $\texttt{b}_2$ two bipositions of P.

- We say $\texttt{b}_1$ **needs** $\texttt{b}_2$ if, for all ${}^{\mathrm{f}}P \leqslant P$, $\texttt{b}_1 \in {}^{\mathrm{f}}P$ implies $\texttt{b}_2 \in {}^{\mathrm{f}}F$.
- We say $\texttt{b}_1$ and $\texttt{b}_2$ are **equinecessary** (written $\texttt{b}_1 \leftrightarrow \texttt{b}_2$) if, for all ${}^{\mathrm{f}}P \leqslant P$, $\texttt{b}_1 \in {}^{\mathrm{f}}P$ iff $\texttt{b}_2 \in {}^{\mathrm{f}}P$

There are many elementary equinecessity cases that are easy to observe. We need only a few one and we define $\texttt{up}(\texttt{b})$ and $\texttt{top}(\texttt{b})$ s.t. $\texttt{b} \leftrightarrow \texttt{up}(\texttt{b})$ and $\texttt{b} \leftrightarrow \texttt{top}(\texttt{b})$ for all $\texttt{b}$.

- $\texttt{up}(\texttt{b})$ is defined for any $\texttt{b} \in \texttt{bisupp}(P)$ which is not on an axiom leaf.
  - $\texttt{up}(\texttt{a},\ \texttt{x},\ \texttt{k}\cdot\texttt{c}) = (\texttt{a}\cdot\ell,\ \texttt{x},\ \texttt{k}\cdot\texttt{c})$, where ℓ is the unique integer s.t. $(a\cdot\ell,\ x,\ k\cdot c) \in \texttt{bisupp}(P)$.
  - If $t(\overline{a}) = \lambda x$, $\texttt{up}(\texttt{a},\ \varepsilon) = (\texttt{a}\cdot 0,\ \varepsilon)$, $\texttt{up}(\texttt{a},\ 1\cdot\texttt{c}) = (\texttt{a}\cdot 0,\ \texttt{c})$ and $\texttt{up}(\texttt{a},\ \texttt{k}\cdot\texttt{c}) = (\texttt{a}\cdot 0,\ \texttt{x},\ \texttt{k}\cdot\texttt{c})$ if $k \geqslant 2$.
  - If $t(\overline{a}) = @$, $\texttt{up}(\texttt{a},\ \texttt{c}) = (\texttt{a}\cdot 1,\ 1\cdot\texttt{c})$.
- $\texttt{top}(\texttt{b})$ is a *right biposition* and is defined by induction for any $\texttt{b} \in \texttt{bisupp}(P)$.
  - If $\texttt{b}$ is not on an axiom leaf, then $\texttt{top}(\texttt{b}) = \texttt{top}(\texttt{up}(\texttt{b}))$.
  - If $t(\overline{a}) = x$, $\texttt{top}(\texttt{a},\ \texttt{x},\ \texttt{k}\cdot\texttt{c}) = \top(\texttt{a},\ \texttt{c}) = (\texttt{a},\ \texttt{c})$

The induction defining is well-founded because of the form of the supports of the 001-terms and because P is quantitative.

Assume $t|_b = (\lambda x.r)s$. A very important case of equinecessity is this one: $(a\cdot 1,\ k\cdot c) \leftrightarrow (a\cdot 10\cdot a_k,\ x,\ k\cdot c)$ and $(a\cdot 1,\ k\cdot c) \leftrightarrow (a\cdot k,\ c)$. Thus, $(a\cdot 10\cdot a_k,\ c) \leftrightarrow (a\cdot k,\ c)$.

B.2 Approximability is stable under (anti)reduction

We assume here that $P \to P'$ (P is still assumed to be quantitative). Let ${}^0B \subset \texttt{bisupp}(P)$ a finite part. We notice that $\texttt{Res}_b$ is defined for any right biposition which is on an axiom leaf typing $y \neq x$.

So, let ${}^0\tilde{B}$ be the set obtained from $\texttt{top}(\texttt{B})$ by replacing any $(a\cdot 10\cdot a_k,\ c)$ by $(a\cdot k,\ c)$. Then $|{}^0\tilde{B}| \leqslant |{}^0B|$ and any $\texttt{b} \in {}^0B$ is equinecessary with a $\tilde{\texttt{b}} \in {}^0\tilde{B}$. So the partial proof of Lemma 4 is valid for ${}^0\tilde{B}$. By equinecessity, it entails it is also valid for ${}^0B$.

For the converse implication, we just have to replace ${}^0B'$ by $\texttt{top}({}^0\texttt{B}')$.

C Lattices of (finite or not) approximations

C.1 Types, Forest Types and Contexts

▶ **Definition 7.** Let U_1 and U_2 two (forest) types. If, as a labelled tree or forest, U_1 is a restriction of U_2, we write $U_1 \leqslant_\infty U_2$. When U_1 is finite, we write simply $U_1 \leqslant U_2$
We set $\mathtt{Approx}(U) = \{{}^{\mathrm{f}}U \mid {}^{\mathrm{f}}U \leqslant U\}$ and $\mathtt{Approx}_\infty(U) = \{U_0 \mid U_0 \leqslant_\infty U\}$

▶ **Lemma 3.** Let $(T_i)_{i\in I}$ be a non-empty family of types, such that $\forall i, j \in I$, $\exists T \in \mathtt{Types}$, T_i, $T_j \leqslant_\infty T$ (*i.e.* T_i, T_j have an upper bound inside $\mathtt{Types}$).
We define the labelled tree $T(I)$ by $\mathtt{supp}(T(I)) = \bigcap\limits_{i\in I} \mathtt{supp}(T_i)$ and $T(I)(c) = T_i(c)$ for any i.
Then, this definition is correct and $T(I)$ is a type (that is finite if one the T_i is). We write $T(I) = \bigwedge\limits_{i\in I} T_i$.

Proof. Since $\mathtt{supp}(T(I)) = \bigcap\limits_{i\in I} \mathtt{supp}(T_i)$, $\mathtt{supp}(T(I))$ is a tree without infinite branch ending by 1^ω.

Let us assume $c \in \mathtt{supp}(T(I))$. Then, for all $i \in I$, $c \in \mathtt{supp}(T_i)$. For i, $j \in I$, there is a T s.t. T_i, $T_j \leqslant_\infty T$. Thus, $T_i(c) = T(c) = T_j(c)$ and the definition of $T(I)$ is correct.

When $T(I)(c) =\to$, then $c \in \mathtt{supp}(T_i)$ for all $i \in I$, so $c \cdot 1 \in \mathtt{supp}(T_i)$ for all $i \in I$, so $c \cdot 1 \in \mathtt{supp}(T(I))$, so $T(I) \in \mathtt{Types}$. ◀

▶ **Lemma 4.** Let $(T_i)_{i\in I}$ be a non-empty family of types, such that $\forall i, j \in I$, $\exists T \in \mathtt{Types}$, T_i, $T_j \leqslant_\infty T$.
We define the labelled tree $T(I)$ by $\mathtt{supp}(T(I)) = \bigcup\limits_{i\in I} \mathtt{supp}(T_i)$ and $T(I)(c) = T_i(c)$ for any i s.t. $c \in \mathtt{supp}(T_i)$.
Then, this definition is correct and $T(I)$ is a type (that is finite if I is finite and all the T_i are). We write $T(I) = \bigvee\limits_{i\in I} T_i$.

