Reduction paths without permutations, or the expressive power of sequence types

Pierre Vial
firstname.name@inria.fr
Inria, Nantes

Abstract— **Recent works have championed a rigid description of resources: whereas in non-rigid paradigms (*e.g.*, standard Taylor expansion or non-idempotent intersection types), bags of resources are multisets and invariant under permutation, in the rigid ones, permutations must be processed explicitly and can be allowed or disallowed. Rigidity enables a fine-grained control of reduction paths and their effects on, *e.g.*, typing derivations. We previously introduced [14], [16] a very constrained coinductive type system (system $\mathtt{S}$) in which permutation is completely disallowed. This raises the question of the possible loss of expressivity in system $\mathtt{S}$ w.r.t. reduction paths, compared to the usual multiset framework or a rigid one allowing permutations. To address this problem, we introduce an extension of system $\mathtt{S}$ called $\mathtt{S}_{\mathrm{op}}$, which features isomorphisms of types and allows retrieving the refinement type system of Asada-Ong-Tsukada using list types and permutations. Then, by proving that every $\mathtt{S}_{\mathrm{op}}$-typing has an isomorphic representation in system $\mathtt{S}$, we show that not only every non-idempotent derivation (*i.e.* every point of the infinitary relation model) can be represented by a rigid, permutation-free derivation, but also that any dynamic behavior may be captured in this way. In other words, we prove that system $\mathtt{S}$ has full expressive power over multiset or permutation-inclusive intersection.**

I. Determinism, Rigidity and Reduction Paths

A. Resources for the λ-Calculus

The attempts at giving a quantitative account of resource consumption by functional programs originates from Girard's Linear Logic [9]. In his wake, several works were proposed to capture the same ideas. In this introduction, we mainly focus on two of them:

- The **resource calculus**, notably by Ehrhard-Regnier [5]: instead of having applications of the form $t\,u$ (one function, one argument), applications are of the form $t\,[u_1,\ldots,u_n]$ (one function, several argument), where $[u_1,\ldots,u_n]$ is a **multiset** (also called a **bag**). Substitution of a variable x with a bag $[u_1,\ldots,u_n]$ in a term t is **linearly** processed: it occurs only if there are exactly n free occurrences of x in t. In that case, each occurrence of x is replaced by one u_i. Since the elements of a multiset can be permuted, this operation is non-deterministic *e.g.*, if $n = 2$ and $t = x\,[x,y]$, then the considered substitution can output $u_1\,[u_2,y]$ or $u_2\,[u_1,x]$. Thus, the redex $(\lambda x.t)[u_1,u_2]$ reduces in two ways: there are **reduction choices**. The usual untyped λ-calculus can be embedded into the resource calculus *via* the **Taylor expansion**, which can be understood as a *linearization* of λ-terms. However, since one does not know how many times a function may use its argument, the Taylor expansion $v \mapsto \tilde{v}$ represents an application $t\,u$ of the usual λ-calculus by a formal series involving $\tilde{t}\,[\,]$, $\tilde{t}\,[\tilde{u}]$, $\tilde{t}\,[\tilde{u},\tilde{u}]$, $\tilde{t}\,[\tilde{u},\tilde{u},\tilde{u}]$... The semantics of the resource calculus is compatible with that of the λ-calculus in that, the computation of the Taylor expansion and that of the Böhm tree of a term commute (*adequation*).
- **Non-idempotent intersection types**, notably by Gardner [8] and de Carvalho [4]: intersection type systems, introduced by Coppo-Dezani [3], feature a type construction $\wedge$ (intersection). In the non-idempotent setting, $\wedge$ can be seen as a free operator. Gardner-de Carvalho's original system $\mathcal{G}$ features an explicit permutation rule allowing to permute types in the context:
$$\frac{\Gamma; x : A_1 \wedge \ldots \wedge A_n \vdash t : B \qquad \rho \in \mathfrak{S}_n}{\Gamma; x : A_{\rho(1)} \wedge \ldots \wedge A_{\rho(n)} \vdash t : B}$$
where $\mathfrak{S}_n$ is the set of permutations of $\{1,\ldots,n\}$. A later presentation of system $\mathcal{G}$, that we call system $\mathscr{R}_0$ here (see Sec. II-A for details), represents intersection with multisets *i.e.* the lists $A_1 \wedge \ldots \wedge A_n$ are inductively collapsed to $[A_1,\ldots,A_n]$. Since $[A_1,\ldots,A_n] = [A_{\rho(1)},\ldots,A_{\rho(n)}]$, the $\mathtt{perm}$-rule becomes obsolete and system $\mathscr{R}_0$ is *syntax-directed*, meaning that a given judgment may be the conclusion of a unique rule. Syntax-direction makes type system more readable and closer to the syntax of the λ-calculus. Indeed, system $\mathscr{R}_0$ features only three typing rules, corresponding to the three constructors of the λ-calculus (x, λx or @).
Because of non-idempotency and relevance (no weakening), system $\mathscr{R}_0$ is *linear*. Moreover, it characterizes head normalization: a term is $\mathscr{R}_0$-typable iff it is head normalizing (HN). Variants of system $\mathscr{R}_0$ give characterizations of weak, strong or weak head normalization (see [2] for a survey). Most interestingly, non-idempotency makes the termination property of system $\mathscr{R}_0$ (if t is $\mathscr{R}_0$-typable, then t is HN) straightforward to prove and the size of $\mathscr{R}_0$-derivations gives upper bounds on the length of normalizing reduction sequences (quantitativity). From the semantic point of view, the typing judgments of $\mathscr{R}_0$ correspond to the points of the relational model of the λ-calculus [1], which is one of the simplest to handle, for the reasons mentioned above.

Contrary to [12], we are mainly interested in non-idempotent intersection type systems and not in the Taylor expansion, but they are related in that, an $\mathscr{R}_0$-typing of a term t naturally gives an element of the Taylor expansion of t that does not reduce to the null sum.

B. Rigidity and Determinism

In this article, we address problems of expressivity related to *rigidity* in the resource frameworks. We present this feature now. First, as noted above, the linear substitution of a variable with the elements of a multiset bag is non-deterministic. For system $\mathscr{R}_0$, this entails that subject reduction is *non-deterministic* (this is further discussed with Fig. 1): we say that system $\mathscr{R}_0$ gives rise to different **reduction choices**.

In [14], we characterized a form of infinitary weak normalization by using non-idempotent types. This required using a **coinductive** (meaning infinitary) grammar of types. However, coinductive grammars give rise to unsound derivations *e.g.*, typings of $\Omega := \Delta\,\Delta$ (with $\Delta = \lambda x.x\,x$), the auto-autoapplication. To recover soundness, coinductive type grammars must come together with a validity criterion. We called the one that we introduced in [14] *approximability*. It turns out (Sec. IV.E. of [14]) that approximability *cannot* be defined with multiset intersection. This led us to introduce system $\mathtt{S}$ (see Sec. II-C), a *rigid* variant of system $\mathscr{R}_0$: multisets are coinductively replaced by *sequences* of types *i.e.* families of types annotated with integers. For instance, $(2{\cdot}S, 3{\cdot}T, 8{\cdot}S)$ is a **sequence type** that features two occurrences of type S and one occurrence of T. We have a "disjoint union" operator $\uplus$ for sequences *e.g.*, $(2{\cdot}S, 3{\cdot}T) \uplus (8{\cdot}S) = (2{\cdot}S, 3{\cdot}T, 8{\cdot}S)$, so that the occurrences of S annotated with 2 can be identified on each side of the equality (**tracking**). In contrast, $[\sigma, \tau] + [\sigma] = [\sigma, \sigma, \tau]$, but, in this equality, we have no way to relate one occurrence of σ in $[\sigma, \sigma, \tau]$ to $[\sigma, \tau]$ rather than $[\sigma]$ and *vice versa* (tracking is impossible with multisets). The lack of tracking is the main cause of non-determinism. In other words, one may say that intuitively, a framework is **rigid** when it enables tracking and determinism. Multiset equality can be seen as *lax* whereas sequential equality is *syntactical* (and thus, highly constraining). The syntactic nature of sequences entails that subject reduction is *deterministic* in system $\mathtt{S}$ (Fig. 4).

In their recent works [11], [12], Asada, Ong and Tsukada proposed another rigid framework, featuring **rigid resource calculus** and a **rigid Taylor expansion**, also satisfying adequation. Their approach has several aims, notably giving a precise account of the weight of reduction paths in probabilistic programming: rigidity makes it possible to enumerate reduction paths *statically* (what is referred to as the *compositional enumeration problem*). This rigid calculus is interpreted in generalized species of structures, which are a special case of cartesian closed bicategories featuring non-trivial isomorphisms. These isomorphisms generalize the `perm`-rule and enable the encoding of convoluted reduction paths. However, without `perm`-rule or type isomorphisms, subject reduction fails in the presence of list types, which is their main drawback. The reason for this failure is that reduction moves subterms in the subject and thus reorganizes the way types are concatenated or built. But this is not a problem with multisets, which are permutation invariants, or with sequences, which are not impacted by the concatenation order since $\uplus$ is commutative *e.g.*, $(2{\cdot}S, 3{\cdot}T) \uplus (8{\cdot}S) = (8{\cdot}S) \uplus (2{\cdot}S, 3{\cdot}T)$. In particular, system $\mathtt{S}$ satisfies subject reduction and expansion without needing a permutation rule.

