Every λ-Term is Meaningful for the Infinitary Relational Model

Pierre Vial
Inria (LS2N CNRS)
Équipe Gallinette
Nantes, France

Abstract

Infinite types and formulas are known to have really curious and unsound behaviors. For instance, they allow to type Ω, the auto-autoapplication and they thus do not ensure any form of normalization/productivity. Moreover, in most infinitary frameworks, it is not difficult to define a type R that can be assigned to *every* λ-term. However, these observations do not say much about what coinductive (*i.e.* infinitary) type grammars are able to provide: it is for instance very difficult to know what types (besides R) can be assigned to a given term in this setting. We begin with a discussion on the expressivity of different forms of infinite types. Then, using the resource-awareness of sequential intersection types (system S) and tracking, we prove that infinite types are able to characterize the *arity* of every λ-terms and that, in the infinitary extension of the relational model, every term has a "meaning" *i.e.* a non-empty denotation. From the technical point of view, we must deal with the total lack of guarantee of productivity for typable terms: we do so by importing methods inspired by first order model theory.

Keywords Curry-Howard, coinductive types, sequence types, order, relational model, non-productive reduction

1 Introduction (Infinite types)

1.1 Some semantical aspects of infinite types

It is well-known that the mere fact of allowing infinite formulas gives birth to unsound/contradictory proof systems. For instance, let A be *any* formula. We then define the infinite formula F_A by $F_A := (((\dots) \to A) \to A) \to A$ *i.e.* $F_A = F_A \to A$ (the letter "F" stands for "fixpoint"). The formula F_A gives rise both to a proof of A and—*via* the Curry-Howard correspondence—to the typing of a term with A, this term being no other than the auto-autoapplication $\Omega := \Delta\,\Delta$ (with $\Delta = \lambda x.x\,x$). This is given by Fig. 1.

Thus, every type A is inhabited by Ω. But given a λ-term t, what types A does t inhabit? A first observation is that every λ-term can easily be typed: let us just define R (standing for "reflexive") by $R = R \to R$. Thus, $R = (R \to R) \to (R \to R) = \dots$ Then, it is very easy to type every term with R. In the inductive steps below, Γ denotes a context that assigns R to every variable of its domain:

$$\frac{}{\Gamma; x{:}R \vdash x{:}R} \qquad \frac{\Gamma; x{:}R \vdash t{:}R}{\Gamma \vdash \lambda x.t{:}R{\to}R\ (=R)} \qquad \frac{\Gamma \vdash t{:}R\ (=R{\to}R) \quad \Gamma \vdash u{:}R}{\Gamma \vdash t\,u : R}$$

Therefore, every λ-term inhabits the type R. Yet, this does not answer the former question: what types does a term t inhabit? As we will see, this question has no simple answer and we will chiefly focus on one aspect of this problem, namely, the typing constraints caused by the *arity* of the λ-terms. Intuitively, the **arity of a λ-term** t (sometimes called its **order** *e.g.*, in [4]) is the supremal n such that $t \to_\beta^* \lambda x_1 \dots x_n.u$ (for some term u) *i.e.* the number of abstractions that one can output from t. For instance, the arity Ω of 0 (it is a **zero term**), the arity of the **head normal form (HNF)** $\lambda x_1 x_2.x\,u_1\,u_2\,u_3$ (with u_1, u_2, u_3 terms) is 2 and the term $\mathtt{Y}_\lambda := (\lambda x.\lambda y.xx)\lambda x.\lambda y.xx$ has an infinite arity, because $\mathtt{Y}_\lambda \to_\beta^n \lambda y_1 \dots \lambda y_n.\mathtt{Y}_\lambda$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}F_A \vdash x{:}F_A}\,\mathsf{ax} \quad \dfrac{}{x{:}F_A \vdash x{:}F_A}\,\mathsf{ax}}{x : F_A \vdash x\,x : A}\,\mathsf{app}}{\vdash \lambda x.x\,x : F_A \to A}\,\mathsf{abs} \quad \dfrac{\dfrac{\dfrac{}{x{:}F_A \vdash x{:}F_A}\,\mathsf{ax} \quad \dfrac{}{x{:}F_A \vdash x{:}F_A}\,\mathsf{ax}}{x : F_A \vdash x\,x : A}\,\mathsf{app}}{\vdash \lambda x.x\,x : F_A}\,\mathsf{abs}}{\vdash \Omega : A}\,\mathsf{app}$$

Figure 1. Typing Ω,inferring A

The **arity of a type** is the number of its top-level arrows *e.g.*, if o_1, o_2 are *type atoms* (or *type variables*), $o_1 \to o_1$, o_1, $o_1 \to o_2 \to o_1$ and $(o_1 \to o_2) \to o_1$ are of respective arities 1, 0, 2, 1. Via Curry-Howard, the constructor λx corresponds to the introduction of an implication, and so, in most type systems, a typed term of the form $\lambda x_1 \dots x_n.u$ is typed with an arrow of arity $\geqslant n$. For instance, if $\lambda x.\lambda y.u$ is typed, then it is so with a type of the form $A \to B \to C$.

Moreover, if a type system satisfies **subject reduction**, meaning that typing is stable under reduction, the above observation entails that, if a typable term is of arity n, then it is typable only with types of arity $\geqslant n$ (the arity of a term is a lower bound for the arity of its possible types). Equivalently, if t is typed with B, then the arity of B *statically* gives an upper bound to the arity of t (static meaning without reduction). A finite *type* has a finite arity (whereas the finite *term* $\mathtt{Y}_\lambda$ has an infinite arity). Yet, unsurprisingly, an infinite type may have an infinite arity *e.g.*, R defined by $R = R \to R$ above. This confirms that the typing of any term t with R is trivial and does not bring any information, since the arity of a term is of course $\leqslant \infty$. However, the facts that, by allowing infinite types, (1) one can type every term with $R = R \to R$ and (2) one can type Ω with any type A, do not mean that finding the types that can be assigned to a given term t is an easy problem in this setting.

1.2 Intersection Types and Arity

We have just seen that subject reduction naturally ensures that simple typing provides an upper bound to the arity of a typed term. Intersection type systems (i.t.s.), introduced by Coppo-Dezani [8], generally satisfy **subject expansion**, meaning that typing is stable under anti-reduction. Those systems feature a type constructor $\wedge$ (intersection) and are designed to ensure equivalences of the form "t is typable iff t is normalizing" and also provide *semantical* proofs of non-type-theoretic properties such as "t is weakly normalizing iff the leftmost-outermost reduction strategy terminates on t" [16]. From subject expansion and the typing of **normal forms (NF)** *i.e.* terminal states, i.t.s. are actually able to *capture* the arity of *some*

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209133

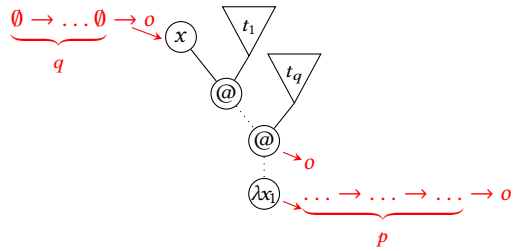


Figure 2. Typing a Head Normal Form in an i.t.s.

λ-terms. For instance, if an i.t.s. characterizes **head normalization (HN)**, then, every HN term t of arity p is typable with a type whose arity is also *equal* to p (and not only bounded below by p).

Let us informally explain why and how the arity of the typable terms is usually captured by i.t.s. For instance, i.t.s. characterizing HN usually feature arrow types having an empty source[1] (that we generically denote by $\emptyset$), meaning that the underlying functions do not look at their argument. Namely, if $t : \emptyset \to B$, then $t\,u$ is typable with B for *any* term u. This allows us to easily type any HNF while capturing its arity: in Fig. 2, one just assigns $\emptyset \to \ldots \to \emptyset \to o$ (arity q) to the head variable x, so that $x\,t_1 \ldots t_q$ is typed with the type atom o and the HNF $\lambda x_1 \ldots x_p.x\,t_1 \ldots t_q$, whose arity is p, is typed with an arrow type of arity p. Then, by subject expansion, one concludes that every HN term of arity p is typable with a type of arity p. The same argument can be adapted to i.t.s. characterizing weak (including infinitary weak) or strong normalization, which also usually capture the arity of their typable terms.

1.3 In Search for Infinite Denotations

Independently from normalization properties, another important facet of i.t.s. is that they also provide **denotational models** for the λ-calculus *i.e.* they associate to each λ-term a denotation $[\![t]\!]$ (usually, $[\![t]\!]$ is a morphism in a category), meaning that $[\![t]\!]$ is invariant under β-conversion (*i.e.* $t_1 =_\beta t_2$ implies $[\![t_1]\!] = [\![t_2]\!]$). For instance, the typing judgments of Gardner-de Carvalho's system $\mathscr{R}_0$ (that we will shortly discuss in § 2.1) correspond to the points of the relational model [5]: indeed, $\{\rhd_{\mathscr{R}_0}\Gamma \vdash t : \tau \mid \Gamma, \tau\} = [\![t]\!]_{\mathtt{rel}}$ for any term t, where the left-hand side corresponds to the set of derivable judgments of system $\mathscr{R}_0$ typing t and the right-hand side, the denotation of t in the relational model. Thus, infinite types are a mean to study the infinitary extension of the relational model.

The (finitary) relational model only gives a (non-empty) denotation to HN terms. This reflects the fact that non-HN (equivalently, **unsolvable**) terms have an infinitary behavior w.r.t. head reduction. In this regard, it is natural to seek whether such terms have an infinitary semantics, since infinitary models bring information on asymptotic aspects of terms *e.g.*, in the recent work of Grellois-Melliès [12, 13]. The first main contribution of this article is to prove that every term has a non-empty interpretation in the infinitary relational model $\mathscr{M}_{\mathtt{rel}}$. This furthers the approach of Curry, aiming at finding more and more "meanings" to λ-terms (see *e.g.*, [8] or § 5.4. on "illative systems" in [7]). One may thus consider that, in $\mathscr{M}_{\mathtt{rel}}$, every λ-term is meaningful (although we do not have yet a deep understanding of this model, beyond arity-related aspects).

An interesting aspect of models is that they allow us to statically discriminate terms from one another, meaning that if $[\![t_1]\!] \neq [\![t_2]\!]$ then $t_1 \neq_\beta t_2$ *i.e.* two terms that do not have the same denotation do not represent two different states of a same program. For instance, the i.t.s. that are able to assign a type of arity n to any (*e.g.*, HN) term of arity n (but not to one of arity $n+1$) can be regarded as **arity-discriminating** for HN terms. This holds for system $\mathscr{R}_0$. We prove that the infinitary extension of system $\mathscr{R}_0$, that we denote $\mathscr{R}$, is arity-discriminating for all λ-terms (not just the HN ones). This second main contribution of the paper (Theorem. 2) extends a feature of system $\mathscr{R}_0$ concerning HN terms to the whole λ-calculus.

1.4 Stability and the Difficulty of Infinitary Typing

We saw above how i.t.s. capture the arity of the typed terms in finite/productive case (productive cases include infinitary normalizing terms). Let us now understand why the method of § 1.2 fails while studying full infinitary typing. Fig. 2 shows well that typing in a given i.t.s. (and in particular, capturing the arity) reduces to typing the "partial" normal forms (*e.g.*, HNF or β-NF). Intuitively, the nodes corresponding to the normalized parts of a term cannot be affected by reduction *e.g.*, the nodes labelled with λx_i, @ and x in Fig. 2 (the "spine" of the HNF). Such nodes are **stable**. In contrast, some terms (the so-called **mute terms** [3]) do not ever give rise to stable positions and are thus totally *unproductive e.g.*, the term Ω.

Thus, there is no clear way to capture the arity of any typable term: the example of Ω shows that the case of totally unstabilizable term must be handled when considering infinite types. Note that Ω is just an example of a mute term, that happens to satisfy the nice fixpoint equation $\Omega \to_\beta \Omega$ and has a simple parsing tree. This partially explains why Ω was easily typable. In general, there is no method to type generic mute terms that do not satisfy an equation.