Proof. Since $\mathtt{supp}(T(I)) = \bigcup\limits_{i\in I} \mathtt{supp}(T_i)$, then $\mathtt{supp}\, T(I)$ is a tree.

Let us assume $c \in \mathtt{supp}(T(I))$ and $c \in \mathtt{supp}(T_i) \cap \mathtt{supp}(T_j)$. Let T be a type s.t. T_i, $T_j \leqslant_\infty T$. Thus, we have $T_i(c) = T(c) = T_j(c)$ and the definition of $T(I)$ is correct.

Moreover, since T_i is a type, there is a $n \geqslant 0$ s.t. $c \cdot 1^n$ is a leaf of $\mathtt{supp}(T_i)$ and $T_i(c \cdot 1^n) = \alpha$ (α is a type variable). Since $T_i \leqslant_\infty T$, $T(c \cdot 1^n) = \alpha$. Since T is a correct type, $T(c \cdot 1^n) = \alpha$ entails that $c \cdot 1^n$ is a leaf of $\mathtt{supp}(T)$ and $T(c \cdot 1^n) = \alpha$. Since $T_j \leqslant_\infty T$, $c \cdot 1^n$ is a leaf of $\mathtt{supp}(T_j)$ and $T_j(c \cdot 1^n) = \alpha$. So $c \cdot 1^n$ is a leaf of $T(I)$ and $T(I)(c) = \alpha$ and $\mathtt{supp}(T(I))$ cannot have an infinite branch ending by 1^ω.

If moreover $T(I)(c) =\to$, then $c \in \mathtt{supp}(T_i)$. Since T_i is a correct type, $c \cdot 1 \in \mathtt{supp}(T_i)$ and thus, $c \cdot 1 \in \mathtt{supp}(T(I))$, so $T(I) \in \mathtt{Types}$. ◀

▶ **Proposition 10.** The set $\mathtt{Types}$ endowed with $\leqslant_\infty$ is a **direct complete partial order (d.c.p.o.)**. The join is given by the above operator.
Moreover, for any type T, $\mathtt{Approx}(T)$ is a distributive lattice and $\mathtt{Approx}_\infty(T)$ is a complete distributive lattice, and the meet is given by the above operator.

Proof. The distributivity stems from the distributivity of the set-theoretic union and intersection. ◀

We can likewise construct the joins and the meets of families of forest types (via the set-theoretic operations on the support), provided every pair of elements have an upper bound. The set `FTypes` also is a d.c.p.o. and for all f.t. F, $\mathtt{Approx}(F)$ is a distributive lattice and $\mathtt{Approx}_\infty(F)$ is a complete distributive lattice.

C.2 A Characterization of Proper Bisupports

Let $B_0 \subset \mathtt{bisupp}(P)$. We want to know on what condition B_0 is the support of a derivation $P_0 \leqslant_\infty P$.

We write A_0 for the set of all underlying outer positions of $\mathtt{b}$, when $\mathtt{b}$ spans over B.

- For all $a \in A_0$, we write $T_0(a)$ the labelled tree induced by $T(a)$ on $\{c \in \mathbb{N}^* \mid (a,\ c) \in B_0\}$.
- For all $a \in A_0$ and $x \in \mathcal{V}$, we write $C_0(a)(x)$ for the function induced by $C(a)(x)$ on $\{c \in \mathbb{N}^* \mid (a,\ x,\ c) \in B_0\}$.

A tedious verification grants that there is a $P_0 \leqslant_\infty P$ s.t. $\mathtt{bisupp}(P_0) = B_0$ iff the conditions below are satisfied:

- Support related conditions: A_0 is a tree s.t.:
  - For all $a \in A_0$ s.t. $t(\overline{a}) = @$, $a \in A_0$ implies $(a \cdot 1,\ \varepsilon) \in B_0$.
  - For all $a \in A_0$ s.t. $t(\overline{a}) = \lambda x$, $a \in A_0$ implies $(a,\ \varepsilon) \in B_0$.

- Inner supports related conditions: for all $a \in A$, $T_0(a)$ is a type and for all $x \in \mathcal{V}$, $C_0(a)(x)$ is a forest type.

- Axiom rule related conditions: for all $x \in \mathcal{V}$, all $a \in \mathtt{Ax}(x)$ and all $c \in \mathtt{supp}(T(a))$, $(a,\ c) \in B_0$ iff $(a,\ x,\ k \cdot c) \in B_0$, where $k = \mathtt{tr}(a)$.

- Abstraction rule related conditions: for all $x \in \mathcal{V}$, all $a \in A$ s.t. $t(\overline{a}) = \lambda x$:
  - For all $c \in \mathbb{N}^*$, $(a,\ 1 \cdot c) \in B_0$ iff $(a \cdot 0,\ c) \in B_0$.
  - For all $c \in \mathbb{N}^*$ and all $k \geqslant 2$, $(a,\ k \cdot c) \in B_0$ iff $(a \cdot 0,\ x,\ k \cdot c) \in B_0$.
  - For all $y \in \mathcal{V} - \{x\}$, $k \geqslant 2$ and $c \in \mathbb{N}^*$, $(a,\ y,\ k \cdot c) \in B_0$ iff $(a \cdot 0,\ y,\ k \cdot c) \in B_0$.

- Application related conditions: for all $a \in A$ s.t. $t(\overline{a}) = @$:
  - For all $c \in \mathbb{N}^*$, $(a,\ c) \in B_0$ iff $(a \cdot 1,\ 1 \cdot c) \in B_0$.
  - For all $k \geqslant 2$ and all $c \in \mathbb{N}^*$, $(a,\ k \cdot c) \in B_0$ iff $(a \cdot k,\ c) \in B_0$.
  - For all $y \in \mathcal{V}$, $k \geqslant 2$ and $c \in \mathbb{N}^*$, $(a,\ y,\ k \cdot c) \in B_0$ iff $\exists! \ell \geqslant 1,\ (a \cdot \ell,\ y,\ k \cdot c) \in B_0$.

▶ **Remark.** If P is not given, that is, if we have a function $P :\ B \to \mathcal{V}_t \cup \{\to\}$ where B is a set of bipositions (*i.e.* $B \subset \mathbb{N}^* \times \mathbb{N}^* \cup \mathbb{N}^* \times \mathcal{V} \times \mathbb{N}^*$), on what condition P is a derivation of t whose support is $A = \{a \in \mathbb{N}^* \mid a \text{ is the underlying pos. of a } \mathtt{b} \in B\}$?
The above conditions adapts well by replacing B_0 by B and A_0 by A and adding the following constraints (mostly on labels):

- $\overline{A} \subset \mathtt{supp}(t)$.
- For all $a \in A$ s.t. $t(\overline{a}) = \lambda x$, $P(a,\ \varepsilon) = \to$ and there is not $k \geqslant 1$ s.t. $a \cdot k \in A$.
- For all $a \in A$ s.t. $\exists x \in \mathcal{V},\ t(\overline{a}) = x$, $C(a)(y)$ is empty for all $y \neq x$ and $\mathtt{Rt}(C(a)(x))$ has exactly one element. Thus, for each $a \in A$ s.t. $t(\overline{a}) = x$, we can still define $\mathtt{tr}(a)$ as the unique k s.t. $\exists c \in \mathbb{N}^*,\ (a,\ x,\ k \cdot c) \in B$.
- For all $a \in A$ s.t. $t(\overline{a}) = @$, $P(a \cdot 1,\ \varepsilon) = \to$ and $a \cdot 0 \notin A$.
- We must have $P(\mathtt{b}) = P(\mathtt{b}')$ for any $\mathtt{b}$ and $\mathtt{b}'$ related in one of the above conditions.