Thus, system $\mathtt{S}$ has apparently the best features: it is rigid and enjoys type invariance under conversion without burdening derivations with permutations. Yet, the lack of permutations in system $\mathtt{S}$ makes that it may *seem* dynamically poor: there is no way to implement more than one reduction choice in a $\mathtt{S}$-derivation.

There is another problem arising from the lack of permutation of system $\mathtt{S}$: in [16], we proved that every λ-term is $\mathscr{R}$-typable, where $\mathscr{R}$ is the coinductive extension of $\mathscr{R}_0$. This entails that every term has a *non-empty* interpretation in the infinitary relational model. Our proof of this result, which uses tracking, can be only obtained by working in system $\mathtt{S}$ instead of $\mathscr{R}$ and then, noticing that sequences collapse to multisets when tracks are erased (*e.g.*, $(2{\cdot}S, 3{\cdot}T, 8{\cdot}S)$ onto $[S, T, S]$). However, this does not exclude the possibility that there are $\mathscr{R}$-derivations (*i.e.* points of the infinitary relational model) that are *not* the collapse of any $\mathtt{S}$-derivation. In other words, this rises the question: can every $\mathscr{R}$-derivation be represented with a $\mathtt{S}$-derivation (question 1)? Due to the *syntactical* equality used in the application rule of system $\mathtt{S}$ and the absence of productivity, this turns out to be a difficult problem (see Sec. II-E). We will prove that the answer is yes.

The question of the collapse allows us to properly address the description of the dynamical expressivity of system $\mathtt{S}$: a derivation of system $\mathtt{S}$ has only one reduction path whereas a $\mathscr{R}$-derivation may have an infinity. Observe that there is no obvious reason why there is a way to represent every reduction path in the non-deterministic setting (using multiset) by appropriately choosing the right $\mathtt{S}$-representation. This rises this second question: given a reduction path $\mathtt{rp}$ of $\mathscr{R}$-derivations Π_0, $\Pi_1 \ldots$, is there a $\mathtt{S}$-derivation whose unique reduction path collapses derivation-wise on every element of $\mathtt{rp}$ (question 2)? It turns out that this is also true.

C. Contributions and Challenges

Contributions: We prove that sequential intersection surjectively collapses to multiset intersection by answering positively to the two questions above. The main contributions of this article are the following:

- We introduce a sequence type system $\mathtt{S}_{\mathtt{op}}$, which allows implementing reduction paths by means of type isomorphisms.
- System $\mathtt{S}_{\mathtt{op}}$ allows retrieving a system that is equivalent to Tsukada-Asada-Ong rigid refinement type system [12], using list types and permutations.
- We prove that every $\mathtt{S}_{\mathtt{op}}$-derivation is isomorphic to a $\mathtt{S}$-derivation *i.e.* a derivation with sequence types without permutation rules or type isomorphisms.
- Concomitantly, we use $\mathtt{S}_{\mathtt{op}}$ to show every point of the infinitary relational model is the collapse of a sequential derivation without permutation.

Once again, subject reduction is not satisfied with list intersection (*i.e.* without permutation/isomorphisms). On another hand, isomorphisms of types enables the description of reduction paths, as shown in [12]. Our article shows that sequence types provide the best of both worlds (multiset and list intersections): one can also endow sequence types with isomorphisms to directly encode reduction path which is the motivates introducing $\mathtt{S}_{\mathtt{op}}$, but it is not necessary since:

- Subject reduction holds with sequences without permutation rule.
- Theorem 6 (every $\mathtt{S}_{\mathtt{op}}$ derivation is isomorphic to an $\mathtt{S}$-derivation) implies that any reduction path can directly be encoded without using type isomorphisms when one chooses a suitable rigid representative.

Structure of the paper and challenges: in Sec. II, we compare various type system and present the problematic of this article and its difficulties in more details. In Sec. III, we present system $\mathtt{S}_{\mathtt{op}}$ and the way reduction and residuation are processed in the presence of type isomorphisms. In Sec. IV, we prove the main theorems.

The most difficult point establishing Theorem 6 is dealing with the lack of productivity. In a finitary/productive framework (for the notion of productivity, see *e.g.*, [6]), this could be possible by studying first the derivations typing a (partial) normal form, for which representation is usually easily ensured, and then proceeding by subject expansion (see Sec. II-E). However, as already noted, typability in system $\mathscr{R}$ does not imply any kind of normalization (every term can be $\mathscr{R}$-typed), even an infinitary one. We then refine the technique that we developed in [16] to study typing derivations without hypothesis of normalization. This technique involves *threads*, which are special sets of markers in the derivation (these markers exist thanks to rigidity) and a first order theory $\mathcal{T}$ pertaining to these threads. We then prove that Theorem 6 by proving that (a) it is true iff $\mathcal{T}$ is consistent and (b) proving that that $\mathcal{T}$ is consistent. The consistence of $\mathcal{T}$ relies on a reduction strategy on *threads* instead of *terms*. The main stage of this proof are presented in Sec. IV.

All our results are valid for the infinitary calculus Λ^{001} [10], but we handle here the case of the finite λ-calculus to lighten the presentation. See the webpage of the author or Chapter 11 and 13 of [15] for complete proofs and details.

II. Non-Idempotent Intersection and Rigid Paradigms

A. Multiset Intersection

Let us recall Gardner and de Carvalho's non-idempotent intersection type system [4], [8] in its non-rigid version. The set of $\mathscr{R}_0$-types is inductively 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 (intersection) type**. Intersection $\wedge$ corresponds to the multiset-theoretic sum $+$. We assume I to be finite, the empty multiset type is denoted by $[\,]$.

An $\mathscr{R}_0$-**context** (Γ or Δ) is a *total* function from $\mathscr{V}$ (the set of term variables) to the set of multiset types. The **domain**

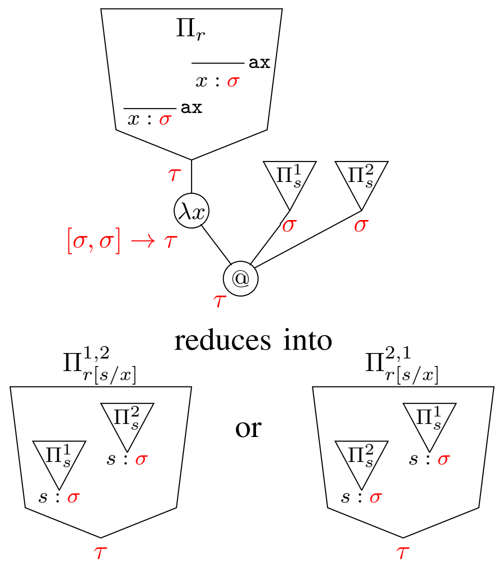


Fig. 1. Non-Determinism of Subject Reduction (Multiset Intersection)

of Γ is given by $\{x \mid \Gamma(x) \neq [\,]\}$. The intersection of contexts $+_{i\in I}\Gamma_i$ is defined point-wise. We 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 $\mathscr{R}_0$-**judgment** is a triple $\Gamma\vdash t:\sigma$ where Γ is an $\mathscr{R}_0$-context, t a term and σ an $\mathscr{R}_0$-type. The set of $\mathscr{R}_0$-derivations is defined by the following rules:

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

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

Here is an example of $\mathscr{R}_0$-derivation, that we call $\Pi_{\mathtt{ex}}$ (the left-hand sides of $\mathtt{ax}$-rules are omitted):

$$\frac{\dfrac{\dfrac{}{x:[o,o',o]\to o'}\,\mathtt{ax}\quad \dfrac{}{x:o}\,\mathtt{ax}\quad \dfrac{}{x:o'}\,\mathtt{ax}\quad \dfrac{}{x:o}\,\mathtt{ax}}{x:[o',[o,o',o]\to o',o,o]\vdash xx:o'}\,\mathtt{app}}{\vdash \lambda x.xx:[o',[o,o',o]\to o',o,o]\to o'}\,\mathtt{app}$$

We write $\Pi \rhd_{\mathscr{R}_0} \Gamma\vdash t:\tau$ to mean that the $\mathscr{R}_0$-derivation Π concludes with the judgment $\Gamma\vdash t:\tau$ and $\rhd\Gamma\vdash t:\tau$ to mean that $\Gamma\vdash t:\tau$ is derivable. No weakening is allowed (*relevance*). System $\mathscr{R}_0$ enjoys both **subject reduction** and **expansion**, meaning that types are invariant under (anti)reduction (if $t\to t'$, then $\rhd\Gamma\vdash t:\tau$ iff $\rhd\Gamma\vdash t':\tau$). A fundamental feature of this system is that a term is head normalizing iff it is $\mathscr{R}_0$-typable.

The $\mathtt{app}$-rule of $\mathscr{R}_0$ is based upon multiset equality and can be restated as follows:

$$\frac{\Gamma\vdash t:[\sigma_i]_{i\in I}\to\tau \quad (\Delta_i\vdash u:\sigma'_i)_{i\in I'} \quad [\sigma_i]_{i\in I}=[\sigma'_i]_{i\in I'}}{\Gamma+(+_{i\in I'}\Delta_i)\vdash t\,u:\tau}\,\mathtt{app}$$

System $\mathscr{R}_0$ has a coinductive version that we call system $\mathscr{R}$. To define $\mathscr{R}$, we just interpret the rules of the type grammar coinductively: in system $\mathscr{R}$, $[\sigma_i]_{i\in I}$ can be of infinite cardinality (I is assumed to be countable). Moreover, there may be an infinite number of nestings *e.g.*, in the $\mathscr{R}$-type ϕ_o satisfying $\phi_o = [\phi_o] \to o$ for a given $o \in \mathscr{O}$. Note that ϕ_o allows us to type Ω with o: using ax-rule concluding with $x : [\phi_o] \vdash x : \phi_o$, we easily obtain $x : [\phi_o]_\omega \vdash x\,x : o$, then $\vdash \Delta : \phi_o$ and finally $\vdash \Omega : o$ ($[\sigma]_\omega$ contains infinitely many occurrences of σ). The derivable $\mathscr{R}$-judgments correspond to the points of the *infinitary* relation model, which has interesting properties: every λ-term has a non-empty denotation (not only solvable terms as in the finite case) and its arity can be captured by this model (Theorems 1 and 2 of [16]). It also gives linear representations of every λ-term. A formal definition of $\mathscr{R}$-types may be found at the end of Sec. II-C.