1.5 Infinitary Typing and Klop's Problem

Let us say a few words about the questions raised by infinitary typing in the non-idempotent intersection type framework *i.e.* by the interpretation of terms in the infinitary relational model.

- One of the fundamental interests of non-idempotent intersection ($A \wedge A \neq A$) is that, in this setting, a type is a **resource** that cannot be duplicated or merged/contracted and that is possibly *consumed* under reduction.
- Moreover, non-idempotent i.t.s. are often **relevant**, meaning that weakening is not allowed.

An i.t.s. that forbids duplication and weakening can be qualified as **linear**, which is the case of system $\mathscr{R}_0$ that we hinted at in § 1.3, and its infinitary version $\mathscr{R}$. As we will see in § 2.1, relevance disables the argument proving that every term is typable with ρ, the non-idempotent counterpart of the type R considered above. However, while trying to characterize a form of infinitary weak normalization, we noticed in [17] that Ω is also typable in $\mathscr{R}$. We recovered soundness by defining a validity criterion, discarding degenerate typing derivations, which was possible by introducing a **rigid** variant of system $\mathscr{R}$, namely system S. System S has many nice features *e.g.*, *tracking* (motivated in § 2.2).

Still, these observations raise the problem of characterizing the set of typable terms (without validity criterion) in the coinductive relevant and non-idempotent framework. In particular, is every term $\mathscr{R}$-typable? Or is there a term t that is not $\mathscr{R}$-typable? Observe that such a term t would not be linearizable, even in an infinite way (since a $\mathscr{R}$/S-derivation typing t induces a linear representation of t). The existence of non-linearizable terms would be very surprising. Therefore, it must be investigated. Since our main theorem

[1] In this article, we put aside *non-strict* types systems (*e.g.*, system $\mathcal{D}\Omega$ [8, 16]), in which types are less constrained by the arity of terms.

(Theorem 1) states that system $\mathscr{R}$ actually types every term, we actually prove that every term is linearizable, as expected.

Note again that the method described at the end of § 1.2 does not work for non-normal terms: naively, when $x\,u$ occurs in t, we would like to assign to x a type of the form $A \to B$, where A is the type of u, and proceed by induction. However, x may be substituted in the course of a reduction sequence, and so, typing constraints on x are not easily readable. Thus, in the productive case, in the purpose of proving that the terms of a given set (*e.g.*, the set of HN terms above) are typable, we escape this problem by typing normal forms (*e.g.*, HNF) and then proceeding by expansion. But, by § 1.4, this cannot work when we want to type every term.

To sum up, due to full resource-awareness (including relevance), typing in the coinductive systems $\mathscr{R}$ and S is intrinsically non-trivial. But the same reason (full resource-awareness) make linear intersection type systems the good framework to study the expressive power of infinite types and to capture the arity of every λ-term. Thus, besides our first goal…

Goal 1. *Capturing the arity of every λ-term with infinite types.*

…we have now a second one, narrowly related to the first:

Goal 2. *Proving that every term is $\mathscr{R}$-typable.*

1.6 A Technical Contribution

We have not addressed yet the way we can study unproductive/mute terms, despite the fact that all the known techniques of intersection type theory fail: we propose a solution to overcome this difficulty, inspired by (a simplified form of) first order model theory, that we mix with techniques specific to the λ-calculus. This is our main technical innovation, since it enables the study of *unproductive* reduction. Thus, beyond the relational model, this work proposes the first use of first order model theory to study an infinitary extension of a finitary model of the λ-calculus and to generalize properties coming from the finite model to every λ-term (*e.g.*, capturing their arity). The proof that every term is $\mathscr{R}$-typable has three main stages: (1) reducing the problem to a set of *stability relations* (§ 3) (2) describing the possible *interactions* between these relations (§ 4.2) (3) describing a procedure of partial (but more importantly *finite*) normalization (§ 5). The same ingredients allow us to capture the arity. The italicized words, as well as the tools of the proof and how they arise, are gradually explained in the paper but we refer to § 3.1, 3.2 and the introduction of § 4 for some high-level input.

1.7 Outline

To sum up, our **main contributions** in this article consist of (1) proving that every term has a non-empty denotation in the infinitary relational model (2) the arity of terms can be captured by infinitary type systems (3) introducing a method giving rise to semantic descriptions of λ-terms, whether they normalize or not.

In § 2, we present two non-idempotent i.t.s. and the notions of relevance and tracking. In § 3, we explain why describing the "form" of the derivations in system S (presented in § 2.2) may help us to prove that every term is S-typable and we characterize these forms. The key notion of **thread** is defined. In § 4, we define, for a given term t, a **nihilating chain** as a proof that t is *not* S-typable. Proving that every term is typable reduces to proving that nihilating chains do not exist (Prop. 2). We prove that, under a positivity condition involving threads, nihilating chains indeed do not exist (Prop. 3). In § 5, we prove that the positivity condition can be assumed without loss of generality, by means of a finite normalization procedure. Thus, no proof of untypability exists and every term is typable. We conclude in § 6 by notably sketching the adaptation of the previous steps giving the expected type-theoretic arity-capture.

2 Infinitary Relevant and Non-Idempotent Intersection

2.1 System $\mathscr{R}$

We now define more formally system $\mathscr{R}$, the *coinductive* version of the finite system $\mathscr{R}_0$, independently introduced by Gardner and de Carvalho [10, 11]. See [6] for a general presentation of $\mathscr{R}_0$. System $\mathscr{R}$ is of good help to understand relevant intersectionbut, as we shall see in § 2.2 and 3.1, it is unfit to express the techniques yielding Theorems 1 and 2, and we refer to it only for heuristic purposes.

The set of $\mathscr{R}$-types is coinductively defined by.

$$\sigma,\ \tau ::= o \in \mathscr{O} \mid [\sigma_i]_{i\in I} \to \tau$$

We call $\mathcal{I} := [\sigma_i]_{i\in I}$ a **multiset type**. The multiset types represent intersection in system $\mathscr{R}_0$ and the intersection operator $\wedge$ is the multiset-theoretic sum: $\wedge_{i\in I}\mathcal{I}_i = +_{i\in I}\mathcal{I}_i$ (*i.e.* $\wedge_{i\in I}[\sigma^i_j]_{j\in J(i)} := +_{i\in I}[\sigma^i_j]_{j\in J(i)}$). We assume I to be countable, the empty multiset type is denoted by $[\,]$ and $[\sigma]_\omega := [\sigma_i]_{i\in\omega}$ with $\forall i \in \omega,\ \sigma_i = \sigma$.

An $\mathscr{R}$-**context** (metavariables Γ, Δ) is a *total* function from $\mathscr{V}$ (the set of term variables) to the set of multiset types. The **domain of** Γ is given by $\{x \mid \Gamma(x) \neq [\,]\}$. The intersection of contexts $+_{i\in I}\Gamma_i$ is defined point-wise. We may write $\Gamma;\Delta$ instead of $\Gamma + \Delta$ when $\mathtt{dom}(\Gamma) \cap \mathtt{dom}(\Delta) = \emptyset$. Given a multiset type $[\sigma_i]_{i\in I}$, we write $x : [\sigma_i]_{i\in I}$ for the context Γ s.t. $\Gamma(x) = [\sigma_i]_{i\in I}$ and $\Gamma(y) = [\,]$ for all $y \neq x$. An $\mathscr{R}$-**judgment** is a triple $\Gamma \vdash t : \sigma$ where Γ is an $\mathscr{R}$-context, t a term and σ an $\mathscr{R}$-type.

The set of $\mathscr{R}$-derivations is defined *inductively* by:

$$\frac{}{x : [\tau] \vdash x : \tau}\ \mathsf{ax} \qquad \frac{\Gamma; x : [\sigma_i]_{i\in I} \vdash t : \tau}{\Gamma \vdash \lambda x.t : [\sigma_i]_{i\in I} \to \tau}\ \mathsf{abs}$$

$$\frac{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau \quad (\Delta_i \vdash u : \sigma_i)_{i\in I}}{\Gamma + (+_{i\in I}\Delta_i) \vdash t\,u : \tau}\ \mathsf{app}$$

As announced in § 1.5, system $\mathscr{R}$ is not only non-idempotent, but also *relevant*. For instance, the K-term $\lambda x.y$ can only be assigned types of the form $[\,] \to \tau$. Indeed, $\lambda x.y$ can only by typed by:

$$\frac{\dfrac{}{x : [\tau] \vdash x : \tau}\ \mathsf{ax}}{x : [\tau] \vdash \lambda y.x : [\,] \to \tau}\ \mathsf{abs}$$

This comes from the fact that y does not occur in x, and thus, by relevance, the constructor λy cannot invoke a (non-empty) type on the left-hand side of the arrow type.

Fig. 1 can adapted and we can type Ω with τ for all $\mathscr{R}$-types τ, by just defining ϕ_τ by $\phi_\tau = [\phi_\tau]_\omega \to \tau$. However, defining ρ by $\rho = [\rho]_\omega \to \rho$ does not allow us to type every term with ρ in system $\mathscr{R}$. To understand why, note first that relevance can be disabled, by replacing ax par $\mathsf{ax}_\mathtt{w}$:

$$\frac{i_0 \in I}{\Gamma; x : [\sigma_i]_{i\in I} \vdash x : \sigma_{i_0}}\ \mathsf{ax}_\mathtt{w}$$

We call $\mathscr{R}_\mathtt{w}$ the type system thus obtained. Then, the proof on p. 1 can be adapted to $\mathscr{R}_\mathtt{w}$, by considering only contexts $\Gamma, \Gamma_t, \Gamma_u$ assigning $[\rho]_\omega$ to all the variables in their domains:

$$\frac{}{x:[\rho]_\omega \vdash x:\rho}\,\mathsf{ax} \qquad \frac{\Gamma; x:[\rho]_\omega \vdash t:\rho}{\Gamma \vdash \lambda x.t : \rho}\,\mathsf{abs} \qquad \frac{\Gamma_t \vdash t:\rho \quad (\Gamma_u \vdash u:\rho)_\omega}{\Gamma_t + \Gamma_u \vdash t\,u : \rho}\,\mathsf{app}$$

Thus, every term is $\mathscr{R}_{\mathsf{w}}$-typable. But note that, by relevance, this proof by induction on the structure of t fails for $\mathscr{R}$. For instance, if x is not in t, $\Gamma \vdash t : \rho$ yields $\Gamma \vdash \lambda x.t : [\,] \to \rho$ ($\neq \rho$!) and since the empty multiset type may occur in unpredictable places in a term, finding an $\mathscr{R}$-typing of any term t is non-trivial. In some sense, $\mathscr{R}$-typability is about capturing the way relevance constrains emptiness to occur! But since a variable x can be substituted under reduction as observed in § 1.5, [ ] may occur in unpredictable places.

2.2 Towards Tracking and Sequential Intersection

Unfortunately, resource-awareness of system $\mathscr{R}$ is not enough to process the proof techniques to be developed here: we also need **tracking**. Let us just give an example to show what the impossibility of tracking means:

$$\frac{\dfrac{}{x:[[\sigma,\sigma]\to\tau]\vdash x:[\sigma,\sigma]\to\tau}\,\mathsf{ax} \quad \dfrac{}{x:[\sigma]\vdash x:\sigma}\,\mathsf{ax} \quad \dfrac{}{x:[\sigma]\vdash x:\sigma}\,\mathsf{ax}}{x : [[\sigma,\sigma] \to \tau, \sigma, \sigma] \vdash x\,x : \tau}\,\mathsf{app}$$

In this derivation, in the context $x : [[\sigma] \to \tau, \sigma, \sigma]$ of the conclusion, one cannot know which particular axiom rule, each red occurrence of σ comes from: there is no possible notion of **pointer** with multiset intersections, which is one thing that we absolutely need to capture the key notion of **support candidate** in § 3.1.