C.3 Meets and Joins of Derivations Families

When P_0, P are two derivations typing the same term, we also write $P_0 \leqslant_\infty P$ to mean that P_0 is the restriction of P on $\texttt{bisupp}(P_0)$. We set $\texttt{Approx}_\infty(P) = \{P_0 \in \texttt{Deriv} \mid P_0 \leqslant_\infty P\}$.

▶ **Lemma 5.** Let $(P_i)_{i\in I}$ be a non-empty family of derivations typing the same term t, such that $\forall i,\, j \in I,\ \exists P \in \texttt{Deriv},\ P_i,\, P_j \leqslant_\infty P$.
We define $P(I)$ by $\texttt{bisupp}(P(I)) = \bigcap_{i\in I} \texttt{bisupp}(P_i)$ and $P(I)(\texttt{b}) = P_i(\texttt{b})$ for any i.
Then, this derivation is correct and the labelled tree $P(I)$ is a derivation (that is finite if one of the P_i is finite). We write $P(I) = \bigwedge_{i\in I} P_i$.

Proof. The proof is done by verifying that $P(I)$ satisfies the characterization of the previous subsection, including Remark C.2. It mostly comes to:
- The correctness of the definition is granted by the upper bound condition.
- The definition $P(I)$ grants proper types and contexts, thanks to subsection C.1 .
- For any $\texttt{b}$ and $\texttt{b}'$ put at stakes in any of the conditions of the previous subsection, $\texttt{b} \in \texttt{bisupp}(P(I))$ iff $\forall i \in I,\ \texttt{b} \in \texttt{bisupp}(P_i)$ iff $\forall i \in I,\ \texttt{b}' \in \texttt{bisupp}(P_i)$ iff $\texttt{b}' \in \texttt{bisupp}(P(I))$.
- The remaining conditions are proven likewise.

◀

▶ **Lemma 6.** Le $(P_i)_{i\in I}$ b a non-empty family of derivations typing the same term, such that $\forall i,\, j \in I,\ \exists P \in \texttt{Deriv},\ P_i,\ P_j \leqslant_\infty P$.
We define the labelled tree $P(I)$ by $\texttt{bisupp}(P(I)) = \bigcup_{i\in I} \texttt{bisupp}(P_i)$ and $P(I)(\texttt{b}) = P_i(\texttt{b})$ for any i s.t. $\texttt{b} \in \texttt{bisupp}(P_i)$.
Then, this definition is correct and $P(I)$ is a derivation (that is finite if I is finite and all the P_i are). We write $P(I) = \bigvee_{i\in I} P_i$.

Proof. The proof is done by verifying that $P(I)$ satisfies the characterization of the previous subsection, as well as for the previous lemma. But here, for any $\texttt{b}$ and $\texttt{b}'$ put at stakes in any of the conditions of the previous subsection, $\texttt{b} \in \texttt{bisupp}(P(I))$ iff $\exists i \in I,\ \texttt{b} \in \texttt{bisupp}(P_i)$ iff $\exists i \in I,\ \texttt{b}' \in \texttt{bisupp}(P_i)$ iff $\texttt{b}' \in \texttt{bisupp}(P(I))$. ◀

The previous lemmas morally define the join and the meet of derivations (under the same derivation) as their set-theoretic union and intersection. More precisely, they entail:

▶ **Proposition 11.** The set of derivations typing a same term t, endowed with $\leqslant_\infty$ is a d.c.p.o. The join of a direct set is given by the above operator.
Moreover, for any derivation P, $\texttt{Approx}(P)$ is a distributive lattice (sometimes empty) and $\texttt{Approx}_\infty(P)$ is a complete distributive lattice, and the meet is given by the above operator.

C.4 Reach of a derivation

▶ **Definition 8.**
- For any derivation P, we set $\texttt{Reach}(P) = \{\texttt{b} \in \texttt{bisupp}(P) \mid \exists\, {}^{\mathrm{f}}P \leqslant P,\ \texttt{b} \in {}^{\mathrm{f}}P\}$.
- If $\texttt{b} \in \texttt{Reach}(P)$, we say $\texttt{b}$ is **reachable**.
- If $B \subset \texttt{bisupp}(P)$, we say B is **reachable** if there is ${}^{\mathrm{f}}P \leqslant P$ s.t. $B \subset \texttt{bisupp}({}^{\mathrm{f}}P)$.

Since $\texttt{Approx}(P)$ is a complete lattice and the bisupports of its elements are finite, we can write $P < \texttt{b} >$ (resp. $P < B >$) for the smallest ${}^{\mathrm{f}}P$ containing $\texttt{b}$ (resp. containing B), for any $\mathtt{b} \in \mathtt{Reach}(P)$ (resp. for any reachable $B \subset \mathtt{bisupp}(P)$).

▶ **Proposition 12.** Let $B \subset \mathtt{bisupp}(P)$. Then B is reachable iff B is finite and $B \subset \mathtt{Reach}(P)$.
In that case, $P < B >= \bigvee_{\mathtt{b} \in B} P < \mathtt{b} >$.

▶ **Definition 9.** If $\mathtt{Reach}(P)$ is non-empty, we define $P < \mathtt{Reach} >$ as the induced labelled tree by P on $\mathtt{Reach}(P)$.

We have actually $P < \mathtt{Reach} >= \bigvee_{\mathtt{b} \in \mathtt{Reach}(P)} P < \mathtt{b} >$, so P is a derivation. By construction, P is approximable.

We can ask ourselves if P is approximable as soon as every biposition at the root is in its reach. It would lead to a reformulation of the approximability condition. We have been unable to answer this question yet.

C.5 Proof of the subject expansion property

We reuse the notations and assumptions of §6.4. We set $A' = \mathtt{supp}(P')$. As mentioned in §4.2, performing an expansion of a term inside demands to choose new axiom tracks. We will do this *uniformly*, *i.e.* we fix an injection $\lfloor \cdot \rfloor$ from $\mathbb{N}^*$ to $\mathbb{N} - \{0, 1\}$ and any axiom rule created at position a will use the axiom track value $\lfloor a \rfloor$.

Assume ${}^{\mathrm{f}}P' \leqslant P'$. Let $N \in \mathbb{N}$ s.t., for all $n \geqslant N$, $b_n \notin \overline{{}^{\mathrm{f}}A'}$ with ${}^{\mathrm{f}}A' = \mathtt{supp}({}^{\mathrm{f}}P')$. For $n \geqslant N$, we write ${}^{\mathrm{f}}P'(n)$ for the derivation replacing t' by t_n in ${}^{\mathrm{f}}P'$. This derivation is correct according to the subject substitution lemma (section 5.3), since $t_n(\overline{a}) = t'(\overline{a})$ for all $a \in {}^{\mathrm{f}}A'$.