For the time being, let us understand why subject reduction is not deterministic in system $\mathscr{R}_0$: in Fig. 1, a redex $(\lambda x.r)s$ is typed with τ and in r, x is assigned *twice* the type σ in two different places. Then, to obtain a derivation Π' typing the reduct $r[s/x]$, we just replace each axiom rule by an argument derivation Π^i_s $(i = 1, 2)$ concluding with $s : \sigma$. There are two possibilities (**reduction choices**) represented on the bottom of the figure. In general, we write $\Pi \xrightarrow{b}_\beta \Pi'$ if Π' is a derivation obtained from Π by reducing a redex at position b.

B. Tracks and Labelled Trees

Let us now recall the formalism that we use for possibly infinite labelled trees and sequences [14]. Let $\mathbb{N}^*$ be the set of finite words on $\mathbb{N}$, the operator $\cdot$ denotes concatenation, ε the empty word and $\leqslant$ the prefix order *e.g.*, $2\cdot1\cdot3\cdot7 \in \mathbb{N}^*$, $2\cdot1 \leqslant 2\cdot1\cdot3\cdot7$. As a letter of $\mathbb{N}$, a natural number is called a **track** and, for reasons to appear, a track $k \geqslant 2$ is called an **argument track** (argument tracks are *mutable*, Sec. IV-A). 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: $\mathtt{supp}(x) = \{\varepsilon\}$, $\mathtt{supp}(\lambda x.t) = \{\varepsilon\} \cup 0 \cdot \mathtt{supp}(t)$ and $\mathtt{supp}(t\,u) = \{\varepsilon\} \cup 1 \cdot \mathtt{supp}(t) \cup 2 \cdot \mathtt{supp}(u)$. If $a \in \mathbb{N}^*$ and $\overline{a} \in \mathtt{supp}(t)$, 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 $t(0\cdot1) = y$ with $t = \lambda x.y\,x$. The **applicative depth** $\mathtt{ad}(a)$ of a is the number of argument tracks that it contains *e.g.*, $\mathtt{ad}(0{\cdot}3{\cdot}2{\cdot}1{\cdot}1) = 2$ and $\mathtt{ad}(0{\cdot}1{\cdot}0{\cdot}0{\cdot}1) = 0$.

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 $a \in A$). The support of a term is a tree. A **forest** is a set of the form $A \backslash \{\varepsilon\}$ for some tree A such that $0, 1 \notin A$. Formally, a **labelled tree** T (resp. **labelled forest** F) is a function to a set Σ, whose domain, called its **support** $\mathtt{supp}(T)$ (resp. $\mathtt{supp}(F)$), is a tree (resp. a forest). If F is a labelled forest, the set of **roots** of F is defined by $\mathtt{Rt}(F) = \mathtt{supp}(F) \cap \mathbb{N}$. The metavariable U denotes either (labelled or not) a tree or a forest. then $U|_a$ is the function defined on $\{a_0 \in \mathbb{N}^* \,|\, a{\cdot}a_0 \in \mathtt{supp}(U)\}$ and $U|_a(a_0) = U(a{\cdot}a_0)$. If U is a tree (resp. forest and $a \neq \varepsilon$), then $U|_a$ is a tree.

Definition 1. *Let U_1 and U_2 be two (labelled or not) trees or forests. A **01-isomorphism** ϕ from U_1 to U_2 is a bijection from $\mathtt{supp}(U_1)$ to $\mathtt{supp}(U_2)$ such that:*

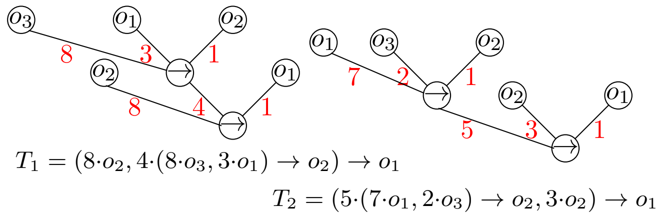

o_3
o_1
o_2
8
3
1
o_2
o_1
8
4
1
$\to$
$\to$
o_1
o_3
o_2
7
2
1
o_2
o_1
5
3
1
$\to$
$\to$
$T_1 = (8{\cdot}o_2, 4{\cdot}(8{\cdot}o_3, 3{\cdot}o_1) \to o_2) \to o_1$
$T_2 = (5{\cdot}(7{\cdot}o_1, 2{\cdot}o_3) \to o_2, 3{\cdot}o_2) \to o_1$

Fig. 2. 01-Isomorphic Labelled Trees (S-Types)

- *ϕ is monotonic for the prefix order and preserves length.*
- *If $a{\cdot}k \in \mathtt{supp}(U_1)$ with $a \in \mathbb{N}^*$ and $k = 0, 1$, then $\phi(a \cdot k) = \phi(a){\cdot}k$.*
- *For all $a \in \mathtt{supp}(U_1)$, $U_2(\phi(a)) = U_1(a)$ (labelled case).*

We write $U_1 \equiv U_2$ when U_1 and U_2 are 01-isomorphic. In Fig. 2, we see two isomorphic labelled trees (which are two S-types, Sec. II-C) w.r.t. ϕ defined by $\phi(\varepsilon) = \varepsilon$, $\phi(1) = 1$, $\phi(4) = 5$, $\phi(4{\cdot}1) = 5{\cdot}1$, $\phi(4{\cdot}3) = 5{\cdot}7$, $\phi(4{\cdot}8) = 5{\cdot}2$, $\phi(8) = 3$. Let F and F' be two 01-isomorphic (labelled) forests. A **root isomorphism** is a function ρ from $\mathtt{Rt}(F)$ to $\mathtt{Rt}(F')$ such that, for all $k \in \mathtt{Rt}(F)$, $F|_k \equiv F'|_{\rho(k)}$ *i.e.* a bijection from $\mathtt{Rt}(F)$ to $\mathtt{Rt}(F')$ that can be extended to a 01-isomorphic from F to F'. Conversely, every isomorphism ϕ from F to F' induces a root isomorphism from F to F', denoted $\mathtt{Rt}(\phi)$.

C. Rigid Types

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 $()$. 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 $F = (2 \cdot o, 5 \cdot o', 8 \cdot o)$. 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 $\mathtt{Rt}(F_1) \cap \mathtt{Rt}(F_2) = \{3\} \neq \emptyset$. Thus, $\uplus$ is not total, but it is associative and commutative. We often abbreviate $(k \cdot S_k)_{k \in K}$ into $(S_k)_{k \in K}$ or $(S_k)_K$.

The **support of a type** (resp. **a sequence type**), which is a tree of $\mathbb{N}^*$ (resp. a forest), is defined by mutual coinduction: $\mathtt{supp}(o) = \{\varepsilon\}$, $\mathtt{supp}(F \to T) = \{\varepsilon\} \cup \mathtt{supp}(F) \cup 1{\cdot}\mathtt{supp}(T)$ and $\mathtt{supp}((T_k)_{k \in K}) = \cup_{k \in K} k \cdot \mathtt{supp}(T_k)$. For instance, $\mathtt{supp}((2 \cdot o, 7 \cdot o') \to o) = \{\varepsilon, 1, 2, 7\}$. Thus, types (resp. sequence types) can naturally be seen as labelled trees (resp. labelled forests) in the sense of Sec. II-B. See Fig. 2.

When we forget about tracks, a finite rigid type collapses to an $\mathscr{R}_0$-type *e.g.*, $(7{\cdot}o_1, 3{\cdot}o_2, 2{\cdot}o_1) \to o$, $(9{\cdot}o_2, 7{\cdot}o_1, 6{\cdot}o_1) \to o$ or $(7 \cdot o_2, 3 \cdot o_1, 2 \cdot o_1) \to o$ all collapse to $[o_1, o_2, o_1] \to o$. The formal definition of $\mathscr{R}$-types (Sec. II-A) as collapses of rigid types needs the following notion:

Definition 2. *Let U_1 and U_2 be two (sequence) types. A **(sequence) type isomorphism** from U_1 to U_2 is a 01-isomorphism from U_1 to U_2.*

For instance, in Fig. 2, both T_1 and T_2 collapse to $[o_2, [o_1, o_3] \to o_2] \to o_1$. A similar notion of isomorphism exists in [12], III.A., which is presented inductively.

The set of $\mathscr{R}$-types can be formally defined as the quotient of the set of $\mathtt{S}$-types by $\equiv$. Multisets types of system $\mathscr{R}$ are the $\equiv$-equivalence classes of sequence types and thus, they indeed allow infinite cardinality and nestings. System $\mathscr{R}$ is defined as $\mathscr{R}_0$ (Sec. II-A), except that we use types of $\mathscr{R}$ instead of just finite types of $\mathscr{R}_0$. A countable version of the binary operator $+$ can be easily defined.