Tracking can be retrieved while keeping most of system $\mathscr{R}_0$'s nice features (*e.g.*, syntax-direction) by considering system S, that we introduced in [17]. System S uses **sequence types** instead of **multiset types** to represent intersection. For instance, instead of having a cardinal 3 intersection $[o, o', o]$, system S considers a cardinal 3 sequence $(2 \cdot o, 5 \cdot o', 8 \cdot o)$. Sequences come along with a disjoint union operator *e.g.*, $(2 \cdot o, 5 \cdot o', 8 \cdot o) = (2 \cdot o, 5 \cdot o') \uplus (8 \cdot o)$: in this equality, the occurrence of o in the left-hand side annotated with 2 unambiguously comes from that which is also annotated with 2 in the right-hand side. We call these annotations **tracks**. In contrast, $[o, o', o] = [o, o'] + [o]$, but there is no way to unambiguously associate to an occurrence of o in the left-hand side the one of $[o, o']$ or the one of $[o]$ in the right-hand side.

Formally, the set of S-types is defined coinductively by:

$$\begin{array}{rcl} T,\, S_k & ::= & o \parallel F \to T \\ F & ::= & (k \cdot S_k)_{k\in K}\ (K \subseteq \mathbb{N} \setminus \{0,1\}) \end{array}$$

The empty sequence type is denoted ( ) and we often write $(S_k)_{k\in K}$ instead of $(k \cdot S_k)_{k\in K}$. The set of top-level tracks of a sequence type is called its set of **roots** and we write *e.g.*, $\mathtt{Rt}(F) = \{2, 5, 8\}$ when $F = (2 \cdot S, 5 \cdot S', 8 \cdot S)$. Note that the disjoint union operator can lead to **track conflict** *e.g.*, , if $F_1 = (2 \cdot o, 3 \cdot o')$ and $F_2 = (3 \cdot o', 8 \cdot o)$, the union $F_1 \uplus F_2$ is not defined, since $\mathtt{Rt}(F_1) \cap \mathtt{Rt}(F_2) = \{3\} \neq \emptyset$.

An S-context C (or D) is a total function from $\mathscr{V}$ to the set of S-types. The operator $\uplus$ is extended point-wise. An S-judgment is a triple $C \vdash t : T$, where C, t and T are respectively an S-context, a term and T an S-type. A **sequence judgment** is a family of judgments $(k \cdot (D_k \vdash u : S_k))_{k\in K}$ (with $K \subseteq \mathbb{N} \setminus \{0,1\}$) that all type the same term u, often just written $(D_k \vdash u : S_k)_{k\in K}$. For instance, if $5 \in K$, then the judgment on track 5 is $C_5 \vdash u : S_5$.

The set of S-derivations is defined inductively by:

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

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

The app-rule can be applied only if there is no track conflict in the context $C \uplus (\uplus_{k\in K} D_k)$. In an ax-rule concluding with $x : (k \cdot T) \vdash x : T$, the track k is called the **axiom track** of this axiom rule. We refer to § III and IV of [17] for additional examples and figures for all what concerns the basics of system S sketched here and thereafter.

Let $S_{\mathsf{ex}} = (2 \cdot o, 7 \cdot o') \to o''$. To gain space, we write $k \vdash x : T$ (with $k \geqslant 2$, $x \in \mathscr{V}$, T S-type) instead of $x : (k \cdot T) \vdash x : T$ in ax-rules. We also indicate the track of argument derivations between *square* brackets *e.g.*, $x : (3 \cdot o)$ [5] means that the argument judgment $x : (3 \cdot o) \vdash x : o$ is on track 5):

$$\dfrac{\dfrac{\dfrac{}{3 \vdash y : S_{\mathsf{ex}}}\,\mathsf{ax} \quad \dfrac{}{3 \vdash x : o\ [5]}\,\mathsf{ax} \quad \dfrac{}{9 \vdash x : o'\ [6]}\,\mathsf{ax}}{x : (3 \cdot o, 9 \cdot o'),\ y : (3 \cdot S_{\mathsf{ex}}) \vdash y\,x : o''}\,\mathsf{app}}{y : (3 \cdot (2 \cdot o, 7 \cdot o') \to o'') \vdash \lambda x.y\,x : (3 \cdot o, 9 \cdot o') \to o''}\,\mathsf{ax}$$

As expected:

Property 1. *Systems* S *and* $\mathscr{R}$ *enjoy subject reduction and expansion*

If tracks are erased, a sequence becomes a multiset and S-derivations collapse on $\mathscr{R}$-derivations (*e.g.*, P_{ex} on P_{ex}), so that an S-typable term is also $\mathscr{R}$-typable. We may thus replace Goal 2 by Goal 3.

Goal 3. *Proving that every term is* S*-typable.*

We reduce the problem (*i.e.* proving that a term t is typable in system S) into a first order theory, that we call $\mathcal{T}_t$. We actually prove that $\mathcal{T}_t$ indeed captures the problem, by means of a proposition that can be interpreted as a (simplified) completeness theorem (see Corollary 1): we show that if $\mathcal{T}_t$ is coherent, then t is S-typable. Then we prove that $\mathcal{T}_t$ is coherent for all terms t. Go to the introduction of § 3.2 and 4 to have a closer descriptions of the proof of the coherence of $\mathcal{T}_t$ and of its main stages.

2.3 Parsing, Pointing

In this technical section, we explain how we may point inside an S-type or an S-derivation, thanks to tracking. We define the support of an S-type and an S-derivation, and also the key notions of biposition and bisupports. Let $\mathbb{N}^*$ denote the set of the finite words on $\mathbb{N}$, the operator $\cdot$ denotes concatenation, ε the empty word and $\leqslant$ the prefix order *e.g.*, $2 \cdot 1 \cdot 3 \cdot 7 \in \mathbb{N}^*$, $2 \cdot 1 \leqslant 2 \cdot 1 \cdot 3 \cdot 7$. Moreover, the **collapse** $\overline{k}$ of a track k is defined by $\overline{k} = \min(k, 2)$. This notation is extended letter-wise on $\mathbb{N}^*$ *e.g.*, $\overline{0 \cdot 5 \cdot 1 \cdot 3 \cdot 2} = 0 \cdot 2 \cdot 1 \cdot 2 \cdot 2$. The **support of term** is defined by induction as expected: $\mathsf{supp}(x) = \{\varepsilon\}$, $\mathsf{supp}(\lambda x.t) = \{\varepsilon\} \cup 0 \cdot \mathsf{supp}(t)$ and $\mathsf{supp}(t\,u) = \{\varepsilon\} \cup 1 \cdot \mathsf{supp}(t) \cup 2 \cdot \mathsf{supp}(u)$. If $a \in \mathbb{N}^*$ and $\overline{a} \in \mathsf{supp}(t)$, we denote by $t|_a$ the subterm of t rooted at position $\overline{a}$ whereas $t(a)$ is the constructor (@, x or λx) of t at position a *e.g.*, $t|_0 = y\,x$ and $t(0 \cdot 1) = y$ with $t = \lambda x.y\,x$.

The **support of a type** (resp. **a sequence type**), which is a tree of $\mathbb{N}^*$ (resp. a forest), is defined by mutual coinduction: $\mathsf{supp}(o) = \{\varepsilon\}$, $\mathsf{supp}(F \to T) = \{\varepsilon\} \cup \mathsf{supp}(F) \cup 1 \cdot \mathsf{supp}(T)$ and $\mathsf{supp}((T_k)_{k\in K}) = \cup_{k\in K} k \cdot \mathsf{supp}(T_k)$ *e.g.*, $\mathsf{supp}(S_{\mathsf{ex}}) = \{\varepsilon, 1, 2, 7\}$. We can define the **support** of a derivation $P \rhd C \vdash t : T$: $\mathsf{supp}(P) = \varepsilon$ if P is an axiom rule, $\mathsf{supp}(P) = \{\varepsilon\} \cup 0 \cdot \mathsf{supp}(P_0)$ if $t = \lambda x.t_0$ and P_0 is the subderivation typing t_0, $\mathsf{supp}(P) = \{\varepsilon\} \cup 1 \cdot \mathsf{supp}(P_1) \cup_{k\in K} k \cdot \mathsf{supp}(P_k)$ if $t = t_1\,t_2$, P_1 is the left subderivation typing t_1 and P_k the subderivation typing t_2 on track k. The P_k ($k \in K$) are called **argument derivations**. For instance, $\mathsf{supp}(P_{\mathsf{ex}}) = \{\varepsilon, 0, 0 \cdot 1, 0 \cdot 5, 0 \cdot 6\}$, $P(0 \cdot 1) = y : (3 \cdot S_{\mathsf{ex}}) \vdash y : S_{\mathsf{ex}}$ and $P(0 \cdot 6) = x : (9 \cdot o') \vdash x : o'$.

Choice function for axiom tracks: Note that, to define an S-derivation typing a term t (thus fulfilling Goal 3), one must one choose an axiom track in every axiom rule so that no conflict arise. In this short article, let us just say that we can escape this problem by resorting to an *arbitrary* injection from $\mathbb{N}^*$ to $\mathbb{N} \setminus \{0,1\}$, that chooses axiom tracks for us: we say that an S-derivation P is a $\lfloor\cdot\rfloor$-derivation if $P(a) = (x : (k \cdot T) \vdash x : T)$ (*i.e.* a is the position of an ax-rule typing x in P), then $k = \lfloor a \rfloor$. If t is $\lfloor\cdot\rfloor$-typable, then t is in particular S-typable, so that we now replace Goal 3 with Goal 4:

Goal 4. *Given an injection $\lfloor\cdot\rfloor : \mathbb{N}^* \to \mathbb{N} \setminus \{0,1\}$, proving that every term is $\lfloor\cdot\rfloor$-typable.*

The injectivity hypothesis allows us to stop bothering about track conflict any further while achieving Goal 4. It is also w.r.t. the function $\lfloor\cdot\rfloor$ that we will capture where emptiness occurs (see § 3.2).

We explain now how to point *inside* types *nested* in S-derivations, or to axioms typing a given variable, and formalize the associated pointers. If P is an S-derivation and $a \in \mathsf{supp}(P)$, then the judgment at position a is denoted $\mathsf{C}^P(a) \vdash t|_a : \mathsf{T}^P(a)$ *e.g.*, $\mathsf{C}^{P_{\mathsf{ex}}}(0{\cdot}6) = x : (9{\cdot}o')$ and $\mathsf{T}^{P_{\mathsf{ex}}}(0 \cdot 6) = o'$. Let P be an S-derivation. A **right biposition** is a pair of the form (a, c), where $a \in \mathsf{supp}(P)$ and $c \in \mathsf{supp}(\mathsf{T}^P(a))$, we write $\mathtt{bisupp}(P)$ for the **(right) bisupport** of P *i.e.* its set of (right) bipositions. If $(a, c) \in \mathtt{bisupp}(P)$, then $P(a, c)$ denotes $\mathsf{T}^P(a, c)$ *e.g.*, $P_{\mathsf{ex}}(0{\cdot}6, \varepsilon) = o'$ and $P_{\mathsf{ex}}(0 \cdot 1, \varepsilon) = o$, $P_{\mathsf{ex}}(0 \cdot 1, 1) = o''$ and $P_{\mathsf{ex}}(\varepsilon, 9) = o'$. Note that, contrary to [17], we only consider right bipositions. For this article, we think a biposition as type symbol ($o \in \mathscr{O}$ or $\to$) nested in a given S-derivation P and we often use this heuristic identification implicitly, most notably when describing Fig. 4.