Then we write ${}^{\mathrm{f}}P'(n,\ k)$ (with $0 \leqslant k \leqslant n$) the derivation obtained by performing k expansions (w.r.t. our reduction sequence and $\lfloor \cdot \rfloor$). Since b_n is not in A, we observe that ${}^{\mathrm{f}}P'(n+1,\ 1) = {}^{\mathrm{f}}P'(t_n)$. Therefore, for all $n \geqslant N$, ${}^{\mathrm{f}}P'(n,\ n) = {}^{\mathrm{f}}P'(N,\ N)$. Since we could replace N by any $n \geqslant N$, ${}^{\mathrm{f}}P$ is morally ${}^{\mathrm{f}}P'(\infty,\ \infty)$. We write $P = P'(\mathtt{init})$ to refer to this deterministic construction.

We set $\mathcal{D} = \{{}^{\mathrm{f}}P'(\mathtt{init}) \mid {}^{\mathrm{f}}P' \leqslant P'\}$. Let us show that $\mathcal{D}$ is a directed set.

Let ${}^{\mathrm{f}}P_1',\ {}^{\mathrm{f}}P_2' \leqslant P'$. We set ${}^{\mathrm{f}}P' = {}^{\mathrm{f}}P_1' \vee {}^{\mathrm{f}}P_2'$. Let N be great enough so that $\forall n \geqslant N,\ b_n \notin \overline{{}^{\mathrm{f}}A'}$ with ${}^{\mathrm{f}}A' = \mathtt{supp}({}^{\mathrm{f}}P')$.

We have ${}^{\mathrm{f}}P_i' \leqslant {}^{\mathrm{f}}P'$, so ${}^{\mathrm{f}}P_i'(N) \leqslant {}^{\mathrm{f}}P(N)$, so, the by monotonicity of *uniform* expansion, ${}^{\mathrm{f}}P_i'(N,\ N) \leqslant {}^{\mathrm{f}}P'(N,\ N)$, i.e. ${}^{\mathrm{f}}P_i(\mathtt{init}) \leqslant {}^{\mathrm{f}}P(\mathtt{init})$.

Since $\mathcal{D}$ is directed, we can set $P = \bigvee_{{}^{\mathrm{f}}P' \leqslant P'} {}^{\mathrm{f}}P'(\mathtt{init})$. Since for any ${}^{\mathrm{f}}P' \leqslant P$ and the associated usual notation, ${}^{\mathrm{f}}C(\varepsilon) = {}^{\mathrm{f}}C'(\varepsilon)$, ${}^{\mathrm{f}}T(\varepsilon) = {}^{\mathrm{f}}T'(\varepsilon)$ and $C(\varepsilon)$, $C'(\varepsilon)$, $T(\varepsilon)$, $T'(\varepsilon)$ are the respective infinite join of ${}^{\mathrm{f}}C(\varepsilon)$, ${}^{\mathrm{f}}C'(\varepsilon)$, ${}^{\mathrm{f}}T(\varepsilon)$, ${}^{\mathrm{f}}T'(\varepsilon)$ when ${}^{\mathrm{f}}P'$ ranges over $\mathtt{Approx}(P')$, we conclude that $C(\varepsilon) = C'(\varepsilon)$ and $T(\varepsilon) = T'(\varepsilon)$.

We can also prove that different choices of coding functions $\lfloor \cdot \rfloor$ yield *isomorphic* derivation typing t. We start by proving it when P' is finite.

D Approximability of the quantitative NF-derivations

We show in this appendix that every quantitative derivation typing a normal form t is approximable. We use the same notations as in Section 6: we consider a derivation P built as in Subsection 6.2, from a normal form t, a d-support A of t and a type $T(a)$ given for each full position of A. It yields a family of contexts $(C(a))_{a\in A}$ and of types $(T(a))_{a\in A}$ such that $P(a)$ is $C(a)\vdash t|_{\overline{a}} : T(a)$ for all $a\in A$.

D.1 Degree of a position inside a type in a derivation

- For each a in A and each position c in $S(a)$ such that $S(a)(c)\neq X_i$, we define the number $\mathtt{d}_s(c)$ by:

- When a is a full node, $\mathtt{d}_s(c)$ is the applicative depth of a.
- When a is an abstraction position: the value of $\mathtt{d}_s(c)$ for the positions colored in red is the applicative depth of a (and of $\mathring{a}$).

$$E(a)(x_1)\rightarrow E(a)(x_2)\rightarrow\ldots\rightarrow E(a)(x_n)\rightarrow T(\mathring{a})$$

- When a is partial: the value of $\mathtt{d}_s(c)$ for the positions colored in red is the applicative depth of a.

$$F_1(a)\rightarrow\ldots\rightarrow F_k(a)\rightarrow T(\mathring{a})$$

For each $a\in A$ and each position c in $T(a)$, we define the number $\mathtt{d}_s(c)$ (that is the applicative depth of the position a' on which c depends) by extending $\mathtt{d}_s$ via substitution.

There again, for each $a\in A$, each variable x and each position c in $C(a)(x)(c)$ we define $\mathtt{d}_s(c)$ by extending $\mathtt{d}_s$ via substitution.

The definition of $\mathtt{d}_s(c)$ in $T(a)$ and $C(a)(x)$ is sound, because in $E(a)(x)$ and $F_k(x)$, there a no symbol other than the X_i. But the X_i disappear thanks to the coinductive definition of $T(a)$: every position c of $T(a)$ will receive a value for $\mathtt{d}_s(c)$.

- For $c\in\mathbb{N}^*$, we set $s(c)=\max(\ell,\ c_0,\ c_1,\ldots,\ c_{n-1})$ where $n=|a|$ and $\ell=|\{0\leqslant i\leqslant n\mid c_i\geqslant 2\}|$.

For each a in A and each position c in $S(a)$ such that $S(a)(c)\neq X_i$, we define the number $\mathtt{d}_i(c)$ by:

- When a is a full node: $\mathtt{d}_i(c)=s(c)$ (c is a position of $T(a)$).
- When a is an abstraction node: if c is a position colored in blue, $\mathtt{d}_i(c)=0$ and if c is colored in red, $\mathtt{d}_i(c)=s(c')$ (when c' is the position corresponding to c in $T(\mathring{a})$, *i.e.* $c=0^k\cdot c'$)

$$E(a)(x_1)\rightarrow E(a)(x_2)\rightarrow\ldots\rightarrow E(a)(x_k)\rightarrow T(\mathring{a})$$

- When a is partial: if c is a position colored in blue, $\mathtt{d}_i(c)=0$ and if c is colored in red, $\mathtt{d}_i(c)=s(c')$ (when c' is the position corresponding to c in $T(\mathring{a})$, *i.e.* $c=0^k\cdot c'$)

$$F_1(a)\rightarrow\ldots\rightarrow F_k(x)\rightarrow T(\mathring{a})$$

We extend likewise $n(c)$ for inner positions of $T(a)$ or in $C(a)(x)$ via substitution.