D. *Trivial Rigid Derivations*

An $\mathtt{S}$-context C (or D) is a total function from $\mathscr{V}$ to the set of $\mathtt{S}$-types. The operator $\uplus$ is extended point-wise. An $\mathtt{S}$-judgment is a triple $C \vdash t : T$, where C, t and T are respectively an $\mathtt{S}$-context, a term and an $\mathtt{S}$-type. A **sequence judgment** is a sequence of judgments $(C_k \vdash t : T_k \ \ [k])_{k\in K}$ with $K \subseteq \mathbb{N} \setminus \{0,1\}$. The notation $[k]$ emphasizes that the index k matters in the syntax, but we often just write $(C_k \vdash t : T_k)_{k\in K}$. For instance, if $5 \in K$, then the *judgment on track* 5 in the sequence is $C_5 \vdash t : S_5$.

The set of $\mathtt{S}$-derivations is defined inductively by:

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

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

The $\mathtt{app}$-rule can be applied only if there are no track conflicts in the context $C \uplus (\uplus_{k\in K} D_k)$. In an $\mathtt{ax}$-rule concluding with $x : (k\cdot T) \vdash x : T$, the track k is called the **axiom track** of this axiom rule. In order to gain space, we write $k \vdash x : T$ (with $k \geqslant 2$, $x \in \mathscr{V}$ and T an $\mathtt{S}$-type) instead of $x : (k \cdot T) \vdash x : T$ in $\mathtt{ax}$-rules. Consider the $\mathtt{S}$-derivation $P_{\mathtt{ex}}$ of Fig. 3. In the $\mathtt{ax}$-rule concluding with $x : (5\cdot o) \vdash x : o$, the axiom track is 5. As hinted at before, $\mathtt{S}$-derivations collapse to $\mathscr{R}$-derivations *e.g.*, $P_{\mathtt{ex}}$ collapses to $\Pi_{\mathtt{ex}}$ from Sec. II-A. Moreover:

Proposition 1 ([14]). *Systems $\mathscr{R}$ and $\mathtt{S}$ enjoy subject reduction and expansion.*

The $\mathtt{app}$-rule of system $\mathtt{S}$ can be restated as follows:

$$\frac{C \vdash t : (S_k)_K \to T \quad (D_k \vdash u : S'_k)_{k\in K'} \quad (S_k)_K = (S'_k)_{K'}}{C \uplus (\uplus_{k\in K} D_k) \vdash t\,u : T}\,\mathtt{app}$$

Thus, in system $\mathtt{S}$, the $\mathtt{app}$-rule requires that, for an application $t\,u$ to be typed, $(S_k)_{k\in K}$, the source of the arrow type given to t must be *syntactically* equal to the sequence type $(S'_k)_{k\in K'}$ given to u (compare with Sec. II-A). A consequence of the rigidity is that subject reduction is *deterministic* in system $\mathtt{S}$, as hinted at by Fig. 4 (compare with Fig. 1).

Let us give $\mathtt{S}$ some high-level inputs on pointers in system $\mathtt{S}$ (again, see Sec. III. and IV. of [14] for complements). We define the support of a $\mathtt{S}$-derivation, and also the key notions of biposition and bisupport: the **support** of $P \rhd C \vdash t : T$ is defined inductively by $\mathtt{supp}(P) = \{\varepsilon\}$ if P is an $\mathtt{ax}$-rule, $\mathtt{supp}(P) = \{\varepsilon\} \cup 0 \cdot \mathtt{supp}(P_0)$ if $t = \lambda x.t_0$ and P_0 is the subderivation typing t_0, $\mathtt{supp}(P) = \{\varepsilon\} \cup 1 \cdot \mathtt{supp}(P_1) \cup_{k\in K} k \cdot \mathtt{supp}(P_k)$ 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, $\mathtt{supp}(P_{\mathtt{ex}}) = \{\varepsilon, 0, 0\cdot 1, 0\cdot 2, 0\cdot 3, 0\cdot 8\}$, $P_{\mathtt{ex}}(0\cdot 1) = x : 4\cdot((8\cdot o, 3\cdot o', 2\cdot o) \to o') \vdash x : (8\cdot o, 3\cdot o', 2\cdot o) \to o'$ and $P_{\mathtt{ex}}(0\cdot 3) = x : (2\cdot o') \vdash x : o'$. If $a \in \mathtt{supp}(P)$, then $P(a)$ is a judgment $C \vdash t : T$. We denote C and T by $\mathtt{C}^P(a)$ and $\mathtt{T}^P(a)$ *e.g.*, $\mathtt{C}^{P_{\mathtt{ex}}}(1\cdot 2) = x : (9 \cdot o)$ and $\mathtt{T}^{P_{\mathtt{ex}}}(1\cdot 2) = o$.

A (right) biposition of an $\mathtt{S}$-derivation P is a pair (a, c), where $a \in \mathtt{supp}(P)$ and $c \in \mathtt{supp}(\mathtt{T}^P(a))$, where the judgment of P at position a is $C \vdash u : T$. In that case, $P(a, c)$ denotes $\mathtt{T}^P(a)(c)$ *e.g.*, $P_{\mathtt{ex}}(1, \varepsilon) = o'$, $P_{\mathtt{ex}}(0\cdot 1, 8) = o$ because $\mathtt{T}^{P_{\mathtt{ex}}}(0\cdot 1) = (8\cdot o, 3\cdot o', 2\cdot o) \to o'$. In this article, we just need to think of bipositions as pairs pointing to type symbols ($o \in \mathscr{O}$ or $\to$) or to labelled edges that are nested in a given rigid derivation. We will do so when informally discussing Fig. 7.

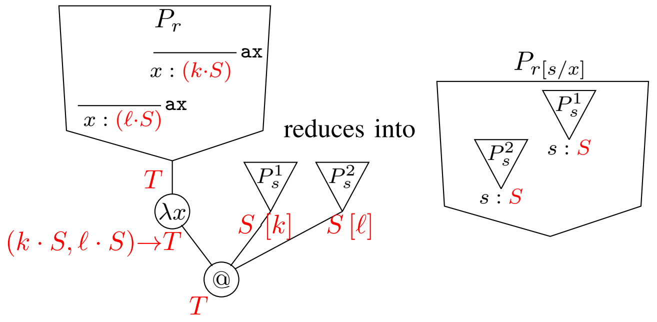


Fig. 4. Deterministic Subject Reduction (Sequential Intersection, $\mathtt{S}$-case)

E. *The Question of Representability*

From the introduction, we recall that we will give a positive answer to this question:

Question: Is every $\mathscr{R}$-derivation the collapse of an $\mathtt{S}$-derivation ?

Attempt 1 (induction on proof structure): if we try to proceed by induction on the structure of an $\mathscr{R}$-derivation Π, we are quickly stuck. For instance, assume that

- $\Pi = \dfrac{\Pi_{\lambda x.r} \rhd \Gamma \vdash \lambda x.r{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Pi_i \rhd \Delta_i \vdash s{:}\sigma_i)_{i\in I}}{\Gamma + (+_{i\in I}\Delta_i) \vdash (\lambda x.r)s : \tau}$
- $\Pi_{\lambda x.r}$ and the Π_i are all $\mathtt{S}$-representable.

The second assumption means that there are $\mathtt{S}$-derivations $P_{\lambda x.r} \rhd C \vdash \lambda x.r : (S_k)_{k\in K} \to T$ and $(P_i \rhd D_i \vdash s : S'_i)_{i\in I'}$ that respectively collapse to $\Pi_{\lambda x.r}$ and the Π_i. But $P_{\lambda x.r}$ and the P_i can be used to represent Π only if (1) we can ensure that the S_k and the S'_i are syntactically equal and (2) we can avoid track conflicts.

For point (1), notice that, given a term t that is $\mathscr{R}$-typable with type τ, there may be some $\mathtt{S}$-types T that collapse to τ such that t is *not* $\mathtt{S}$-typable with T *e.g.*, if $\overline{S} = \sigma = \overline{S'}$, but $S \neq S'$, then $T := (2\cdot S) \to S'$ collapses to $\tau := [\sigma] \to \sigma$ and $I := \lambda x.x$ can be $\mathscr{R}$-typed with τ and $\mathtt{S}$-typed with $(2\cdot S) \to S$ or $(5\cdot S') \to S'$, but *not* with T.

Thus, some typing constraints could *a priori* forbid that we can equalize the S_k and the S'_i since it is difficult to describe the $\mathtt{S}$-types of r and s that collapse to τ or σ_i.

$$P_{\texttt{ex}} = \cfrac{\cfrac{\cfrac{}{4 \vdash x : (8{\cdot}o, 3{\cdot}o', 2{\cdot}o) \to o'}\,\texttt{ax} \quad \cfrac{}{9 \vdash x : o \;[2]}\,\texttt{ax} \quad \cfrac{}{2 \vdash x : o' \;[3]}\,\texttt{ax} \quad \cfrac{}{5 \vdash x : o \;[8]}\,\texttt{ax}}{x : (2{\cdot}o', 4{\cdot}(8{\cdot}o, 3{\cdot}o', 2{\cdot}o) \to o', 5{\cdot}o, 9{\cdot}o) \vdash xx : o'}\,\texttt{app}}{\vdash \lambda x.xx : (2{\cdot}o', 4{\cdot}(8{\cdot}o, 3{\cdot}o', 2{\cdot}o) \to o', 5{\cdot}o, 9{\cdot}o) \to o'}\,\texttt{abs}$$

Fig. 3. An $\mathtt{S}$-derivation

Attempt 2 (expanding normal forms): a fundamental method of intersection types is to type/study normal forms and then use subject expansion to extend the properties of the type system to every term that can be reduced to a normal form. Can we do the same here? Since system $\mathscr{R}$ types every term, it does not ensure any form of normalization/productivity, and the method alluded to cannot work in the general case (it does though in the *finite* one).