Assume that P types t. We set $A = \mathsf{supp}(P)$ and $B = \mathtt{bisupp}(P)$. If $x \in \mathscr{V}$, $a \in A$, we set $\mathsf{Ax}^P_a(x) = \{a_0 \in A \mid a \leqslant a_0,\ t(a) = x, \nexists a'_0,\ a \leqslant a'_0 \leqslant a_0,\ t(a'_0) = \lambda x\}$ (occurrences of x in P above a, that are not bound w.r.t. a) *e.g.*, $\mathsf{Ax}^{P_{\mathsf{ex}}}_{\varepsilon}(x) = \emptyset$ (x is bound at the root), but $\mathsf{Ax}^{P_{\mathsf{ex}}}_{0}(x) = \{0 \cdot 5, 0 \cdot 6\}$ (x is not bound at position 0). Technically, this notation is crucial to harness relevance (see polar inversion, § 3.3) but the important thing to remember is that, thanks to tracking, in system S, one can unambiguously designate the axiom rules typing the variable of a λx.

2.4 Typing some Notable Terms in System $\mathscr{R}$

We now use system $\mathscr{R}$ to type a few terms satisfying fixpoint equations. Some of them are not head normalizing. Let $\Delta_f = \lambda x.f(x\,x)$, $\mathsf{Y} = \lambda f.\Delta_f\Delta_f$ (Y is *Curry fixpoint combinator*). Moreover, if $I = \lambda x.x$ and $K = \lambda xy.x$, then $\mathsf{Y}\,I \to \Omega$ (satisfying $\Omega \to_\beta \Omega$), $\mathsf{Y}\,f \to \mathsf{Y}_f := \Delta_f\,\Delta_f$ (satisfying $\mathsf{Y}_f \to_\beta f(\mathsf{Y}_f)$) and $\mathsf{Y}\,K \to_\beta \mathsf{Y}_\lambda := (\lambda x.\lambda y.xx)\lambda x.\lambda y.xx$ (satisfying $\mathsf{Y}_\lambda \to_\beta \lambda y.\mathsf{Y}_\lambda$).

Iterating reduction on Y_f and Y_λ infinitely many times, we see that Y_f (resp. Y_λ) *strongly converges* to the infinitary term $f^\omega := f(f(\ldots))$ (resp. $\lambda y.\lambda y.\ldots$) in the sense of [9, 15]. Thus, Ω and Y_f are both zero terms (§ 1.1) and Y_λ a term of infinite arity. The term Ω is actually a *mute term* (see § 1.4) and Y_f is a term whose Böhm tree (see [1], chapter 10) f^ω does not contain $\bot$.

Because of rule abs and subject reduction, a term of arity n may only be typed with a type of arity $\geqslant n$, as explained in § 1.1. But some $\mathscr{R}$-derivations can capture more precisely the arity of terms: for each $\mathscr{R}$-type τ, we define coinductively ϕ_τ by $\phi_\tau = [\phi_\tau]_\omega \to \tau$ and we consider the following typing of Y (omitting left-hand sides of ax-rules):

$$\Pi_{\Delta_f} = \dfrac{\dfrac{}{f : [\tau] \to \tau}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{x : \phi_\tau}\,\mathsf{ax} \quad \left(\dfrac{}{x : \phi_\tau}\,\mathsf{ax}\right)_\omega}{x : [\phi_\tau]_\omega \vdash x\,x : \tau}\,\mathsf{app}}{f : [[\tau] \to \tau]; x : [\phi_\tau]_\omega \vdash f(x\,x) : \tau}\,\mathsf{app}}{f : [[\tau] \to \tau] \vdash \Delta_f : \phi_\tau\ (= [\phi_\tau]_\omega \to \tau)}\,\mathsf{abs}$$

$$\Pi_{\mathsf{Y}} = \dfrac{\dfrac{\Pi_{\Delta_f} \qquad (\Pi_{\Delta_f})_\omega}{f : [[\tau] \to \tau]_\omega \vdash \Delta_f\,\Delta_f : \tau}\,\mathsf{app}}{\vdash \mathsf{Y} : [[\tau] \to \tau]_\omega \to \tau}\,\mathsf{abs}$$

Thus, Y is $\mathscr{R}$-typable with $[[\tau] \to \tau]_\omega \to \tau$ for *any* type τ.

Using suitable instances or variants of Π_{Y}, we can build $\Pi_\Omega \rhd \vdash \Omega : \tau$ (for any τ) and $\Pi_\lambda \rhd \vdash \mathsf{Y}_\lambda : [\,] \to [\,] \to \ldots$ By instantiating τ with a type variable o, we get $\vdash \Omega : o$ and $\vdash \mathsf{Y}_f : o$. Thus, the zero terms Ω and Y_f are typed with types of arity 0 whereas Y_λ (whose arity is infinite) is typed with a type of infinite arity, as it was constrained to be.

We will generalize this result (not only for terms built from a fixpoint combinator like Ω or $\lambda x.\Omega$) and show that, for all *pure* terms t of arity n, there is an $\mathscr{R}$-derivation typing t with a type of arity n (Theorem 2).

3 Bisupport Candidates

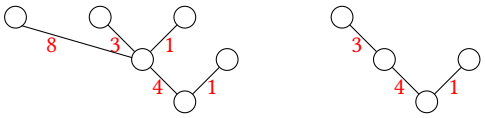
In this section, we characterize, for a given term t, its *bisupport candidates i.e.* the (potential) forms of the derivations typing t. By "form", we intuitively mean a set of unlabelled positions (that must be stable under some suitable relations). We make explicit that idea by studying first the possible forms of an S-type in § 3.1. The notion of unlabelled position has a meaning only because tracks of S allow us to define suitable pointers. It would be impossible in system $\mathscr{R}$.

3.1 A Toy Example: Support Candidates for Types

In this section, we explain how the notion of "form" of a support can be formalized by giving a characterization of the supports of S-types in terms of stability conditions.

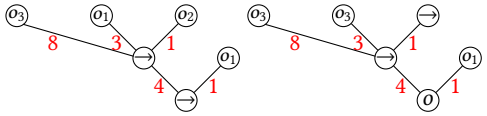
The definition of a particular S-type T can be understood as a two-step process: first, we choose the support $C := \mathsf{supp}(T)$, next, we choose the type labels $T(c)$ (in the signature $\mathscr{O} \cup \{\to\}$) given to the positions $c \in C$. However, not all the subsets $C \subseteq \mathbb{N}^*$ are fit to be the support of a type, and not all the possible decorations of a suitable set C yield a correct type.

For instance, let us consider the two sets of positions C_1 and C_2 below. Do they define the supports of some types T_1 and T_2?



$C_1 = \{\varepsilon,\ 1,\ 4,\ 4{\cdot}1,\ 4{\cdot}3,\ 4{\cdot}8\}$ $\quad C_2 = \{\varepsilon,\ 1,\ 4,\ 4{\cdot}3\}$

As it turns out, C_1 is the support of a type *e.g.*, $(4 \cdot (8 \cdot o_3, 3 \cdot o_1) \to o_2) \to o_1$ (figure below). By contrast, no type T may satisfy $\mathsf{supp}(T) = C_2$, because a non-terminal node of a type (necessarily an arrow) should have a child on track 1 (by convention, its right-hand side), but $4 \in C_2$ and $4 \cdot 1 \notin C_2$.



Type $(4{\cdot}(8{\cdot}o_3,\ 3{\cdot}o_1) \to o_2) \to o_1$ $\quad$ Wrong decoration

This motivates the following notion: a **support candidate** of type is a subset $C \subseteq \mathbb{N}^*$ such that there exists a type T satisfying $C = \mathsf{supp}(T)$. Given a support candidate C, it is easy to define a correct type whose support is C:

- The label of non-terminal nodes of C should be arrows.
- the leaves of C should be decorated with type atoms $o \in \mathscr{O}$.

So was done for the decoration on the left-hand side, representing the type $(4 \cdot (8 \cdot o_3, 3 \cdot o_1) \to o_2) \to o_1$. In contrast, the decoration on the right-hand side is incorrect: ε (non-terminal) is labelled with $o \in \mathscr{O}$ and $4 \cdot 1$ (leaf) with $\to$.

The observations about C_1 and C_2 above suggest considering two relations $\to_{\mathtt{t1}}$ and $\to_{\mathtt{t2}}$ defined by:

- For all $c \in \mathbb{N}^*$, $k \in \mathbb{N}$, $c \cdot k \to_{\mathtt{t1}} c$.
- For all $c \in \mathbb{N}^*$, $k \geqslant 2$, $c \cdot k \to_{\mathtt{t2}} c \cdot 1$.

A set of positions C is closed under $\to_{\mathtt{t1}}$ (*i.e.* $c_1 \in C$ and $c_1 \to_{\mathtt{t1}} c_2$ entails $c_2 \in C$) iff it is a tree. Stability under condition $\to_{\mathtt{t2}}$ means that if a node c is not terminal, then it has a child on track 1.

Lemma 1. *Let $C \subseteq \mathbb{N}^*$. Then C is a type support candidate (i.e. there exists a type T s.t. $C = \mathsf{supp}(T)$) iff C is non-empty and is closed under $\to_{\mathtt{t1}}$ and $\to_{\mathtt{t2}}$.*

Thus, relations $\to_{\mathtt{t1}}$ and $\to_{\mathtt{t2}}$ are enough to characterize support candidates. We call them **stability relations** *e.g.*, the good candidate support C_1 is stable under $\to_{\mathtt{t1}}$ and $\to_{\mathtt{t2}}$, whereas $4 \cdot 3 \in C_2$, $4 \cdot 3 \to_{\mathtt{t2}} 4 \cdot 1$ but $4 \cdot 1 \notin C_2$, so that the bad candidate support C_2 is not stable under $\to_{\mathtt{t2}}$.

When $c_1 \to_{\mathtt{t1}} c_2$ or $c_1 \to_{\mathtt{t2}} c_2$, we say that c_1 **subjugates** c_2, because c_1 demands c_2 to ensure a correct formation of the support.

3.2 Toward the Characterization of Bisupport Candidates

For the remainder of this paper, we fix an injection $\lfloor\cdot\rfloor : \mathbb{N}^* \to \mathbb{N} \setminus \{0, 1\}$. By Goal 4, we want to prove that every term t is $\lfloor\cdot\rfloor$-typable. By analogy with the notion of candidate supports for types (previous section), the idea is to characterize the **bisupport candidates** for the $\lfloor\cdot\rfloor$-derivations typing a given term t *i.e.* sets $B \subseteq \mathbb{N}^* \times \mathbb{N}^*$ s.t. there exists a $\lfloor\cdot\rfloor$-derivation P typing t satisfying $B = \mathtt{bisupp}(P)$ (Prop. 1 to come). We proceed by defining in § 3:

- $\mathbb{B}^t$, the set of the potential bipositions of a derivation typing a term t (in this Section 3.2).
- On $\mathbb{B}^t$, we define a relation $\to_\bullet$ (which is actually the union of 7 stability relations). More precisely:
  - There is a special constant symbol $\mathsf{p}_\bot$ in $\mathbb{B}^t$, that roughly indicates "untypability" or "emptiness".
  - The term t is typable iff there is a non-empty subset B of $\mathbb{B}^t$, such that B is stable under $\to_\bullet$ and does *not* contain $\mathsf{p}_\bot$ (compare this statement with Lemma 1). Such a B is the support of a derivation typing t. This equivalence is given by the "completeness-like" statement of Corollary 1.