▶ **Definition 10.** If c is a position in $T(a)$ or in $C(a)(x)$, the **degree** of c is defined by $\deg c=\max(\mathtt{d}_i(c),\ \mathtt{d}_i(c))$.

D.2 More formally...

For $c \in \mathbb{N}^*$, we set $s(c) = \max(\ell,\ c_0,\ c_1, \ldots,\ c_{n-1})$ where $n = |a|$ and $\ell = |\{0 \leqslant i \leqslant n \mid c_i \geqslant 2\}|$.

For all $a \in A$ and $k \in \mathbb{N}$, we set $S^{\,0}(a) = X_a$ and $S^{k+1}(a) = S^k(a)[S(a')/X_{a'}]_{a' \in \mathbb{N}^*}$.

For all $k \in \mathbb{N}$, we set $\mathtt{supp}^*(S^k(a)) = \{c \in \mathtt{supp}(S^k(a)) \mid S^k(a)(c) \neq X_{a'}\}$.

If $c \in \mathtt{supp}(T(a))$, there is a minimal n s.t. $c \in \mathtt{supp}^*(S^k(a))$. We denote it $\mathtt{cd}(a)(c)$ (call-depth of c at pos. a).

In that case, there are unique $c' \in \mathtt{supp}(T(a))$ and $a' \in A$ s.t. $c' \leqslant c$, $S^{k-1}(a)(c') = X_{a'}$ (we have necessarily $a \leqslant a'$). We write $a' = \mathtt{sp}(a)(c)$ (source position of c at pos. a) and $c' = \mathtt{lcp}(a)(c)$ (last calling position of c at pos. a).
Then, we set $\mathtt{d}_s(a)(c) := \mathrm{ad}(a')$.

With the same notations, $T(a')$ is of the shape $C(a_1)(x_1) \to \ldots C(a_k)(x_k) \to C(\mathring{a}')$ or $\mathrm{R}_1(a') \to \mathrm{R}_2(a') \to \ldots \to \mathrm{R}_k(a') \to T(\mathring{a}')$, where the forest type $\mathrm{R}_k(a')$ is $(\lfloor a_0 \rfloor \cdot T(a_0))_{a_0 \in AP_k(a')}$. There are two cases:

- $c = c' \cdot 0^j$ with $j < k$: we set $n(a)(c) = 0$.
- $c = c' \cdot 0^k : c$" (with $c" \in \mathtt{supp}(T(\mathring{a}'))$): we write $\mathtt{sip}(c) = c$" (source inner position of c at pos. a) and set $\mathtt{d}_i(a)(c) = s(c")$.

▶ **Definition 11.** If c is a position in $T(a)$ or in $C(a)(x)$, the **degree** of c at pos. a is defined by $\deg(a,\ c) = \max(\mathtt{d}_s(c),\ \mathtt{d}_i(c))$.

▶ **Lemma 7.** For all $k \in \mathbb{N}$ and all $a,\ c \in \mathbb{N}^*$, we have $a \in A_n$ and $c \in \mathtt{supp}(S^k_n(a))$ iff $a \in A,\ c \in \mathtt{supp}(S^k(a))$ and $\deg(a,\ c) \leqslant n$.
In that case, $S^k(a)(c) = S^k_n(a)(c)$.

Proof. By a simple but tedious induction on k.

- Case $k = 0$:
  If $a \in A_n$ and $c \in \mathtt{supp}(S^0_n(a))$, then $\mathrm{ad}(a) \leqslant n$ (by def. of A_n) and $c = \varepsilon$. By definition, $\mathtt{d}_s(a)(\varepsilon) = \mathrm{ad}(a) \leqslant n$ and $\mathtt{d}_i(a)(\varepsilon) = 0$. Thus, $\deg(a,\ c) \leqslant n$.
  Conversely, if $c \in \mathtt{supp}(S^0(a))$ and $\deg(a,\ \varepsilon) \leqslant n$, we have likewise $c = \varepsilon$ and $\mathtt{sp}(a)(\varepsilon) = a$, so $\mathrm{ad}(a) \leqslant n$. Thus, $a \in A_n$ and then $c \in S^0_n(a)$.

- Case $k+1$:
  If $a \in A_n$ and $c \in \mathtt{supp}(S^{k+1}_n(a))$, we assume that $a \notin \mathtt{supp}(S^k_n(a))$ (case already handled by IH). We set $a' = \mathtt{sp}_n(a)(c)$ and $c' = \mathtt{lcp}_n(a)(c)$ (thus, $S^k_n(a)(c') = X_{a'}$). By IH, we have also $a' \in A$, $a' = \mathtt{sp}(a)(c)$ and $c' = \mathtt{lcp}(a)(c)$. We have two subcases, depending if $S^{k+1}_n(a)(c) = X_{a"}$ holds or not.

  - If $S^{k+1}_n(a)(c) = X_{a"}$ (with necessarily $a" \in A_n$), then $c = c' \cdot 0^j : \ell$ with $j < \mathtt{rdeg}(a')$ and ℓ integer and, by IH, $c' \in \mathtt{supp}(S^k(a))$ and $S^k(a)(c') = X_{a'}$.
    Then $c = c' \cdot 0^j : \ell \in \mathtt{supp}(S^{k+1}(a))$, $\mathtt{d}_s(a)(c) = \mathrm{ad}(a") \leqslant n$ (since $a" \in A_n$) and $\mathtt{d}_i(a)(c) = 0$. So we have $\deg(a, c) \leqslant n$.

  - If $S^{k+1}_n(a)(c) \neq X_{a"}$ for all a", then $c = c' \cdot 0^j$ with $j < \mathtt{rdeg}(a')$ or $c = c' \cdot 0^{\mathtt{rdeg}(a')} \cdot c$" with $c" \in \mathtt{supp}(T_n(\mathring{a}')) \subset \mathtt{supp}(T_n(\mathring{a}'))$.
    In both cases, $c \in \mathtt{supp}(S^{k+1}(a))$. In the former one, $\mathtt{d}_s(a)(c) = 0$ and in the latter

one, $\mathtt{d}_s(c) = s(c") \leqslant n$ (because $c" \in \mathtt{supp}(T_n(\mathring{a}')))$. Therefore, $\deg(a,\ c) \leqslant n$.

Conversely, if $a \in A,\ c \in \mathtt{supp}(S^{k+1}(a))$ and $\deg(a,\ c) \leqslant n$, we assume that $a \notin \mathtt{supp}(S^k(a))$ (case already handled by IH). We set $a' = \mathtt{sp}(a)(c)$ and $c' = \mathtt{lcp}(a)(c)$ (thus, $S^k(a)(c') = X_{a'}$). By IH, we have also $a,\ a' \in A_n,\ a' = \mathtt{sp}_n(a)(c)$ and $c' = \mathtt{lcp}_n(a)(c)$. Likewise, we have two subcases, according to whether $S^{k+1}(a)(c) = X_{a"}$ or not.