The solution is then to reuse the method that we developed in [16], which allows us dealing with non-normalizing terms. The remainder of the paper is then dedicated to answering yes to the above question *i.e.* proving:

Theorem 2 (Representation)**.** *Every $\mathscr{R}$-derivation is the collapse of an* $\mathtt{S}$*-derivation.*

The above discussion suggests that system $\mathtt{S}$ is too constraining for the question of representability to be addressed directly and that we should relax the app-rule. This motivates the definition of **hybrid derivations** by replacing, in system $\mathtt{S}$, the app-rule by:

$$\cfrac{C \vdash t : (S_k)_K \to T \quad (D_k \vdash u : S'_k)_{k \in K'} \quad (S_k)_K \equiv (S'_k)_{K'}}{C \uplus (\uplus_{k \in K} D_k) \vdash t\,u : T}\,\texttt{app}_{\texttt{h}}$$

We call this modified system $\mathtt{S}_{\mathtt{h}}$. We thus relax the syntactic equality of app into equality up to sequence type isomorphism (Def. 2). The notations of system $\mathtt{S}$ generalize to system $\mathtt{S}_{\mathtt{h}}$ *e.g.*, $\mathtt{supp}(P)$, $\mathtt{bisupp}(P)$, $\mathtt{T}^P$ and so on. If, in an $\mathtt{S}_{\mathtt{h}}$-derivation P, the above $\texttt{app}_{\texttt{h}}$-rule occurs at position a, one denotes $(S_k)_{k \in K}$ by $\mathtt{L}^P(a)$ (standing for "Left") and $(S'_k)_{k \in K'}$ by $\mathtt{R}^P(a)$ ("Right"). Then, the $\texttt{app}_{\texttt{h}}$-rule requires that $\mathtt{L}^P(a) \equiv \mathtt{R}^P(a)$ *i.e.* $\mathtt{R}^P(a)$ and $\mathtt{L}^P(a)$ must be isomorphic. Thus, a hybrid derivation P is **trivial** (*i.e.* just an $\mathtt{S}$-derivation) when for all $a \in \mathtt{supp}_{@}(P) := \{a \in \mathtt{supp}(A) \,|\, t(a) = @\}$ (*i.e.* for all app-rules of P), $\mathtt{L}^P(a) = \mathtt{R}^P(a)$. When P is clear from the context, we omit it and just write $\mathtt{L}(a)$ and $\mathtt{R}(a)$.

Fig. 7 gives an example of an $\mathtt{S}_{\mathtt{h}}$-derivation. Note that all the contexts replaced by $\ldots$ can be computed from the given information *e.g.*, $x : (5 \cdot (8{\cdot}o) {\to} (8{\cdot}o, 9{\cdot}o) {\to} o'), y : (3{\cdot}o) \vdash x\,y : (8{\cdot}o, 9{\cdot}o) {\to} o'$ and that the $\texttt{app}_{\texttt{h}}$-rules are correct. Indeed, $\mathtt{L}(\varepsilon) = (8{\cdot}o, 9{\cdot}o) \equiv (3{\cdot}o, 5{\cdot}o) = \mathtt{R}(\varepsilon)$ and $\mathtt{L}(1) = (5{\cdot}(8{\cdot}o) {\to} (8{\cdot}o, 9{\cdot}o) {\to} o') \equiv (6{\cdot}(3{\cdot}o) {\to} (2{\cdot}o, 7{\cdot}o) {\to} o') = \mathtt{R}(1)$.

Observe that rule $\texttt{app}_{\texttt{h}}$ is no more and no less constraining than the app-rule of system $\mathscr{R}$ since $(S_k)_{k \in K} \equiv (S'_k)_{k \in K'}$ iff the multiset types $\overline{(S_k)_{k \in K}}$ and $\overline{(S'_k)_{k \in K'}}$ are equal, according to the end of Sec. II-C. Thus, the problem of the collapse is not difficult in the case of $\mathtt{S}_{\mathtt{h}}$:

Proposition 3. *If Π is an $\mathscr{R}$-derivation, then it is the collapse of an* $\mathtt{S}_{\mathtt{h}}$*-derivation P.*

Proof. By induction on the structure of Π. □

III. Pseudo-Subject Reduction and Interfaces

In this section, we explore the effect of reduction on system $\mathtt{S}_{\mathtt{h}}$, which satisfies a relaxed form of subject reduction and expansion:

Proposition 4 (Pseudo-Subject Reduction and Expansion)**.**

- *If $t \to t'$ and $\rhd_{\mathtt{S}_{\mathtt{h}}} C \vdash t : T$, then $\rhd_{\mathtt{S}_{\mathtt{h}}} C \vdash t' : T'$ for some $T' \equiv T$.*
- *If $t \to t'$ and $\rhd_{\mathtt{S}_{\mathtt{h}}} C \vdash t' : T'$, then $\rhd_{\mathtt{S}_{\mathtt{h}}} C \vdash t : T$ for some $T \equiv T'$.*

Let us now explain how reduction is handled with hybrid derivations and why the type T of t may be replaced by an isomorphic type T' in $\mathtt{S}_{\mathtt{h}}$ after one reduction step. Reduction is not deterministic in system $\mathtt{S}_{\mathtt{h}}$, so we endow hybrid derivations with type isomorphisms, which allows us to encode reduction paths (Sec. III-A and Lemma 5) gives rise to the notion of *operable* derivations.

A. Encoding Reduction Choices with Interfaces

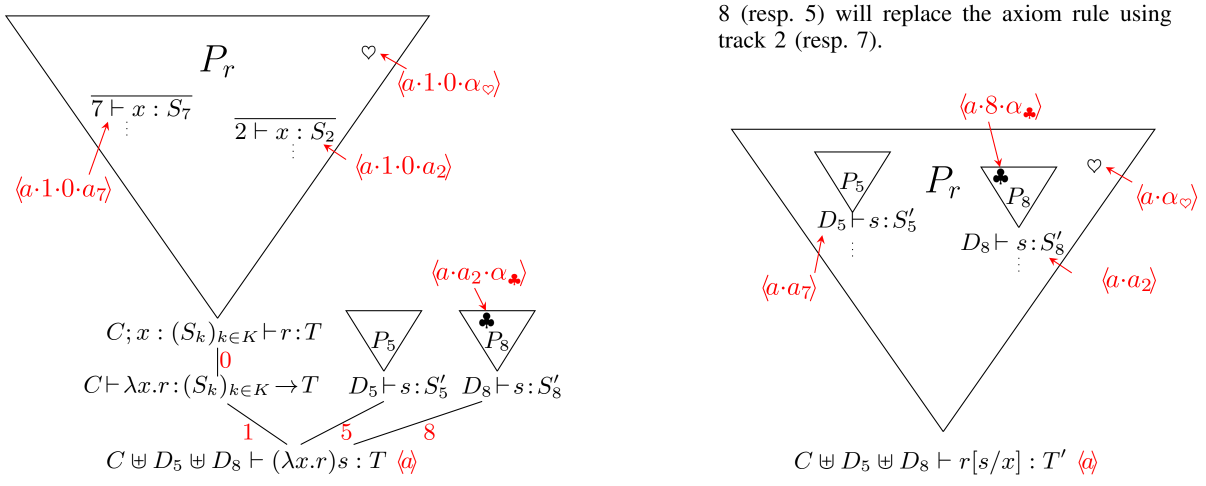
In this section, we explain Prop. 4 *i.e.* reduction in system $\mathtt{S}_{\mathtt{h}}$. We assume that $t|_b = (\lambda x.r)s$, $t \xrightarrow{b} t'$ (so that $t'|_b = r[s/x]$), each axiom rule concluding with $x : (k{\cdot}S_k) \vdash x : S_k$ will be replaced by a subderivation $P_{k'} \rhd D_{k'} \vdash s : S'_{k'}$ satisfying $S'_{k'} \equiv S_k$. However, the $\texttt{app}_{\texttt{h}}$-rule states that there exists an isomorphism from $(S_k)_K$ to $(S'_k)_{K'}$ but does not specify this isomorphism. This entails that there may be many ways to produce P' typing t' from P typing t : in system $\mathtt{S}_{\mathtt{h}}$, there are also reduction choices, as in system $\mathscr{R}$.

All this is illustrated by the left part of Fig. 5: under the same hypotheses, we assume that $a \in \mathtt{supp}(P)$ is such that $\overline{a} = b$ (thus, a is the position of a judgment typing the redex to be fired) and that there are exactly 2 ax-rules typing x above a, using axiom tracks 2 and 7. Notice that the ax-rule typing x on track 7 (assigning S_7) must be above $a{\cdot}1{\cdot}0$, so that its position is of the form $a{\cdot}1{\cdot}0{\cdot}a_7$. Likewise for the other ax-rule assigning S_2 on track 2. We omit ax-rules right-hand sides. We also indicate the position of a judgment between angle brackets *e.g.*, $\langle a{\cdot}1{\cdot}0{\cdot}a_7 \rangle$ means that judgment $x : (7{\cdot}S_7) \vdash x : S_7$ is at position $a{\cdot}1{\cdot}0{\cdot}a_7$.