Let us now define $\mathbb{B}^t$ by first noticing that not every position $a \in \mathbb{N}^*$ (or biposition $(a, c) \in \mathbb{N}^* \times \mathbb{N}^*$) may be in a derivation typing a given term t. For instance, we have $\mathsf{supp}(\lambda x.y\, x) = \{\varepsilon,\ 0,\ 0{\cdot}1,\ 0{\cdot}2\}$, so, if P types $\lambda x.x\, x$, then $a \in \mathsf{supp}(P)$ implies $\overline{a} = \varepsilon,\ 0,\ 0 \cdot 1$ or $0 \cdot 2$ *i.e.* $\overline{\mathsf{supp}(P)} \subseteq \{\varepsilon,\ 0,\ 0 \cdot 1,\ 0 \cdot 2\}$. For instance, $\mathsf{supp}(P_{\mathsf{ex}}) = \{\varepsilon,\ 0,\ 0 \cdot 1,\ 0 \cdot 2,\ 0 \cdot 5,\ 0 \cdot 6\}$. More generally, if t is a term, we set $\mathbb{A}^t = \{a \in \mathbb{N}^* \mid \overline{a} \in \mathsf{supp}(t)\}$ and $\mathbb{B}^t = (\mathbb{A}^t \times \mathbb{N}^*) \cup \{\mathsf{p}_\bot\}$ (where $\mathsf{p}_\bot$ is an "empty biposition" constant), so that, if P is a $\lfloor\cdot\rfloor$-derivation typing t, then a position (resp. a biposition) of P must be in $\mathbb{A}^t$ (resp. in $\mathbb{B}^t \setminus \{\mathsf{p}_\bot\}$) *i.e.* $\mathsf{supp}(t) \subseteq \mathbb{A}^t$ and $\mathtt{bisupp}(P) \subset \mathbb{B}^t \setminus \{\mathsf{p}_\bot\}$.

The constant $\mathsf{p}_\bot$ roughly materializes emptiness and will be used to describe how "relevance-related emptiness" is constrained to occur in $\lfloor\cdot\rfloor$-derivations (see polar inversion in § 3.3).

We omit P and t from some notations. We set $\mathbb{A}_a(x) = \{a_0 \in \mathbb{A} \mid a \leqslant a_0,\ t(a_0) = x,\ \nexists a_0',\ a \leqslant a_0' < a_0,\ t(a_0') = \lambda x\}$. Thus, with the notation $\mathtt{Ax}_a^P$ of § 2.3, if P is a $\lfloor\cdot\rfloor$-derivation, then $\mathtt{Ax}_a^P(x) \subset \mathbb{A}_a(x)$ for all $a \in \mathsf{supp}(P)$, $x \in \mathscr{V}$ and $\mathbb{A}_a(x)$ may be considered as the set of position candidates for $\mathtt{ax}$-rules typing x above a.

Now, remember that the function $\lfloor\cdot\rfloor$ has been fixed to choose axiom tracks for us (§ 2.3): if x a variable and a_0 an axiom position candidate for x (*i.e.* $t(a_0) = x$), then a potential $\lfloor\cdot\rfloor$-derivation P containing a_0 has an axiom of the form $P(a_0) = x : (k \cdot T) \vdash x : T$ with $k = \lfloor a_0 \rfloor$. Thus, if $t(a) = \lambda x$ and we set $\mathtt{Tr}_\lambda(a) = \{\lfloor a_0 \rfloor \mid a_0 \in \mathbb{A}_{a \cdot 0}(x)\}$, then $\mathtt{Tr}_\lambda(a)$ is the set of axiom tracks dedicated to x above the $\mathtt{abs}$-rule at position a by the function $\lfloor\cdot\rfloor$. It is interesting in that, *e.g.*, if $t(a) = \lambda x$ and $8 \notin \mathtt{Tr}_\lambda(a)$, then we can assert that, if there exists a $\lfloor\cdot\rfloor$-derivation P and $a \in \mathsf{supp}(P)$, then $P(a) = (S_k)_{k \in K} \to T$ with $8 \notin K$. Indeed, by definition of $\mathtt{Tr}_\lambda(a)$, there is no axiom position candidate a_0 for x above a whose axiom track is 8.

Thus, when a variable x is not at some places in t, $\lfloor\cdot\rfloor$ constrains emptiness to "occur" at some particular tracks if we perform an abstraction λx, and then, we informally say *e.g.*, that **emptiness occurs on track 8 w.r.t. position** a under the above assumption. This give us more fine-grained information about occurrences of emptiness in a derivation typing t: system $\mathtt{S}$ (enriched with $\lfloor\cdot\rfloor$) will provide us information about emptiness *track by track*. This is precisely what we need to understand typability in the relevant and non-idempotent framework (remember § 1.5) in that, we have to ensure that emptiness does not compromise typability: intuitively, emptiness must not propagate everywhere in the derivations typing a given term t. If it did, a derivation typing t would be empty (*i.e.* t would not be typable) and we want to show that this does not happen, in the purpose of proving that every term is typable in $\mathtt{S}$

3.3 Tracking a Type in a Derivation

Let us now express the stability conditions (as in § 3.1) that a $\lfloor\cdot\rfloor$-bisupport candidate for a derivation typing t should satisfy. We will need to ensure the points below:

- Identification of the components (*i.e.* the bipositions) of a same type T in a derivation from bottom to top (see Fig. 3): relation of **ascendance** $\to_{\mathtt{asc}}$.
- Identification of the components of type given in an $\mathtt{ax}$-rule to a variable x (S_5 in Fig. 3) and its occurrence called by the abstraction λx: relation of **polar inversion** $\to_{\mathtt{pi}}$.
- Identification of the matching components of the types of u and v in the $\mathtt{app}$-rule typing $u\, v$ (types S_k in the $\mathtt{app}$-rule of Fig. 3): relation of **consumption** $\to$.
- Correct type formation, as in Sec. 3.1: extensions of relations $\to_{\mathtt{t1}}$ and $\to_{\mathtt{t2}}$.
- The type of a subterm of the form $\lambda x.u$ is an arrow type (and not a type variable): relation $\to_{\mathtt{abs}}$.

By lack of space, most of the proofs are omitted for the remainder of the paper and we can only give a few details on the concepts that we use. We refer to the webpage of the author, or Chapters 11 and 12 of [18] for all the details and more examples and heuristics.

In Fig. 3, we indicate the *position* of a judgment between *angle* (≠square) brackets *e.g.*, $C; x : (S_k)_{k \in K} \vdash t : T\ \langle a \cdot 0 \rangle$ means that judgment $C; x : (S_k)_{k \in K} \vdash t : T$ is at position $a \cdot 0$. We denote by $\mathtt{pos}$

Abstraction rule

$$\dfrac{\dfrac{}{x : (5{\cdot}S_5) \vdash x : S_5 \ \langle \mathsf{pos}(5) \rangle}\,\mathsf{ax}}{\dfrac{C;\, x : (S_k)_{k\in K} \vdash u : T \ \langle a{\cdot}0 \rangle}{C \vdash \lambda x.u : (S_k)_{k\in K} \to T \ \langle a \rangle}\,\mathsf{abs}} \quad (\text{with } 5 \in K,\ 8 \notin K)$$

Application rule

$$\dfrac{C \vdash u : (S_k)_{k\in K} \to T \ \langle a{\cdot}1 \rangle \qquad (D_k \vdash v : S_k \ \langle a{\cdot}k \rangle)_{k\in K}}{C \uplus (\uplus_{k\in K} D_k) \vdash u\, v : T \ \langle a \rangle}\,\mathsf{app}$$

Figure 3. Ascendance, Polar Inversion and Consumption

the (partial) converse of $\lfloor\cdot\rfloor$ *e.g.*, if $a_0 := \mathsf{pos}(7)$ exists, then a_0 is the axiom position candidate whose axiom track is 7: concretely, this just means that, if there exists a $\lfloor\cdot\rfloor$-derivation P typing t s.t. $a_0 \in \mathsf{supp}(P)$, then $P(A) = x : (7{\cdot}S) \vdash x : S$ for some type S and x.

■ Assume that, in a $\lfloor\cdot\rfloor$-derivation P, we find an abs-rule at position a as in Fig 3: the judgment $C; x : (S_k)_{k\in K} \vdash u : T$ (pos. $a \cdot 0$) yields $C \vdash \lambda x.u : (S_k)_{k\in K} \to T$ below (pos. a). The occurrence of T in the conclusion of the rule is intuitively the same as that in its premise: we say the former is the **ascendant** of the latter, since it occurs above in the typing derivation. Likewise, in the app-rule, the occurrence of T in $C \uplus_{k\in K} D_k \vdash u\, v : T$ stems from that of premise $C \vdash u : (S_k)_{k\in K} \to T$: the first occurrence of T is also the ascendant of T in the conclusion of the rule. Ascendance induces a stability relation $\to_{\mathsf{asc}}$ on the set of type elements *i.e.* on $\mathbb{B}^t$, the set of candidate bipositions of t, that can be formally defined by:

- $(a, c) \to_{\mathsf{asc}} (a \cdot 1, 1 \cdot c)$ if $t(a) = @$.
- $(a, 1 \cdot c) \to_{\mathsf{asc}} (a \cdot 0, c)$ if $t(a) = \lambda x$.

For instance, in Fig 4, the red (resp. the blue) occurrences of o' are ascendants of one another. They correspond to bipositions $(\varepsilon, 1^2) \to_{\mathsf{asc}} (0, 1) \to_{\mathsf{asc}} (0^2, \varepsilon) \to_{\mathsf{asc}} (0^2 \cdot 1, 1)$ (resp. $(\varepsilon, 1 \cdot 4 \cdot 1) \to_{\mathsf{asc}} (0, 4 \cdot 1)$) from bottom to top.

■ Assume that $5 \in K$ in the abs-rule at position a in Fig. 3. Then the occurrence of S_5 in $(S_k)_{k\in K} \to T$ at pos. a stems from an axiom rule concluding with $x : (5 \cdot S_5) \vdash x : S_5$ at pos. $\mathsf{pos}(5)$: we say that the occurrence S_5 (in $(S_k)_{k\in K} \to T$) is the **polar inverse** of the occurrence of S_5 in the axiom rule. Assume on the contrary that $8 \notin K$. So S_8 does not exist and there is no ax-rule typing x and using axiom track 8 above a. Morally, S_8 is empty.

More generally, if $t(a) = \lambda x$ and $k_0 \geqslant 2$, the function $\lfloor\cdot\rfloor$…

- …either gives a unique axiom position candidate for the type on track k in $(S_k)_{k\in K}$: this happens when $\exists a_0 \geqslant a \cdot 0$ s.t. $\lfloor a_0 \rfloor = k_0$ (*i.e.* , when $k_0 \in \mathsf{Tr}_\lambda(a)$ by construction of $\mathsf{Tr}_\lambda(a)$). In that case, $a_0 = \mathsf{pos}(k_0)$.
- …or tells us that $k_0 \notin K$ *i.e.* there cannot be a (top-level) type on track k_0 in $(S_k)_{k\in K} \to T$. In other words, this indicates that emptiness on track k_0 w.r.t. position a, which formally happens whenever no $a_0 \geqslant a \cdot 0$ satisfies $\lfloor a_0 \rfloor$ (*i.e.* $k_0 \notin \mathsf{Tr}_\lambda(a)$). In that case, $\mathsf{pos}(k_0)$ is undefined. We consider the type S_{k_0} (that intuitively does not exist) be the polar inverse of an empty type.