- If $S^{k+1}(a)(c) = X_{a"}$, then, by def. of $\mathtt{d}_i$, we have $\mathtt{d}_i(a)(c) = \mathrm{ad}(a")$, so $\mathrm{ad}(a") \leqslant \deg(a,\ c) \leqslant n$, so $a" \in A_n$. Since $a \leqslant a'$, $a \in A_n$.
  Moreover, $c = c' \cdot 0^j \cdot \ell$ with $j < \mathtt{rdeg}(a')$ and ℓ integer. Since $a" \in A_n$, we have also $c \in S_n^{k+1}(a)$ and $S_n^{k+1}(a)(c) = X_{a"}$.

- If $S_n^{k+1}(a)(c) \neq X_{a"}$ for all $a"$, then $c = c' \cdot 0^j$ with $j < \mathtt{rdeg}(a')$ or $c = c' \cdot 0^{\mathtt{rdeg}(a')} \cdot c"$ with $\mathtt{supp}(T(\mathring{a}'))$.
  Since $\mathtt{d}_s(c) = s(c")$, $s(c") \leqslant n$, so $c" \in \mathtt{supp}(T_n(\mathring{a}'))$. Thus, $c \in \mathtt{supp}(S_n^{k+1}(a))$ and $S_n^{k+1}(a)(c) = S^{k+1}(a)(c)$.

◀

D.3 A complete sequence of derivation approximations

Let n be an integer.

- We set $A_n = \{a \in A \mid \text{the applicative depth of } a \text{ is} \leqslant n\}$.
- We define $T_n(a)$ and $C_n(a)(x)$ by removing all positions c such that $\deg c > n+1$.
- We define the finite labelled tree P_n by $\mathtt{supp}(P_n) = A_n$ and, for each $a \in A_n,\ P(a) = C_n(a) \vdash t|_a : T_n(a)$.

▶ **Proposition 13.** The labelled tree P_n is a finite derivation and $P_n \leqslant P$. It is actually the derivation obtained by the trivial construction w.r.t. T_n and A_n.

Proof. It is a straightaway consequence of lemma 7.
We use the notations of the previous subsection and write $\tilde{T}_n,\ \tilde{C}_n,\ \tilde{P}_n$ for the type, context and derivation obtained by the trivial construction based w.r.t. $(A_n,\ T_n)$.

Since $\tilde{T}_n(a) = S_n(a)[\tilde{T}_n(a')/X_{a'}]_{a' \in \mathbb{N}^*}$, if $c \in \mathtt{supp}(\tilde{T}_n(a))$, there is k s.t. $c \in \mathtt{supp}(S_n^k(a))$ and $S_n^k(a)(c) = \tilde{T}_n(a)$. By lemma 7, we have also $c \in \mathtt{supp}(S^k(a))$, $\deg(a,\ c) \leqslant n$ and $S^k(a)(c) = S_n^k(a)(c)$. Thus, $T_n(a)(c) = \tilde{T}_n(a)(c)$. Conversely, we show likewise that if $c \in \mathtt{supp}(T_n(a))$, then $c \in \mathtt{supp}(\tilde{T}_n(a))$.

Thus, for all $a \in A_n$, $\tilde{T}_n(a) = T_n(a)$. It also entails that $C_n(a)(x) = \tilde{C}_n(a)(x)$ for all $a \in A_n$ and $x \in \mathcal{V}$. ◀

▶ **Corollary 1.** The derivation P is approximable.

Proof. Let ${}^0B \subset \mathtt{bisupp}(P)$ be a finite set. Let n be the maximal degree of a biposition of B. Then, ${}^0B \subset \mathtt{bisupp}(P_n)$ and $P_n \leqslant P$ is finite. ◀

▶ **Corollary 2.**
- Each normal form $t \in \Lambda^{001}$ admits a approximable and unforgetful derivation.
- Every quantitative derivation typing a normal form is approximable.

Proof.
- Comes from the previous corollary and Proposition 7.
- Comes from the previous corollary and Proposition 8.

◀

E Isomorphisms between rigid derivations

Let P_1 and P_2 be two rigid derivations typing the same term t. We write A_i, C_i, T_i for their respective supports, contexts and types.

A **derivation isomorphism** ϕ from P_1 to P_2 is given by:

- $\phi_{\mathtt{supp}}$, 01-tree isomorphism from A_1 to A_2.
- For each $a_1 \in A_1$:
  - A type isomorphism $\phi_{a_1} : T_1(a_1) \to T_2(\phi_{\mathtt{supp}}(a_1))$
  - For each $x \in \mathcal{V}$, a forest type isomorphism $\phi_{a_1|x} : C_1(a_1)(x) \to C_2(\phi_{\mathtt{supp}}(a_1))(x)$

such that the following "rules compatibility" conditions hold:

- If $t(\overline{a_1}) = \lambda x$, then:
  - $\phi_{a_1}(1 \cdot c) = 1 \cdot \phi_{a_1 \cdot 0}(c)$ and $\phi_{a_1}(k \cdot c) = \phi_{a_1 \cdot 1|x}(k \cdot c)$ for any $k \geqslant 2$ and $c \in \mathbb{N}^*$
  - $\phi_{a_1|y} = \phi_{a_1 \cdot 0|y}$ for any $y \in \mathcal{V}$, $y \neq x$.
- If $t(\overline{a_1}) = @$:
  - $\phi_{a_1}(c) = \mathtt{Tl}(\phi_{a_1 \cdot 1}(1 \cdot c))$, for any $c \in \mathbb{N}^*$, where $\mathtt{Tl}(k \cdot c) = c$ (removal of the first integer in a finite sequence).
  - $\phi_{a_1|x} = \bigcup_{\ell \geqslant 1} \phi_{a_1 \cdot \ell}$ (the functional join must be defined because of the app-rule).

The above rules means that ϕ must respect different occurrences of the "same" (from a moral point of view) biposition. For instance:

- Assume $t(\overline{a_1}) = \lambda x$, then $T_1(a_1) = C_1(a_1 \cdot 0)(x) \to T_1(a_1 \cdot 0)$. So, any inner position c_1 inside $T(a_1 \cdot 0)$ can be "identified" to the inner position $1 \cdot c_1$ inside $T_1(a_1)$. Thus (forgetting about the indexes), if ϕ maps c_1 on c_2 (inside $T_2(a_2)$, then ϕ should map $1 \cdot c_1$ on $1 \cdot c_2$.
- Assume $t(\overline{a_1}) = @$. Then, the forest type $C(a_1)(x)$ if the union of the $C(a_1 \cdot \ell)(x)$ (for ℓ spanning over $\mathbb{N} - \{0\}$). Then ϕ should map every inner position $k \cdot c$ inside $C(a_1)(x)$ according to the unique $C(a_1 \cdot \ell)$ to which it belong.

▶ **Lemma 8.** *If $P_1 \xrightarrow{b} P_1'$, $P_2 \xrightarrow{b} P_2'$, then $P_1 \equiv P_2$ iff $P_1' \equiv P_2$.*

Proof. Let $\alpha_1' \in A_1'$. We set $\alpha_1 = \mathtt{Res}_b^{-1}(\alpha_1')$, $\alpha_2 = \phi_{\mathtt{supp}(\alpha_1)}$, $\alpha_2' = \mathtt{Res}_b(\alpha_2)$ ($\mathtt{Res}_b$ is meant w.r.t. P_1 or P_2 according to the cases). Then we set $\phi'_{\mathtt{supp}} = \mathtt{Res}_b \circ \phi_{\mathtt{supp}} \circ \mathtt{Res}_b^{-1}$. Thus, $\alpha_2' = \phi'_{\mathtt{supp}}(\alpha_1')$.