By typing constraints, there must be two argument derivations typing s with types isomorphic to S_2 and S_7. We assume that those two argument derivations are on track 5 and 8 and

Assumptions: $\rho_a(2) = 8$, $\rho_a(7) = 5$ (so that $S_2 \equiv S'_8$, $S_7 \equiv S'_5$)

Comment: since $\rho_a(2) = 8$, $\rho_a(7) = 5$, the argument subderivation P_8 (resp. P_5) on track 8 (resp. 5) will replace the axiom rule using track 2 (resp. 7).



P_r
$7 \vdash x : S_7$
$2 \vdash x : S_2$
$\langle a{\cdot}1{\cdot}0{\cdot}\alpha_\heartsuit\rangle$
$\langle a{\cdot}1{\cdot}0{\cdot}a_2\rangle$
$\langle a{\cdot}1{\cdot}0{\cdot}a_7\rangle$
$\langle a{\cdot}a_2{\cdot}\alpha_\clubsuit\rangle$
$C; x : (S_k)_{k\in K} \vdash r : T$
0
$C \vdash \lambda x.r : (S_k)_{k\in K} \to T$
P_5
P_8
$D_5 \vdash s : S'_5$
$D_8 \vdash s : S'_8$
1
5
8
$C \uplus D_5 \uplus D_8 \vdash (\lambda x.r)s : T$ $\langle a\rangle$
$\langle a{\cdot}8{\cdot}\alpha_\clubsuit\rangle$
$\langle a{\cdot}\alpha_\heartsuit\rangle$
$\langle a{\cdot}a_7\rangle$
$\langle a{\cdot}a_2\rangle$
$D_5 \vdash s : S'_5$
$D_8 \vdash s : S'_8$
$C \uplus D_5 \uplus D_8 \vdash r[s/x] : T'$ $\langle a\rangle$

Subderivation typing the redex **Subderivation typing the reduct**

Fig. 5. Subject Reduction and Residuals

conclude with $D_k \vdash s : S'_k$ $(k = 5, 8)$ where *e.g.*, $S_2 \equiv S'_8$ and $S_7 \equiv S'_5$. If moreover, $S_2 \equiv S_7$, then each $\mathtt{ax}$-rule typing x can be replaced by P_5 as well as by P_8: there is a reduction choice. In all cases, the type of $r[s/x]$ may change *e.g.*, if $x : S_7$ corresponds to the head variable of r, then S_7 is replaced by an isomorphic type S_5 (or S_8), so T may also be replaced by an isomorphic T'.

We now use **root type isomorphisms** (Sec. II-B, Def. 2) to retrieve some determinism and represent particular reduction choices in system $\mathtt{S_h}$. Let P be a hybrid derivation:

- Let $a \in \mathtt{supp}_@(P)$. A **root interface** (resp. an **interface**) at position a is a root isomorphism (resp. a sequence type isomorphism) from $\mathtt{L}^P(a)$ to $\mathtt{R}^P(a)$.
- Let $b \in \mathtt{supp}(t)$ such that $t(b) = @$. A **total (root) interface** at position b is a family of (root) interfaces for all $a \in \mathtt{supp}(P)$ s.t. $\overline{a} = b$.
- A **total interface** is the datum for an interface ϕ_a for all $a \in \mathtt{supp}_@(P)$.

For all $a \in \mathtt{supp}_@(P)$, we write $\mathtt{Inter}^P(a)$ for the set of interfaces at pos. a in P. This allows defining:

Definition 3. *An **operable derivation** is a hybrid derivation endowed with a total interface.*

Definition 3 corresponds to a new system $\mathtt{S_{op}}$, which has rules $\mathtt{ax}$, $\mathtt{abs}$ and an $\mathtt{app}$-rule parametrized with a sequence type isomorphism ϕ:

$$\frac{C \vdash t : (S_k)_K \to T \quad (D_k \vdash u : S'_k)_{K'} \quad \phi : (S_k)_K \overset{\text{iso.}}{\to} (S'_k)_{K'}}{C \uplus (\uplus_{k\in K} D_k) \vdash t\, u : T}\,\mathtt{app}_\phi$$

The restriction of $\mathtt{S_{op}}$ to *finite* types gives a counterpart of [12] rigid refinement intersection type systems based on sequences instead of lists and with the difference that we do not specify a resource calculus (which is implicit here).

Assuming that $S_2 \equiv S_7$ in Fig. 5, we have seen above that the two $\mathtt{ax}$-rules typing x could be indifferently replaced by P_5 or P_8. But if a is endowed with the root interface ρ_a s.t. $\rho_a(2) = 8$ and $\rho_a(7) = 5$, then the $\mathtt{ax}$-rule typing x on track 2 *must* be replaced by P_8 and the other one by P_5, as on the right part of the figure. Indeed, a root interface at position b specifies a reduction choice at position b. Compare this process with Fig. 4: implicitly, a trivial derivation P is endowed with a trivial interface (ϕ_a is the *identity* from $\mathtt{L}^P(a)$ to $\mathtt{R}^P(a)$ for all $a \in \mathtt{supp}_@(P)$) and there is no way, in system $\mathtt{S}$, to specify that P^1_s should replace $x : (\ell \cdot S)$ instead of $x : (k \cdot S)$.

B. Residuation

In Sec. III-A, we explained how to capture a *one-step* reduction choice with a root interface. To capture every $\mathscr{R}$-reduction path (of any length $\leqslant \infty$) with operable derivation *i.e.* with *total* interfaces (what we do in § III-C), we elaborate in this section a residuation theory for system $\mathtt{S_h}$.

Root interfaces allow us defining a suitable notion of **residuals** for **positions** (the residual of $\alpha \in \mathtt{supp}(P)$, if it exists, is a $\alpha' \in \mathtt{supp}(P')$ that may be denoted $\mathtt{Res}^\rho_b(\alpha')$ since it depends both on b and $(\rho)_{\overline{a}=b}$).

Now, if instead of endowing P with a total *root* interface $(\rho_a)_{\overline{a}=b}$ at position b, we endow it with a total interface $(\phi_a)_{\overline{a}=b}$ at position b, we define below a notion of **residuals** for right bipositions, as we did for system $\mathtt{S}$ (Sec. IV.D and Fig. 1 in [14]): the residual of (α, c) may be then denoted $\mathtt{Res}^\phi_b(\alpha, c)$.

Interestingly, in that case, residuation can also be defined for *interfaces*: more precisely, if P is endowed with a total interface at position b and $\alpha \in \mathtt{supp}_@(P)$ is such that $\overline{\alpha} \neq b$, then α has a residual $\alpha' := \mathtt{Res}^\phi_b(\alpha)$ w.r.t. ϕ and there is a bijection $\mathtt{ResI}^\phi_b(\alpha)$ from $\mathtt{Inter}^P(\alpha)$ to $\mathtt{Inter}^{P'}(\alpha')$.

The remainder of Sec. III-B is technical and sketches the construction of this various notions of residuation (positions, bipositions, interfaces). We then assume that $P \rhd C \vdash t : T$, $t|_b = (\lambda x.r)s$ and $t \xrightarrow{b}_\beta t'$ (thus, $t'|_b = r[s/x]$).

Conventions on metavariables a and α The letter a will stand for a position of P that corresponds to the root of the redex (*i.e.* $a \in \mathtt{supp}(P)$ and $\overline{a} = b$) and the letter α for other positions in $A := \mathtt{supp}(P)$ or in $\mathbb{N}^*$. When $\overline{a} = b$, $\mathtt{Ax}_\lambda(a)$ denotes the axiom rules above a types the variable x of the redex and we set $\mathtt{Tr}_\lambda(a) = \{\mathtt{tr}^P(\alpha_0) \,|\, \alpha_0 \in \mathtt{Ax}_\lambda(a)\} = \mathtt{Rt}(\mathtt{T}^P(a \cdot 1))$, so that $\mathtt{Tr}_\lambda(a)$ is the set of the axiom tracks used above a to type the variable of the redex. For instance, in Fig. 5, $\mathtt{Ax}_\lambda(a) = \{a{\cdot}1{\cdot}0{\cdot}a_2,\ a{\cdot}1{\cdot}0{\cdot}a_7\}$ and $\mathtt{Tr}_\lambda(a) = \{2, 7\}$.

Positions: We first assume that P is endowed with a root interface $(\rho_a)_{\overline{a}=b}$ at position b. We define the residual position $\mathtt{Res}_b(\alpha)$ for each $\alpha \in \mathtt{supp}(P)$ except when α is of the form a, $a \cdot 1$ or $a \cdot 1{\cdot}0 \cdot a_k$ (for some a satisfying $\overline{a} = b$). We begin with discussing the symbols $\heartsuit$ and $\clubsuit$ in Fig.5. In Fig. 5, $\heartsuit$ represents a judgment nested in P_r. Thus, its position must be of the form $a \cdot 1{\cdot}0 \cdot \alpha_\heartsuit$. After reduction, the app-rule and abs-rule at positions a and $a \cdot 0$ have been destroyed and the position of this judgment $\heartsuit$ will be $a \cdot \alpha_\heartsuit$. We set then $\mathtt{Res}_b(a \cdot 1{\cdot}0 \cdot \alpha_\heartsuit) = a \cdot \alpha_\heartsuit$.

Likewise, $\clubsuit$ represents a judgment nested in the argument derivation P_8 on track 8 w.r.t. a. Thus, its position must be of the form $a{\cdot}8{\cdot}\alpha_\clubsuit$ where $a{\cdot}8$ is the root of P_8. After reduction, P_8 will replace the ax-rule typing x on track $\rho_a^{-1}(8)$ *i.e.* 2, so its root will be at $a \cdot a_2$ (by definition of a_2). Thus, after reduction, the position of judgment $\clubsuit$ will be $a \cdot a_2 \cdot \alpha_\clubsuit$. We set then $\mathtt{Res}_b(a \cdot 8 \cdot \alpha_\clubsuit) = a \cdot a_2 \cdot \alpha_\clubsuit$.