Polar inversion also induces a stability relation $\to_{\mathsf{pi}}$ on $\mathbb{B}^t$, that can be formally defined by:

$$\dfrac{\dfrac{\dfrac{}{4 \vdash x : (8{\cdot}o, 3{\cdot}o', 2{\cdot}o) \to o'}\,\mathsf{ax} \quad \dfrac{}{9 \vdash x : o\ [2]}\,\mathsf{ax} \quad \dfrac{}{2 \vdash x : o'\ [3]}\,\mathsf{ax} \quad \dfrac{}{5 \vdash x : o\ [8]}\,\mathsf{ax}}{\ldots \vdash x\, x\, x : o'}\,\mathsf{app}}{\vdash \lambda x.x\, x : (2 \cdot o', 4 \cdot (8 \cdot o, 3 \cdot o', 2 \cdot o) \to o', 5 \cdot o, 9 \cdot o) \to o'}\,\mathsf{abs}}{\vdash \lambda y x.x\, x : (\,) \to (2 \cdot o', 4 \cdot (8 \cdot o, 3 \cdot o', 2 \cdot o) \to o', 5 \cdot o, 9 \cdot o) \to o'}\,\mathsf{abs}$$

Figure 4. Threads, Ascendance and Consumption

- $(a, k \cdot c) \to_{\mathsf{pi}} (\mathsf{pos}(k), c)$ if $k \in \mathsf{Tr}_\lambda(a)$ (first case)
- $(a, k \cdot c) \to_{\mathsf{pi}} \mathsf{p}_\perp$ if $k \notin \mathsf{Tr}_\lambda(a)$ (second case)

For instance, in Fig. 4, the top blue occurrence of o' is the polar inverse of the top red one: formally, $(0, 4 \cdot 1) \to_{\mathsf{pi}} (0^2 \cdot 1, 1)$. Now, we may understand the use of constant $\mathsf{p}_\perp$: it indicates biposition that *cannot* be in any potential $\lfloor\cdot\rfloor$-derivation typing t. More precisely, $\mathsf{p}_\perp$ is here to play the role of the polar inverse of all the bipositions that cannot exist, because of the choices made by the function $\lfloor\cdot\rfloor$.

3.4 Type Formation, Type Consumption

In this subsection, we conclude the definitions of the stability relations that characterize the form of S-derivations, yielding the notion of **subjugation** (as in § 3.1), between candidate bipositions.

■ The notion of **consumption** is related to rule app. Assume $t(a) = @$, $t|_a = u\, v$ with $u : (S_k)_{k\in K} \to T$ and $v : S_k$ for all $k \in K$ as in Fig. 3 so that $u\, v$ can be typed with T. Each type S_k occurs in $(S_k)_{k\in K} \to T$ and $v : S_k$. However, it is absent in the type of $u\, v$: we say it has been **consumed**. Formally, we set, for all $(a, c) \in \mathbb{B}^t$, $k \geqslant 2$ s.t. $t(a) = @$:

- $(a \cdot 1, k \cdot c) \to_a (a \cdot k, c)$

In Fig. 4, the orange and the purple occurrences of o are consumed in the app-rule: formally, $(0^2 \cdot 1, 8) \to_{0^2} (0^2 \cdot 8, \varepsilon)$.

We set $\to = \cup\{\to_a \mid a \in \mathbb{A},\ t(a) = @\}$ and write $\leftarrow$ for the symmetric relation.

Let P be a $\lfloor\cdot\rfloor$-derivation typing a term t. If $\mathsf{p}_1 \to_{\mathsf{asc}} \mathsf{p}_2$ or $\mathsf{p}_1 \to_{\mathsf{pi}} \mathsf{p}_2$ or $\mathsf{p}_1 \to \mathsf{p}_2$, then $\mathsf{p}_1 \in \mathtt{bisupp}(P)$ iff $\mathsf{p}_2 \in \mathtt{bisupp}(P)$ (by construction of those relations).

■ Relations $\to_{\mathsf{t1}}$ and $\to_{\mathsf{t2}}$ ensure that the types are correctly defined and are natural extensions of those of Sec 3.1:

- For all $(a, c) \in \mathbb{B}^t$ and $k \in \mathbb{N}$, $(a, c \cdot k) \to_{\mathsf{t1}} (a, c)$.
- For all $(a, c) \in \mathbb{B}^t$ and $k \geqslant 2$, $(a, c \cdot k) \to_{\mathsf{t2}} (a, c \cdot 1)$.

■ The relation below $\to_{\mathsf{abs}}$ ensures that, if $\lambda x.u$ is a typed subterm of t, then its type T is an arrow type.

- if $t|_a = \lambda x.u$, then$(a, \varepsilon) \to_{\mathsf{abs}} (a, 1)$

■ The "big-step" stability relation $\to_{\mathsf{down}}$ below roughly states that the support of a potential derivation is a tree:

- $(a', c) \to_{\mathsf{down}} (a, \varepsilon)$
- $\mathsf{p}_\perp \to_{\mathsf{down}} (a, \varepsilon)$

Note that Lemma 7 would not hold without considering $\to_{\mathsf{down}}$.

■ We set $\to_\bullet = \to \cup \leftarrow \cup \to_{\mathsf{t1}} \cup \to_{\mathsf{t2}} \cup \to_{\mathsf{abs}} \cup \to_{\mathsf{down}}$. If $\mathsf{p}_1 \to_\bullet \mathsf{p}_2$, notice that, by construction, $\mathsf{p}_1 \in \mathtt{bisupp}(P)$ implies $\mathsf{p}_2 \in \mathtt{bisupp}(P)$. We say then that p_1 **subjugates** p_2, generalizing § 3.1.

3.5 Characterizing Bisupport Candidates with Threads

We prove now that the relations above are indeed enough to express a sufficient condition of typability (Corollary 1).

As we have seen, if P is a $\lfloor\cdot\rfloor$-derivation, then $\mathtt{bisupp}(P)$ is closed under $\to_{\mathtt{asc}}$, $\mathtt{asc}\leftarrow$, $\to_{\mathtt{pi}}$, $\mathtt{pi}\leftarrow$, $\to$, $\leftarrow$, $\to_{\mathtt{t1}}$, $\to_{\mathtt{t2}}$, $\to_{\mathtt{abs}}$ and $\to_{\mathtt{down}}$. Of course, $\mathtt{p}_\perp$, the empty biposition, cannot be in P. It turns out that it is enough to characterize candidate bisupports (Prop. 1). Let $\equiv$ be the reflexive, transitive, symmetric closure of $\to_{\mathtt{asc}} \cup \to_{\mathtt{pi}}$. We have:

Proposition 1. *Let $B \subseteq \mathbb{B}^t$. Then B is a $\lfloor\cdot\rfloor$-candidate bisupport for a derivation typing t (i.e. there exists a $\lfloor\cdot\rfloor$-derivation s.t. $B = \mathtt{bisupp}(P)$) iff (1) B is non-empty, (2) B is closed under $\equiv$ and $\to_\bullet$, and (3) B does not contain $\mathtt{p}_\perp$.*

In other words, a set of *real* bipositions (*i.e.* excluding $\mathtt{p}_\perp$) is the bisupport of an actual $\lfloor\cdot\rfloor$-derivation typing t when it is closed under the relations $\equiv$ and $\to_\bullet$. If such a set exists, then t is typable.

Proof sketch. The necessity of these conditions has been discussed in § 3.3 and 3.4. Conversely, assume that $\emptyset \neq B \subset \mathbb{B}^t \setminus \{\mathtt{p}_\perp\}$ is closed under $\equiv$ and $\to_\bullet$. We want a derivation P s.t. $\mathtt{bisupp}(P) = B$. For that, we need to suitably decorate the $\mathtt{p} \in B$. Mainly, a non-terminal biposition must be labelled with $\to$ and a terminal one with a fixed type atom o, in order to get correct types (as in § 3.1). On can check that P is a correct S-derivation using the definition of $\equiv$ and $\to_\bullet$. □

From now on, it will be better to reason modulo $\equiv$ (it may already be guessed that $\equiv$ *should* commute with $\to$, $\to_{\mathtt{t1}}$, . . ., which is made explicit in § 4.2) and to focus on subjugation.

Definition 1. *Let t be a term and $\lfloor\cdot\rfloor : \mathbb{N}^* \to \mathbb{N} \setminus \{0,1\}$ an injection, and $\to_{\mathtt{asc}}$, $\to_{\mathtt{pi}}$ the relations of ascendance and polar inversion in $\mathbb{B}^t$ defined w.r.t. $\lfloor\cdot\rfloor$.*

- *An **ascendant thread** is an equivalence class of relation $\equiv_{\mathtt{asc}}$, the reflexive, transitive, symmetric closure of $\to_{\mathtt{asc}}$.*
- *A **thread** (metavariable θ) is an equivalence class of relation $\equiv$ (see Fig. 4).*
- *The **quotient set** $\mathbb{B}^t/\!\equiv$ is denoted $\mathtt{Thr}$.*

In Fig. 4, the red occurrences of o' correspond to an ascendant thread and the blue one to another. Their union constitutes a (full) thread, that we denote θ_a. Likewise, the green and the orange occurrences of o respectively correspond to the negative and the positive part of a thread θ_b. The unique purple occurrence of o corresponds to a singleton thread θ_c.

The notation $\mathtt{Thr}$ implicitly depends on t and $\lfloor\cdot\rfloor$. The thread of $(a,c) \in \mathbb{B}$ is written $\mathtt{thr}(a,c)$ and we set:

$\theta_\varepsilon = \mathtt{thr}(\varepsilon,\varepsilon)$	$\theta_\perp = \mathtt{thr}(\mathtt{p}_\perp)$
"root thread"	"thread of emptiness"

If $\mathtt{thr}(\mathtt{p}) = \theta$, we say that θ **occurs at biposition** $\mathtt{p}$, also written $\theta : \mathtt{p}$ or $\mathtt{p} : \theta$ *e.g.*, $\theta_a : (\varepsilon, 1^2)$ or $\theta_a : (0, 4{\cdot}1)$.

We consider now the extension of every other relation modulo $\equiv$. Namely, we write $\theta_1 \tilde{\to}_a \theta_2$ if $\exists \mathtt{p}_1, \mathtt{p}_2,\ \theta_1 = \mathtt{thr}(\mathtt{p}_1),\ \theta_2 = \mathtt{thr}(\mathtt{p}_2),\ \mathtt{p}_1 \to_a \mathtt{p}_2$. Thus, $\theta_1 \tilde{\to}_a \theta_2$ iff $\theta_1 : \mathtt{p}_1 \to_a \mathtt{p}_2 : \theta_2$ for some $\mathtt{p}_1, \mathtt{p}_2$. In that case, we say that θ_1 (resp. θ_2) has been **left-consumed** (resp. **right-consumed**) at biposition $\mathtt{p}_1$ (resp. $\mathtt{p}_2$) *e.g.*, in Fig. 4, $\theta_b : (0^2{\cdot}1, 8) \tilde{\to}_{0^2} (0^2{\cdot}8, \varepsilon) : \theta_c$. We do likewise for $\to_{\mathtt{t1}}$, $\to_{\mathtt{t2}}$, $\to_{\mathtt{abs}}$, $\to_{\mathtt{down}}$, $\to_\bullet$, thus defining $\tilde{\to}_{\mathtt{t1}}$, $\tilde{\to}_{\mathtt{t2}}$, $\tilde{\to}_{\mathtt{abs}}$, $\tilde{\to}_{\mathtt{down}}$, $\tilde{\to}_\bullet$, whose reflexive transitive closure of relation $\tilde{\to}_\bullet$ is denoted $\tilde{\to}_\bullet^*$.

Corollary 1. *If $\theta_\perp$ is not in the transitive closure of $\{\theta_\varepsilon\}$ by $\tilde{\to}_\bullet$, then t is typable in* S *(by means of a $\lfloor\cdot\rfloor$-derivation).*

Proof. Let $\mathbb{B}_{\min} = \{\mathtt{p} \in \mathbb{B} \mid \theta_\varepsilon \tilde{\to}_\bullet^* \mathtt{thr}(\mathtt{p})\}$ *i.e.* $\mathbb{B}_{\min}$ is the union of the reflexive transitive closure of θ_ε under $\tilde{\to}_\bullet$. If $\theta_\varepsilon \to_\bullet{}^* \theta_\perp$ does not hold, then $\mathbb{B}_{\min}$ satisfies the hypotheses of Prop. 1. So there is a derivation P s.t. $\mathtt{bisupp}(P) = \mathbb{B}_{\min}$ and thus, t is typable. □

Analogies with first order model theory: given $t \in \Lambda$ and $\lfloor\cdot\rfloor$ and keeping in mind the intuition of bisupport candidates, let $\mathcal{T}_{t,\lfloor\cdot\rfloor}$ be the first order theory whose set of constants is $\mathtt{Thr}(P)$, that features one unary predicate symbol $\mathtt{inBis}$ (standing for "is in bisupport") and whose set of axioms is $\{\mathtt{inBis}(\theta_1) \equiv \mathtt{inBis}(\theta_2) \mid \theta_1, \theta_2 \in \mathtt{Thr}(P),\ \theta_1 \tilde{\to}_\bullet \theta_1\} \cup \{\neg\mathtt{inBis}(\theta_\perp)\}$. Then Corollary 1 states that there *exists* a $\lfloor\cdot\rfloor$-derivation P typing t iff $\mathcal{T}_{t,\lfloor\cdot\rfloor}$ is not contradictory: this is a sort of completeness result. Of course, it remains to be proved that $\mathcal{T}_{t,\lfloor\cdot\rfloor}$ is not contradictory (given any t). And this will be done using a technique closely associated to the λ-calculus: a finite reduction strategy (presented in § 5).