We set $\phi'_{\alpha_1'} = \phi_{\alpha_1}$. Observing the form of $C_1(\alpha_1)(y)$ (for $y \neq x$) given in Subsection 4.1, we set $\phi'_{\alpha_1'|y} = \phi_{\alpha_1|y} \cup \bigcup_{k \in K} \phi_{a(k)|x}$ with $K = \mathtt{AxTr}(\alpha_1, x, k)$ and $a(k) = \mathtt{pos}(a_1, x, k)$. ◀

Notice that ϕ' is defined deterministically from ϕ.

▶ **Proposition 14.** If P_1 and P_2 are isomorphic and type the term t (we do not assume them to be approximable), $t \to^\infty t'$, yielding two derivation P_1', P_2 according to section 5.3, then P_1' and P_2' are also isomorphism.

Proof. We reuse all the hypotheses and notations of section 5.3 and we consider an isomorphism $\phi : P_1 \to P_2$.

For all $n \in \mathbb{N}$, let P_1^n, P_2^n and ϕ^n be the derivations and derivation isomorphisms obtained after n steps of reduction from P_1, P_2 and ϕ. Let $\alpha_1' \in A_1'$ and $N \in \mathbb{N}$ such that, for all $n \geqslant N$, $|b_n| > |\alpha_1'|$. But then, for any $n \geqslant N$, $C_i^n(\alpha')(x) = C_i'\alpha')(x)$, $T_i'(\alpha') = T_i^n(\alpha')$. So we can set $\phi'_{\mathtt{supp}}(\alpha_1) = \phi^n_{\mathtt{supp}}(\alpha_1)$, $\phi'_{\alpha'} = \phi^N_{\alpha'}$. ◀

F An Infinitary Type System with Multiset Constructions

F.1 Rules

We present here a definition of type assignment system $\mathcal{M}$, which is an infinitary version of De Carvalho's system $\mathcal{M}_0$.

If two (forest) types U_1 and U_2 are isomorphic, we write $U_1 \equiv U_2$. The set $\mathtt{Types}_{\mathcal{M}}$ is the set $\mathtt{Types} / \equiv$ and the set $\mathcal{M}(\mathtt{Types})$ of multiset types is defined as $\mathtt{FTypes} / \equiv$.

If U is a forest or a rigid type, its equivalence class is written $\overline{U}$. The equivalent class of a forest type F is the multiset type written $[\overline{F_{|k}}]_{k \in \mathtt{Rt}(F)}$ and the one of the rigid type $F \to T$ is the type $\overline{F} \to \overline{T}$. If α is a type variable, $\overline{\alpha}$ is written simply α (instead of $\{\alpha\}$). It defines coinductively the multiset style writing of $\overline{U}$.

Countable sum $\sum_{i \in I} \overline{F^i}$ is defined on $\mathcal{M}(\mathtt{Types})$ by using a bijection j from the pairwise disjoint countable sum $\coprod_{\mathbb{N}} \mathbb{N} - \{0, 1\}$ to $\mathbb{N} - \{0, 1\}$, replacing I by a part of $\mathbb{N}$ and each root k of F^i by the integer given by $j(i, k)$ (so that the equivalence classes are preserved).

A $\mathcal{M}$-context is a function from the set of term variables $\mathcal{V}$ to the set $\mathtt{Types}_{\mathcal{M}}$. The set of $*$-derivations, written $\mathtt{Deriv}_*$ is defined coinductively by the following rules:

$$\dfrac{}{x : [\tau] \vdash x : \tau \ (\text{at } \varepsilon)}\ \text{ax} \qquad \dfrac{\dfrac{P'}{\Gamma \vdash t : \tau \ (\text{at } 0)}}{\Gamma - x \vdash \lambda x.t : \Gamma(x) \to \tau \ (\text{at } \varepsilon)}\ \text{abs}$$

$$\dfrac{\dfrac{P'}{\Gamma \vdash t : [\sigma_i]_{i \in I} \to \tau \ (\text{at } 1)} \qquad \left(\dfrac{P'_k}{\Delta_i \vdash u : \sigma_i \quad (\text{at } k_i)} \right)_{i \in I}}{\Gamma + \sum_{i \in I} \Delta_i \vdash t(u) : \tau \ (\text{at } \varepsilon)}\ \text{app}$$

In the app-rule, the k_i must be pairwise distinct integers $\geqslant 2$.

Let P_1 and P_2 be two $*$-derivations. A $*$-isomorphism from P_1 to P_2 is a 01-labelled isomorphism from P_1 to P_2 and the set $\mathtt{Deriv}_{\mathcal{M}}$ is defined by $\mathtt{Deriv}_* / \equiv$.

From now on, we write $\mathtt{Types}$ and $\mathtt{Deriv}$ instead of $\mathtt{Types}_{\mathcal{M}}$ and $\mathtt{Deriv}_{\mathcal{M}}$. An element of $\mathtt{Deriv}$ is usually written Π, whereas an element of $\mathtt{Deriv}_*$ is written P. Notice the derivation Π and Π' of Subsection 2.2 are objects of $\mathtt{Deriv}$.

F.2 Quantitativity and Coinduction

Let Γ be any context. Using the infinite branch of f^ω, we notice we can give the following variant of derivation Π' (subsection 2.2), which still respects the rules of system $\mathcal{M}$:

$$\dfrac{\dfrac{}{f : [[\alpha] \to \alpha] \vdash f^\omega : [\alpha] \to \alpha}\ \text{ax} \qquad \dfrac{\Pi'_\Gamma}{f : [[\alpha] \to \alpha]_{n \in \omega} + \Gamma \vdash f^\omega : \alpha}}{f : [[\alpha] \to \alpha]_{n \in \omega} + \Gamma \vdash f^\omega : \alpha}\ \text{app}$$

If, for instance, we choose the context Γ to be $x : \tau$, from a quantitative point of view, the variable x (that is not in the typed term f^ω) should not morally be present in the context. We have been able to "call" the type τ by the mean of an infinite branch. Thus, we can enrich the type of any variable in any part of a derivation, as long it is below an infinite branch (neglecting the bound variables). It motivates the following definition:

▶ **Definition 12.** ▬ A $*$-derivation P is **quantitative** if, for all $a \in \mathtt{supp}(P)$, $\Gamma(a)(x) = [\tau(a')]_{a' \in \mathtt{Ax}(a)(x)}$.
▬ A derivation Π is quantitative if any of its $*$-representatives is (in that case, all of them are quantitative).

In the next subsection, we show that a derivation Π from system $\mathcal{M}$ can have both quantitative and unquantitative representatives in the rigid framework. It once again shows that rigid constructions allow a more fine-grained control than system $\mathcal{M}$ does on derivations.