- Paradigm $\clubsuit$: if $\alpha = a \cdot k_{\mathtt{R}} \cdot \alpha_0$ where $\overline{a} = b$ and $k_{\mathtt{R}} \geqslant 2$, then $\mathtt{Res}_b(\alpha) = a \cdot a_{k_{\mathtt{L}}} \cdot \alpha_0$ with $k_{\mathtt{L}} = \rho_a^{-1}(k_{\mathtt{R}})$.
- Paradigm $\heartsuit$: if $\alpha = a{\cdot}1{\cdot}0{\cdot}\alpha_0$ where $a \in \mathtt{supp}(P), \overline{a} = b$ and $\alpha_0 \neq a_k$, then $\mathtt{Res}_b(\alpha) = a \cdot \alpha_0$.
- Outside the redex: if $b \not\leqslant \overline{\alpha}$, then $\mathtt{Res}_b(\alpha) = \alpha$

Residuation for positions easily allows defining, from $(\rho_a)_{\overline{a}=b}$, a *unique* derivation $P' \rhd C \vdash t' : T'$ with $T \equiv T'$, as expected. Derivation P' is denoted $\mathtt{Res}_b^\rho(P)$. The function $\mathtt{Res}_b^\rho$ is a partial injection from $\mathtt{supp}(P)$ to $\mathtt{supp}(P')$.

Bipositions: when b is endowed with a total interface $(\phi_a)_{\overline{a}=b}$ inducing a root interface $(\rho_a)_{\overline{a}=b}$, the residual $\mathtt{Res}_b^\phi(\alpha, \gamma)$ of a bipositions is defined by induction on a (starting at axiom rules and going downwards). By lack of space, we give only the base cases. The full construction (which can be found in Sec. B.1. of [15]) is long but quite natural.

- If $\alpha \notin \cup_{\overline{a}=b} \mathtt{Ax}_\lambda(a)$ *i.e.* α is an ax-rule that does not type the variable of the redex, then $\mathtt{Res}_b^\phi(\alpha, \gamma) = (\alpha', \gamma)$ with $\alpha' = \mathtt{Res}_b^\rho(\alpha)$.
- If $\overline{a} = b$, $\alpha \in \mathtt{Ax}_\lambda(a)$ and $\ell = \mathtt{tr}(\alpha)$ *i.e.* α is an ax-rules typing the variable of the redex, then $\mathtt{Res}_b(\alpha, \gamma) = (\alpha', \gamma')$ with $\alpha' = \mathtt{Res}_b^\phi(\alpha)$, $\ell' = \rho_\alpha(\ell)$ and $\ell' \cdot \gamma' = \ell \cdot \phi_\alpha(\gamma)$. An important observation is that this definition is correct, because $\gamma \in \mathtt{supp}(T^P(\alpha))$ iff $\ell \cdot \gamma \in \mathtt{supp}(\mathtt{L}^P(a))$. Moreover, in this case, defining $\mathtt{Res}_b(\alpha, \gamma)$ would not be possible with the datum of $(\rho_a)_{\overline{a}=b}$ only.

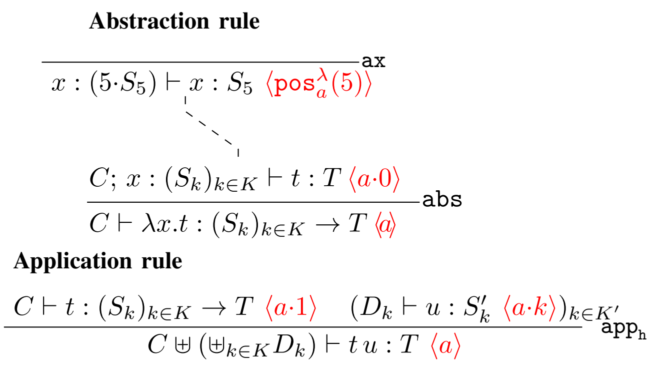

Abstraction rule
$x : (5{\cdot}S_5) \vdash x : S_5\ \langle \mathtt{pos}_a^\lambda(5) \rangle$
ax
$C;\, x : (S_k)_{k \in K} \vdash t : T\ \langle a{\cdot}0 \rangle$
abs
$C \vdash \lambda x.t : (S_k)_{k \in K} \to T\ \langle a \rangle$
Application rule
$C \vdash t : (S_k)_{k \in K} \to T\ \langle a{\cdot}1 \rangle \quad (D_k \vdash u : S'_k\ \langle a{\cdot}k \rangle)_{k \in K'}$
$\mathtt{app_h}$
$C \uplus (\uplus_{k \in K} D_k) \vdash t\,u : T\ \langle a \rangle$
endowed with an interface $\phi_a : (S_k)_{k \in K} \to (S'_k)_{k \in K'}$

Fig. 6. Ascendance and Polar Inversion

The construction of $\mathtt{Res}_b(\cdot, \cdot)$ induces, for all $\alpha \in \mathtt{supp}(A)$, a type isomorphism from $\mathtt{T}^P(\alpha)$ to $\mathtt{T}^{P'}(\alpha')$ with $\alpha' = \mathtt{Res}_b^\phi(\alpha)$ *i.e.* ϕ maps types from the initial derivation P to isomorphic types in the reduct derivation $P' = \mathtt{Res}_b^\phi(P) := \mathtt{Res}_b^\rho(P)$.

Interface isomorphisms: Assume that P is endowed with a total interface $(\phi_a)_{\overline{a}b}$. Then it is easy to check that:

- $\mathtt{Res}_a^\phi$ is a *bijection* from $\mathtt{supp}_@(P) \setminus \{a \,|\, \overline{a} = b\}$ to $\mathtt{supp}_@(P')$ with $P' = \mathtt{Res}_b^\phi(P)$.
- For all $\alpha \in \mathtt{supp}_@(P)$, $\alpha' = \mathtt{Res}_b^\phi$, the function $\mathtt{Res}_b(\cdot, \cdot)$ induces a sequence type isomorphism $\mathtt{ResL}_{b|\alpha}^P$ from $\mathtt{L}^P(\alpha)$ to $\mathtt{L}^{P'}(\alpha')$ (resp. $\mathtt{ResR}_{b|\alpha}^P$ from $\mathtt{R}^P(\alpha)$ to $\mathtt{R}^{P'}(\alpha')$) *i.e.* residuation maps the left-hand sides (resp. the right-hand sides) of interfaces of the initial derivation P to isomorphic left-hand sides (resp. right-hand sides) of interfaces of the reduct derivation P'.

Then, for each application nodes $\alpha \in \mathtt{supp}_@(P)$, $\alpha' \in \mathtt{supp}_@(P')$ with $\alpha' = \mathtt{Res}_b(\alpha)$ and $\overline{\alpha} \neq b$ (*i.e.* α is not the application of the redex), there is a canonical bijection $\mathtt{ResI}_{b|\alpha}^\phi$ from $\mathtt{Inter}^P(\alpha)$ to $\mathtt{Inter}^{P'}(\alpha')$ defined by $\mathtt{ResI}_{b|\alpha}(\psi) = \mathtt{ResR}_{b|\alpha} \circ \psi \circ \mathtt{ResL}_{b|\alpha}^{-1}$, so that the following diagram commutes:

$$\begin{tikzcd}
\mathtt{L}^P(\alpha) \arrow[r, "\psi"] \arrow[d, "\mathtt{ResL}_{b|\alpha}^\phi"'] & \mathtt{R}^P(\alpha) \arrow[d, "\mathtt{ResR}_{b|\alpha}^\phi"] \\
\mathtt{L}^{P'}(\alpha') \arrow[r, "\mathtt{ResI}_{b|\alpha}^\phi(\psi)"'] & \mathtt{R}^{P'}(\alpha')
\end{tikzcd}$$

When P is endowed with a total interface (ϕ_a) ($a \in \mathtt{supp}_@(P)$), which gives a unique $P' = \mathtt{Res}_b^\phi$, the function $\mathtt{ResI}$ allows defining the residual interface of ϕ on P' by setting, for all $\alpha' \in \mathtt{supp}_@(P')$, $\alpha \in \mathtt{supp}_@(P)$ with $\alpha' = \mathtt{Res}_b^\phi(\alpha)$ and $\overline{a} \neq b$, $\phi'_{\alpha'} := \mathtt{ResI}_{\alpha|b}^{\hat{\phi}}(\phi_\alpha)$ where $\hat{\phi} = (\phi_a)_{\overline{a}=b}$. In others words, $\phi'_{\alpha'} = \mathtt{ResR}_{b|\alpha}^\phi \circ \phi_\alpha \circ (\mathtt{ResL}_{b|\alpha}^\phi)^{-1}$.

C. *Capturing Full Reduction Paths*

We may now explain why every reduction path starting at a $\mathscr{R}$-derivation Π may be capture by means of an operable derivation. Assume that $\Pi_0 \rhd \Gamma \vdash t : \tau$ and $\Pi_0 \xrightarrow{b_0}_\beta \Pi_1 \ldots \Pi_n \xrightarrow{b_n}_\beta P_n \ldots$ is a reduction sequence of length $\ell \leqslant \infty$.