4 Nihilating Chains

We begin § 4 with a global description of the key steps leading to the fulfillment of Goal 4 (every term is $\lfloor\cdot\rfloor$-typable) giving the final result (every term is $\mathscr{R}$-typable) and a presentation of the central notion of nihilating chain.

For the purpose of proving that every term is typable, we want to prove that, for each term t and injection $\lfloor\cdot\rfloor : \mathbb{N}^* \to \mathbb{N} \setminus \{0,1\}$, there is a $\lfloor\cdot\rfloor$-derivation typing t. According to Corollary 1, we must show that $\theta_\perp$ is not in the reflexive transitive closure of θ_ε by $\tilde{\to}_\bullet$. A proof of $\theta_\varepsilon \tilde{\to}_\bullet^* \theta_\perp$ would involve a nihilating chain:

Definition 2.

- *A **chain** is a finite sequence of the form $\theta_0 \tilde{\to}_\bullet \theta_1 \tilde{\to}_\bullet \ldots \tilde{\to}_\bullet \theta_m$.*
- *When $\theta_0 = \theta_\varepsilon$, $\theta_m = \theta_\perp$, the chain is said to be **nihilating**.*

In order to apply Corollary 1, we must then prove that there are no nihilating chains. In other words, this corollary implies:

Proposition 2. *If the nihilating chains do not exist, then every term is $\lfloor\cdot\rfloor$-typable, and thus, also $\mathscr{R}$-typable.*

We proceed *ad absurdum* and consider $\theta_0 \tilde{\to}_\bullet \theta_1 \tilde{\to}_\bullet \ldots \tilde{\to}_\bullet \theta_m$ with $\theta_0 = \theta_\varepsilon$ and $\theta_m = \theta_\perp$. However, $\tilde{\to}_\bullet$ can be $\tilde{\to}$, $\tilde{\leftarrow}$, $\tilde{\to}_{\mathtt{t1}}$, $\tilde{\to}_{\mathtt{t2}}$, $\tilde{\to}_{\mathtt{abs}}$ or $\tilde{\to}_{\mathtt{down}}$. The structure of the proof is the following:

- We define (Definition 3) the notion of *polarity* for bipositions: a biposition is negative when it is created by an $\mathtt{abs}$-rule (modulo $\to_{\mathtt{asc}}$) and positive otherwise.
- The termination of a finite collapsing strategy (Sec. 5.2) guarantees that positivity can be assumed to only occur at suitable places in the chain without loss of generality. In that case, we say that the chain is *normal* (Definition 4).
- In normal chains, the different cases of subjugation interact well (§ 4.2), so that, from any normal chain, we may build another that begins with $\theta_\varepsilon \tilde{\to}_\bullet \theta_1$. This is easily shown to be impossible, which entails that nihilating chains do not exist and that every term is S-typable.

4.1 Polarity and Threads

In this section, we define the key notion of syntactic polarity.

If $\mathtt{p} \to_{\mathtt{asc}} \mathtt{p}_i$ $(i = 1, 2)$ then $\mathtt{p}_1 = \mathtt{p}_2$ ($\to_{\mathtt{asc}}$ is functional) and we write $\mathtt{p}_1 = \mathtt{p}_2 = \mathtt{asc}(\mathtt{p})$. We set, for all $\mathtt{p} \in \mathbb{B}$, $\mathtt{Asc}(\mathtt{p}) = \mathtt{asc}^i(\mathtt{p})$, where i is maximal (*i.e.* $\mathtt{asc}^i(\mathtt{p})$ is defined, but not $\mathtt{asc}^{i+1}(\mathtt{p})$). Thus, $\mathtt{Asc}(\mathtt{p})$ is the **top ascendant** of $\mathtt{p}$ *e.g.*, in Fig. 4, the top red (resp. blue) occurrence of o' is the top ascendant of the other ones (resp.

one). A top ascendant is either located in an ax-node (*e.g.*, the top red ascendant in Fig. 4) or in an abs-node (*e.g.*, the blue ones), motivating:

Definition 3.

- *Let* $\mathtt{p} \in \mathbb{B}^t \setminus \{\mathtt{p}_\perp\}$ *and* $(a_0, c_0) = \mathtt{Asc}(\mathtt{p})$. *We define the* ***polarity*** *of* p *as follows: if* $t(a_0) = x$ *for some* $x \in \mathcal{V}$, *then we set* $\mathtt{Pol}(\mathtt{p}) = \oplus$ *and if* $t(a_0) = \lambda x$, *then we set* $\mathtt{Pol}(\mathtt{p}) = \ominus$. *We also set* $\mathtt{Pol}(\mathtt{p}_\perp) = \ominus$.
- *If* $\mathtt{thr}(\mathtt{p}) = \theta$ *and* $\mathtt{Pol}(\mathtt{p}) = \oplus/\ominus$, *we say that* θ *occurs positively/negatively at biposition* p.
- *If* θ *is left/right-consumed at* p *and* $\mathtt{Pol}(\mathtt{p}) = \oplus$ *(resp.* $\mathtt{Pol}(\mathtt{p}) = \ominus$*), we say that* θ *is left/right-consumed positively (resp. negatively) at biposition* p.

Then, we write for instance $\theta_1 \overset{\oplus}{\tilde{\to}}_a \overset{\ominus}{} \theta_2$ to mean that θ_1 is left-consumed positively and θ_2 is right-consumed negatively in the app-rule at position a. In Fig. 4, the blue occurrences of o' are negative, the red ones are positive and $\theta_b \overset{\oplus}{\tilde{\to}}_{1^2} \overset{\oplus}{} \theta_c$.

4.2 Interactions in Normal Chains

In § 4.2, we present the notion of normal chain and explicit some interaction properties that allow us to simplify/rewrite them.

As it has been discussed in § 1.5 and 2.1, the possibility for a variable x (of a redex or of a redex to be created later) to be substituted in a reduction sequence is problematic. Intuitively, a biposition is negative when it was "created" in an abstraction λx and that left-consumption is associated to left-hand sides of application. Thus, a negative left-consumption hints at the presence of redex (this intuition will be made more explicit in Sec. 5.2). More precisely, it indicates the presence of what we will call a *redex tower*. This suggests the following notion:

Definition 4. *A chain is* ***normal*** *if no thread is left-consumed negatively in it (there is no link of the form* $\theta_i \overset{\ominus}{\tilde{\to}} \theta_{i+1}$ *or* $\theta_i \overset{\ominus}{\tilde{\leftarrow}} \theta_{i+1}$*).*

Normal chains can be handled! The **interaction lemmas** below describe some commutations between stability relations.

Lemma 2. *If* $\theta_1 \tilde{\to}_{\mathtt{t1}} \theta_2$ *and* $\theta_2 \tilde{\to} \theta_4$, *then,* $\exists \theta_3$, $\theta_1 \to \theta_3$ *and* $\theta_3 \tilde{\to}_{\mathtt{t1}} \theta_4$.

Lemma 3. *If* $\theta \overset{\oplus}{\tilde{\to}} \theta'$, *there is no* θ_0 *s.t.* $\theta_0 \tilde{\to}_{\mathtt{abs}} \theta$ *or* $\theta_0 \tilde{\to}_{\mathtt{down}} \theta$.

Lemma 4. *If* $\theta_1 \tilde{\to}_{\mathtt{t2}} \theta_2$ *and* $\theta_2 \overset{\oplus}{\tilde{\to}} \theta_4$, *then,* $\exists \theta_3$, $\theta_1 \overset{\oplus}{\tilde{\to}} \theta_3$ *and* $\theta_3 \tilde{\to}_{\mathtt{t2}} \theta_4$.

Lemma 5.

- *If* $\mathtt{thr}(\mathtt{p}) = \theta_\perp$, *then* $\mathtt{Pol}(\mathtt{p}) = \ominus$.
- *If* $\theta \tilde{\to}_{\mathtt{t1}} \theta_\perp$ *or* $\theta \to_{\mathtt{t2}} \theta_\perp$, *then* $\theta = \theta_\perp$.
- *We cannot have* $\theta \tilde{\to}_{\mathtt{abs}} \theta_\perp$ *or* $\theta \tilde{\to}_{\mathtt{down}} \theta_\perp$.

Goal 4 (almost) at Hand Using Lemmas 2, 3, 4, 5, it is not difficult to define an algorithm taking a normal nihilating chain as input (if one exists) and outputting a chain of the form $\theta_\varepsilon = \theta_0 \overset{\oplus}{\tilde{\to}} \theta_1 \overset{\oplus}{\tilde{\to}} \ldots \overset{\oplus}{\tilde{\to}} \theta_\ell = \theta_\perp$. See § 12.3.3 in [18] for details.
Then, one proves that there is no θ such that $\theta_\varepsilon \overset{\oplus}{\tilde{\to}} \theta$. This implies that there is no chain of the form $\theta_\varepsilon = \theta_0 \overset{\oplus}{\tilde{\to}} \theta_1 \overset{\oplus}{\tilde{\to}} \ldots \overset{\oplus}{\tilde{\to}} \theta_\ell = \theta_\perp$, and then, that there is no normal nihilating chain:

Proposition 3. *There is no normal nihilating chain.*

Proposition 3 *almost* proves that every term is $\lfloor\cdot\rfloor$-typable (by Proposition 2). Almost, because only the non-existence of *normal* nihilating chains is ensured for now (and not that of nihilating chains in general). The only point that will remain to be verified is that *normal* nihilating chains can be considered without loss of generality (which is the object of § 5).

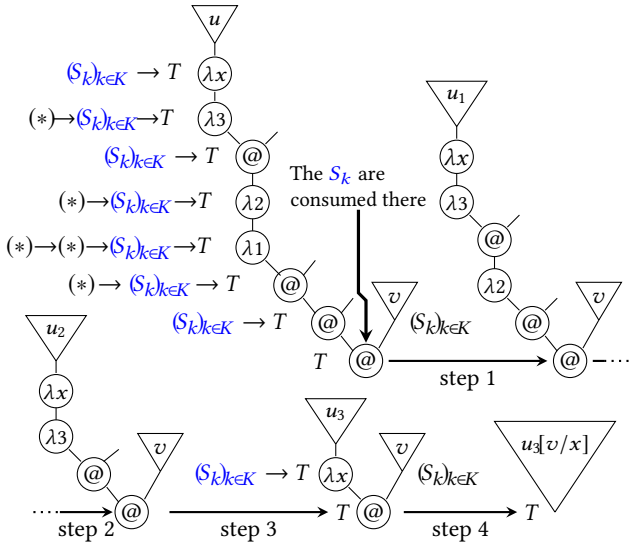


Figure 5. Collapsing a Redex Tower

5 Normalizing Nihilating Chains

5.1 Residuation and Subjugation

In this section, we explain why negative left-consumption in a nihilating chain can be avoided (without loss of generality). By Prop. 2 and 3, this will allow us to prove that every term is typable. The fact that system S is relevant, non-idempotent, rigid and syntax-directed entails that if $P \rhd C \vdash t : T$ and $t \overset{b}{\to}_\beta t'$, then, there is a unique derivation $P' \rhd C \vdash t' : T$ obtained from P by subject reduction (thus, subject reduction is *deterministic* in system S). Moreover, intuitively, every part of P' comes from a part of P and so, every position and right biposition of P' can be thought as the **(quasi-)residual** of position or (right) biposition of P'. We do not give details (that can be found in § IV and Fig. 1 in [17]), but this induces a function $\mathtt{QRes}_b$ from the right bisupport of P to that of P'. The function $\mathtt{QRes}_b$ turns out to be compatible with thread-membership:

Lemma 6. *If* $\mathtt{p}_1 \equiv \mathtt{p}_2$ *in* P, *then* $\mathtt{QRes}_b(\mathtt{p}_1) \equiv \mathtt{QRes}_b(\mathtt{p}_2)$ *in* P'.