F.3 Representatives and Dynamics

A rigid derivation P (with the usual notations C, t, T) **represents** a derivation Π if the $*$-derivation P_* defined by $\mathtt{supp}(P_*) = \mathtt{supp}(P)$ and $P_*(a) = \overline{C(a)} \vdash t|_{\overline{a}} : \overline{T(a)}$, is a representative of Π. We write $P_1 \overset{\mathcal{M}}{\equiv} P_2$ when P_1 and P_2 both represent the same derivation Π.

▶ **Proposition 15.** If a rigid derivation P is quantitative, then the derivation $\overline{P}$ (in system $\mathcal{M}$) is quantitative.

Proposition 8 makes easy to prove that:

▶ **Proposition 16.** If Π is a quantitative derivation typing a normal form, then, there is a quantitative rigid derivation P s.t. $\overline{P} = \Pi$.

Proof. Let $P(*)$ be a $*$-derivation representing Π. We set $A = \mathtt{supp}(P(*))$ and for all full position $a \in A$, we choose a representative $T(\mathring{a})$ of $\tau(a)$. We apply then the special construction, which yields a rigid derivation P such that $P_* = P(*)$ (we show that, for all $a \in A$, $T(a)$ represents $\tau(a)$). ◀

We can actually prove that every quantitative derivation can be represented with a quantitative rigid derivation and that we can endow it with every possible infinitary reduction choice ([13]). However, a quantitative derivation can also have an unquantitative rigid representative (see below Π' and $\tilde{P}'$).

Actually, whereas $P_1 \equiv P_2$ (Subsection 3.3) means that P_1 and P_2 are isomorphic in every possible way, $P_1 \overset{\mathcal{M}}{\equiv} P_2$ is far weaker: we explicit in this subsection big differences in the dynamical behaviour between two rigid representatives of the derivations Π and Π' of Subsection 2.2.

We omit the right side of axiom rules, *e.g.* $f : ((2 \cdot \alpha) \to \alpha)_2$ stands for $f : ((2 \cdot \alpha) \to \alpha)_2 \vdash f : (2 \cdot \alpha) \to \alpha$.

- Let P_k $(k \geqslant 2)$ and P be the following rigid derivations:

$$P_k = \dfrac{\dfrac{}{f : ((2\cdot\alpha)\to\alpha)_k \vdash \text{ (tr. 1)}} \qquad \dfrac{\dfrac{}{x : (\gamma)_2 \vdash x : \gamma \text{ (tr. 1)}} \qquad \left(\dfrac{}{x : (\gamma)_i \vdash x : \gamma \text{ (tr. } i-1)}\right)_{i \geqslant 4}}{x : (\gamma)_{i \geqslant 2} \vdash xx : \alpha \text{ (tr. 2)}}}{\dfrac{f : ((2\cdot\alpha)\to\alpha)_k \vdash f(xx) : \alpha \text{ (tr. 0)}}{f : ((2 : \alpha)\to\alpha)_k \vdash \Delta_f : \gamma}}$$

$$P = \dfrac{P_2 \text{ (tr. 1)} \qquad (P_k \text{ (tr. } k-1))_{k \geqslant 3}}{f : ((2 : \alpha)\to\alpha)_{k \geqslant 2} \vdash \Delta_f \Delta_f}$$

- Let $\tilde{P}_k$ $(k \geqslant 2)$ and $\tilde{P}$ be the following rigid derivations:

$$\tilde{P}_k =$$

$$\dfrac{\dfrac{\dfrac{f : ((2\cdot\alpha)\to\alpha)_k \ \text{(tr. 1)}}{} \qquad \dfrac{\dfrac{}{x : (\gamma)_3 \vdash x : \gamma \ \text{(tr. 1)}} \quad \dfrac{}{x : (\gamma)_2 \vdash x : \gamma \ \text{(tr. 2)}} \quad \left(\dfrac{}{x : (\gamma)_i \vdash x : \gamma \ \text{(tr. } i-1\text{)}}\right)_{i\geqslant 4}}{x : (\gamma)_{i\geqslant 2} \vdash xx : \alpha \ \text{(tr. 2)}}}{f : ((2\cdot\alpha)\to\alpha)_k \vdash f(xx) : \alpha \ \text{(tr. 0)}}}{f : ((2\cdot\alpha)\to\alpha)_k \vdash \Delta_f : \gamma}$$

$$\tilde{P} = \quad \dfrac{\tilde{P}_2 \ \text{(tr. 1)} \qquad \left(\tilde{P}_k \ \text{(tr. } k-1\text{)}\right)_{k\geqslant 3}}{f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 2} \vdash \Delta_f\Delta_f}$$

- The rigid derivations P and $\tilde{P}$ both represent Π. Morally, subject reduction in P will consist in taking the first argument P_3, placing it on the first occurrence of x in $f(xx)$ (in P_2) and putting the other P_k ($k \geqslant 4$) in the different axiom rules typing the second occurrence of x in the same order. There is a simple decrease on the track number and we can go this way towards f^ω.

The rigid derivation $\tilde{P}$ process the same way, except it will always skip $\tilde{P}_3$ ($\tilde{P}_3$) will stay on track 2). Morally, we perform subject reduction "by-hand" while avoiding to ever place P_3 in head position.

The definitions of section 5.3 show that infinitary reductions performed in P and $\tilde{P}$ yield respectively to P' and $\tilde{P}'$ below.

$$P' =$$

$$\dfrac{\dfrac{}{f : ((2\cdot\alpha)\to\alpha)_2 \ \text{(tr. 1)}}\,\text{ax} \qquad \dfrac{\dfrac{}{f : ((2\cdot\alpha)\to\alpha)_3 \ \text{(tr. 1)}}\,\text{ax} \qquad \begin{matrix} P' \\ \vdots \\ f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 4} \vdash f^\omega : \alpha \ \text{(tr. 2)} \end{matrix}}{f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 3} \vdash f^\omega : \alpha \ \text{(tr. 2)}}\,\text{app}}{f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 2} \vdash f^\omega : \alpha}\,\text{app}$$

$$\tilde{P}' =$$

$$\dfrac{\dfrac{}{f : ((2\cdot\alpha)\to\alpha)_2 \ \text{(tr. 1)}}\,\text{ax} \qquad \dfrac{\dfrac{}{f : ((2\cdot\alpha)\to\alpha)_4 \ \text{(tr. 1)}}\,\text{ax} \qquad \begin{matrix} \tilde{P}' \\ \vdots \\ f : ((2\cdot\alpha)\to\alpha)_{k=3\vee k\geqslant 5} \vdash f^\omega : \alpha \ \text{(tr. 2)} \end{matrix}}{f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 4} \vdash f^\omega : \alpha \ \text{(tr. 2)}}\,\text{app}}{f : ((2\cdot\alpha)\to\alpha)_{k\geqslant 2} \vdash f^\omega : \alpha}\,\text{app}$$

Thus, P' and $\tilde{P}'$ both represent Π' (from subsec. 2.2), but P' is quantitative whereas $\tilde{P}'$ is not (the track 3 w.r.t. f does not end in an axiom leaf). Thus, quantitativity is not stable under s.c.r.s.

Moreover, it is easy to check that P and P' approximable (reuse the finite derivations of Subsection 2.2). Thus, Π and Π' have both approximable and not approximable approximations. It provides a new argument for the impossibility of formulating approximability in system $\mathcal{M}$.