This Lemma allows us to define (quasi-)residuals for *threads*. We set $\mathtt{Res}_b(\theta) = \mathtt{thr}'(\mathtt{QRes}_b(\mathtt{p}))$ for any $\mathtt{p} : \theta$ (where $\mathtt{thr}'(\cdot)$ denotes threads in $\mathbb{B}^{t'}$). By case analysis, we have:

Lemma 7. *Let* $\theta_1, \theta_2 \in \mathtt{Thr}$. *We set* $\theta'_i = \mathtt{Res}_b(\theta_i)$ $(i = 1, 2)$.

- *If* $\theta_1 \tilde{\to} \theta_2$, *then* $\theta'_1 \tilde{\to} \theta_2$ *or* $\theta'_1 = \theta'_2$.
- *If* $\theta_1 \tilde{\to}_{\mathtt{t1}} \theta_2$, *then* $\theta'_1 \tilde{\to}_{\mathtt{t1}} \theta'_2$ *or* $\theta'_1 = \theta'_2$.
- *If* $\theta_1 \tilde{\to}_{\mathtt{t2}} \theta_2$, *then* $\theta'_1 \tilde{\to}_{\mathtt{t2}} \theta'_2$, $\theta'_1 \tilde{\to}_{\mathtt{down}} \theta'_2$ *or* $\theta'_1 = \theta'_2$.
- *If* $\theta_1 \tilde{\to}_{\mathtt{abs}} \theta_2$, *then* $\theta'_1 \tilde{\to}_{\mathtt{abs}} \theta'_2$ *or* $\theta'_1 = \theta'_2$.
- *If* $\theta_1 \tilde{\to}_{\mathtt{down}} \theta_2$, *then* $\theta'_1 \tilde{\to}_{\mathtt{down}} \theta'_2$ *or* $\theta'_1 = \theta'_2$.

See § 12.4.1 in [18] for the proofs of Lemmas 6 and 7.

Finally, $\mathtt{Res}_b(\theta_\varepsilon) = \theta_\varepsilon$ and $\mathtt{Res}_b(\theta_\perp) = \theta_\perp$ as expected, so Lemma 7 implies that, if there is a nihilating chain for t of length m, then there is one for t' of length $\leqslant m$ (whenever $t \to^* t'$).

5.2 The Collapsing Strategy

We explain now how to *normalize* a chain *i.e.* discard negative left-consumption. This will allow us to use Proposition 3 to finally conclude that nihilating chains do not exist.

The idea is that if $\theta_{\mathsf{L}} \overset{\ominus}{\tilde{\to}}_a \theta_{\mathsf{R}}$, then either $t|_a$ is a redex and we have $\mathtt{Res}_b(\theta_{\mathsf{L}}) = \mathtt{Res}_b(\theta_{\mathsf{R}})$ (*i.e.* θ_1 and θ_2 are **collapsed** by the reduction step) or θ_{L} passes through a redex. When we reduce this redex, the "height" of θ_{L} will decrease. More precisely, the 2nd case is associated with the notion of **redex tower**, which is more or less a *finite* nesting of redexes, that can – more importantly – be collapsed in a *finite* number of steps. A case of negative left-consumption of a sequence type $(S_k)_{k\in K}$ (which is the domain of the abstraction $\lambda x.u$), coming along with a redex tower, is represented in Fig. 5 (we write λi instead of λx_i and $(*)$ for matterless sequence types). The sequence type $(S_k)_{k\in K}$ of negative polarity is "called" by the node λx at the top of the figure and consumed at the bottom app-rule. The initial redex tower is reduced in 4 steps, so that its height decreases and finally, the types S_k, that were left-consumed negatively, are destroyed in the final term $u_3[v/x]$. .

Lemma 8. *If $\theta_1 \overset{\ominus}{\tilde{\to}} \theta_2$, then there is a reduction path* $\mathtt{rs}$ *such that* $\mathtt{Res}_{\mathtt{rs}}(\theta_1) = \mathtt{Res}_{\mathtt{rs}}(\theta_2)$ *(residuation naturally extends along with* $\mathtt{rs}$*).*

This Lemma, along with the conclusion of § 5.1, yields:

Proposition 4. *There is a reduction strategy (the "**collapsing strategy**") producing a normal nihilating chain from any nihilating chain.*

6 Applications

We can now prove that every term is $\lfloor\cdot\rfloor$-typable (and thus, also $\mathscr{R}$-typable, by Goal 4), using Prop. 2, the residuation of threads (S 5.1), the collapsing strategy (Prop. 4) and the non-existence of normal threads (Prop. 3), which is ensured by the Interaction Lemmas.

Theorem 1. *Every λ-term is typable in the relevant and non-idempotent intersection type system $\mathscr{R}$.*

By the same techniques (the complete proofs are in § 12.4.5 of [18]), system $\mathscr{R}$ discriminates terms w.r.t. their arities, as claimed:

Lemma 9. *Let t be a zero term and o a type atom, then there is context C such that $C \vdash t : o$ is* S*-derivable.*

Proof sketch. Let t be a term s.t. $\theta_\varepsilon \overset{\sim}{\to}_\bullet^* \mathtt{thr}(\varepsilon, 1)$ *i.e.* s.t. $(\varepsilon, 1) \in \mathbb{B}_{\min}$ (see Corollary 1), which implies that the type of t cannot be a type atom by the proof of this same corollary. We prove that t is of arity $\geqslant 1$, which is enough to conclude.

For that, we consider a *λ-chain i.e.* a chain of the form $\theta_\varepsilon = \theta_0 \overset{\sim}{\to}_\bullet \ldots \overset{\sim}{\to}_\bullet \theta_m = \mathtt{thr}(\varepsilon, 1)$, of minimal length. The notion of normal chains extends to λ-chains and by the collapsing strategy, we can replace t by a reduct t' s.t. the considered chain is normal.

Using *ad hoc* interaction lemmas, we prove that the normality of the chain entails $\theta_\varepsilon \overset{\sim}{\to}_{\mathtt{abs}} \mathtt{thr}(\varepsilon, \varepsilon)$. Collapsing then redex towers, we may reduce t' to an abstraction $\lambda x.t''$. Q.E.D. □

Theorem 2. *Let t be a term of arity n. Then there is a context Γ and a type τ of arity n (see Sec. 2.4) such that $\Gamma \vdash t : \tau$ is $\mathscr{R}$-derivable.*

Proof sketch. When $n = \infty$, this comes from Theorem 1, subject reduction and the abs-rule. When $n \in \mathbb{N}$, we use Lemma 9 and (finite) subject expansion (Proposition 1). □

Conclusion: We proved that every term is typable in a reasonable relevant intersection type system (Theorem 1). If we take the typing rules of S *coinductively* (and not only the type grammar), we can also type every infinitary λ-term [15].

The techniques that we have developed here build, to the best of our knowledge, the first bridge between first-order model theory and the study of models of the pure λ-calculus. They are actually modular: we also use them, in a companion paper, to prove that every multiset-based derivation is the collapse of a sequential derivation [19]. This suggests that these techniques could be used to study the coinductive version of finitary models of the λ-calculus and extend some of their semantical properties to all λ-terms.

By setting, for each term t, $[\![t]\!]_{\mathtt{rel}_\infty} = \{\rhd_{\mathscr{R}}\Gamma \vdash t : \tau \mid \Gamma, \tau\}$ (*cf.* § 1.3), one defines the **infinitary version of the relation model**, in which, by Theorem 1, no term has a trivial denotation, including the mute terms. This model is thus **non-sensible** [2] since it does not equate all the non-head normalizing terms (*e.g.*, Ω and $\lambda x.\Omega$ of respective arities 0 and 1) by Theorem 2.

We presented a first semantical result about this model (Theorem 2), but its equational theory has yet to be studied. According to the same theorem, this model equates all the closed zero terms. It then differs both from the non-sensible model of Berarducci trees and that of Lévy-Longo trees, respectively related to Λ^{111} and Λ^{001} in [15]. This work may suggest a new notion of tree, that could shed some light on Open Problem # 18 of TLCA (the problem of finding trees related to various contextual equivalences).

The study of infinitary models (beyond infinite tree models) is at its early stages, but it already provides descriptions of the infinitary behaviors of λ-terms (*cf.* Grellois-Melliès' infinitary model of Linear Logic in [12, 13]). The semantical implications of the main theorem (every term is $\mathscr{R}$-typable) remain to be understood and the proof techniques presented here can certainly be used to study infinitary models or coinductive/recursive type systems *before* they are endowed with some validity or guard condition, or maybe to build other models of pure λ-calculus, for instance, to get some semantical proof of the *easiness* [14] of sets of mute terms, as in [4].

References

[1] H. Barendregt. *The Lambda-Calculus: Its Syntax and Sematics.* Ellis Horwood series in computers and their applications. Elsevier, 1985.

[2] A. Berarducci. Infinite lambda-calculus and non-sensible models. *Lecture Notes in Pure and Applied Mathematics*, 180:339–377, 1996.

[3] A. Berarducci and B. Intrigila. Some new results on easy lambda-terms. *Theor. Comput. Sci.*, 121(1&2):71–88, 1993.

[4] A. Bucciarelli, A. Carraro, G. Favro, and A. Salibra. Graph easy sets of mute lambda terms. *Theor. Comput. Sci.*, 629:51–63, 2016.

[5] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Not enough points is enough. In *CSL 2007, Lausanne*, pages 298–312.

[6] A. Bucciarelli, D. Kesner, and D. Ventura. Non-idempotent intersection types for the lambda-calculus. *MSCS*, 2017.

[7] F. Cardone and J. R. Hindley. *Lambda-Calculus and Combinators in the 20th Century.* 2009.

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

[9] L. Czajka. A coinductive confluence proof for infinitary lambda-calculus. In *RTA-TLCA 2014, Vienna Summer of Logic, VSL.*

[10] D. de Carvalho. *Sémantique de la logique linéaire et temps de calcul.* PhD thesis, Université Aix-Marseille, Nov. 2007.

[11] P. Gardner. Discovering needed reductions using type theory. In *TACS 94, Sendai.*

[12] C. Grellois and P. Melliès. An infinitary model of linear logic. In *FoSSaCS 2015.*

[13] C. Grellois and P. Melliès. Relational semantics of linear logic and higher-order model checking. In *CSL 2015, Berlin.*

[14] G. Jacopini. A condition for identifying two elements of whatever model of combinatory logic. In *Lambda-Calculus and Computer Science Theory, 1975.*

[15] R. Kennaway, J. W. Klop, M. R. Sleep, and F. de Vries. Infinitary lambda calculus. *Theor. Comput. Sci.*, 175(1):93–125, 1997.

[16] J. Krivine. *Lambda-calculus, types and models.* Ellis Horwood series in computers and their applications. Masson, 1993.

[17] P. Vial. Infinitary intersection types as sequences: A new answer to Klop's problem. In *LICS, 2017, Reykjavik.*

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

[19] P. Vial. Representing permutations without permutations, or the expressive power of sequence types, http://arxiv.org/abs/1610.06399, (submitted).