Let P be a hybrid derivation collapsing on Π (Proposition 3). Then, one builds by induction on n a sequence $A_n \subseteq A_{n+1} \subseteq A := \mathtt{supp}(P)$ along with interface isomorphisms $(\phi_a)_{a \in A_n}$ of P such that, for all $n \in \mathbb{N}$, $(\phi_a)_{a \in A_n}$ captures the reduction sequence until step n and gives a residual derivation P_n collapsing on Π_n: we set $P_0 = P$ and $A_0 = \emptyset$ and to define (ϕ_a) for $a \in A_{n+1} \setminus A_n$, one first define an interface $(\psi_a)_{\overline{a}=b, a \in \mathtt{supp}(P_n)}$ in P_n at position b_n capturing the step $\Pi_n \xrightarrow{b_n}_\beta \Pi_{n+1}$. Then, applying $\mathtt{ResI}^{-1}$ n times on the ψ_a, one obtains interface isomorphisms $(\phi_a)_{a \in C}$ in P. We then set $A_{n+1} = A_n \cup C$ and $P_{n+1} = \mathtt{Res}^{\psi}_{b_n}(P_n)$. We thus obtain:

Lemma 5 (Representation)**.** *Every sequence of reduction choices of length $\leqslant \omega$ in a derivation Π can be built-in in an operable derivation P collapsing on Π.*

The above lemma epitomizes the use of a rigid paradigm along with type isomorphism to capture reduction paths in a non-deterministic system, as shown in [12].

IV. Representation Theorem

In this final section, we sketch the proof of the main theorem of this article: every $\mathscr{R}$-derivation is the collapse of an $\mathtt{S}$-derivation. First, we need to render this statement more handleable by considering *isomorphisms of derivations*.

Since a hybrid derivation P is a tree of $\mathbb{N}^*$ that is labelled with rigid judgments, and for all $a \in \mathtt{supp}(P)$, $\mathtt{T}^P(a)$ (which is a type of $t|_a$ in P) is also a labelled tree of $\mathbb{N}^*$, it is easy to define the notion of isomorphism from one hybrid derivation P_1 to another P_2 by using suitably the notion of 01-isomorphism (Sec. II-B). The formal definition given below ensures that two hybrid derivations P_1 and P_2 are isomorphic iff they collapse to the same $\mathscr{R}$-derivation. An isomorphism Φ from a $\mathtt{S_h}$-derivation P_1 to another P_2 is the datum of:

- A 01-isomorphism of unlabelled tree from $\mathtt{supp}(P_1)$ to $\mathtt{supp}(P_2)$, $\Psi_{\mathtt{supp}}$.
- For all axiom rules $a \in \mathtt{supp}(P_1)$, a type isomorphism Ψ_a from $\mathtt{T}^P(a)$ to $\mathtt{T}^{P'}(a')$ with $a' = \Psi_{\mathtt{supp}}(a)$.

We do not give the details of the well-foundedness of this definition, which are not difficult. For instance, one needs to check that that $\Psi_{\mathtt{supp}}$ induces a bijection from the set of axiom rules of P to that of P, or from $\mathtt{supp}_@(P)$ to $\mathtt{supp}_@(P')$. Moreover, this definition entails that such a Ψ induces, for all $a_1 \in \mathtt{supp}(P_1)$, a sequence type isomorphism $\Psi^{\mathtt{L}}_{a_1}$ from $\mathtt{L}^{P_1}(a_1)$ to $\mathtt{L}^{P_2}(a_2)$ and another from $\Psi^{\mathtt{R}}_{a_1}$ from $\mathtt{R}^{P_1}(a_1)$ to $\mathtt{R}^{P_2}(a_2)$ with $a_2 = \Psi_{\mathtt{supp}}(a_1)$.

From Sec. III-A, we recall that an operable derivation is a hybrid derivation P endowed with a total interface $(\phi_a)_{a \in \mathtt{supp}_@(P)}$. Some isomorphisms of hybrid derivations induce **isomorphism of operable derivations**. Morally, if P_1 and P_2 are *operably* isomorphic then they collapse to the same $\mathscr{R}$-derivation and encode the same $\mathscr{R}$-reduction path. Formally, if P_1 and P_2 are operable derivations (resp. endowed with (ϕ^1_a) and (ϕ^2_a)), then an isomorphism Ψ of hybrid derivations from P_1 to P_2 is actually an isomorphism of operable derivations if Ψ commutes with the interfaces of P_1 and P_2 *i.e.* for all $a_i \in \mathtt{supp}_@(P_i)$ $(i = 1, 2)$ with $a_2 = \Psi_{\mathtt{supp}}(a_1)$, the following diagram commutes:

$$\begin{tikzcd}
\mathtt{L}^{P_1}(a_1) \arrow[r, "\phi^1_{a_1}"] \arrow[d, "\Psi^{\mathtt{L}}_{a_1}"'] & \mathtt{R}^{P_1}(a_1) \arrow[d, "\Psi^{\mathtt{R}}_{a_2}"] \\
\mathtt{L}^{P_2}(a_2) \arrow[r, "\phi^2_{a_2}"'] & \mathtt{R}^{P_2}(a_2)
\end{tikzcd}$$

Now, since we remember that a trivial derivation is endowed with identity interfaces, Theorem 2 is a consequence of the following one, that we are going to prove:

Theorem 6. *Every operable derivation is isomorphic to a trivial derivation.*

With Lemma 5, this theorem means that any $\mathscr{R}$-derivation Π and any sequence of reduction choices w.r.t. Π can be encoded by an $\mathtt{S}$-derivation P, as expected from the Introduction.

A. Getting a Trivial Derivation considering Tracks Threads

The remainder of this paper is dedicated to the proof of Theorem 6 above holds.

We recall that tracks are numbers that label edges (of types or of derivations). Let P be an operable derivation. We want to find a trivial derivation P_0 that is isomorphic to P (as an operable derivation). But roughly speaking, defining an isomorphism of operable derivation whose domain is P is a matter of giving new values to the tracks which occur in P *i.e.* relabelling the edges of the derivation P. These new values must be chosen appropriately to:

(1) respect the typing rules of $\mathtt{S_h}$
(2) respect the interface of P and yield a trivial derivation.

In System $\mathtt{S_h}$, tracks 0 and 1 are special (they are dedicated to the premise of the $\mathtt{abs}$-rule or the left-premise of the $\mathtt{app_h}$-rule and also to the target of $\rightarrow$) and their value is fixed by 01-isomorphisms: as a matter of fact, an isomorphism of derivation should not relabel the left-hand side of an application (labelled with 1) into an argument branch (labelled with $k \geqslant 2$) or the premise of an abstraction (labelled with 0). But the value of tracks $\geqslant 2$ may be changed (in that, tracks $\geqslant 2$ are an extra-specification layer on multiset intersection). That is why we say that they are *mutable*. We write $\mathtt{E}(P)$ for the set of edges nested in P whose tracks are mutable. There are 3 kinds of $\mathtt{e} \in \mathtt{E}(P)$:

- The edges of the source of arrows nested in types in $\mathtt{T}^P(a)$ (*inner mutable edges*).
- The edges leading to an argument derivation in some $\mathtt{app_h}$-rule (*argument edge*).
- The *axiom edges*[1] which are labelled with axiom tracks in contexts.

We abusively designate a mutable edge by its deepest vertex *e.g.*, the inner edge joining the bipositions $(0{\cdot}1, \varepsilon)$ and $(0{\cdot}1, 8)$ (which which is inside $\mathtt{T}^P(0{\cdot}1)$) is denoted $(0{\cdot}1, 8)$, the argument edge joining the positions 0 and $0{\cdot}3$ is denoted $0{\cdot}3$.

[1]the vocable "axiom *edge*" is improper: the axiom track in an $\mathtt{ax}$-rules labels the root of a singleton sequence type, not an actual edge in a type/tree.

We now discuss the moves of a type (and its edges) inside a hybrid derivation by looking at Fig. 7, before explaining in what this concerns the trivialization of hybrid derivations *via* the notions of brotherhood and consumption below.

Some occurrences of 8 are colored in blue or in red: they all correspond to the label of an edge that "moves" inside the derivation. For instance, each red (resp. blue) occurrence of 8 can be identified to the one just below *via* the typing rules of system $\mathtt{S}_{\mathtt{h}}$: we say that the former is the **ascendant** of the latter. The same can be said about the colored occurrences of 9, 2 and 7. We call a series of ascendants an **ascendant (edge) thread** *e.g.*, the set of the red (resp. blue) occurrences of 8 correspond to an ascendant thread (resp. to another).

Moreover, since the abs-rule "calls" all the assigned types of the bound variable, the top *blue* occurrence of 8 is called by the constructor λx and correspond to the top *red* occurrence of 8. We say that the former occurrence of 8 is the **polar inverse** of the latter.

Ascendance and polar inversion induce a congruence between mutable edges, that we also denote $\equiv$ and that identifies labelled edges w.r.t. the moves of the types inside P. Intuitively, two labelled edges $\mathtt{e}_1$ and $\mathtt{e}_2$ are congruent iff the typing rules of $\mathtt{S}_{\mathtt{h}}$ constrain them to be labelled with the same track. We denote by $\mathtt{Thr}_{\mathtt{E}}(P)$ the quotient set of $\mathtt{E}(P)$ by $\asymp$ and an element $\theta \in \mathtt{Thr}_{\mathtt{E}}(P)$ is called a **mutable edge thread** or simply a **thread**. A thread is composed of at most two ascendant threads. An occurrence of a thread is said to be **negative** if its top ascendant is in an abs-rule and **positive** in every other case (top ascendant in an ax-rule or argument edge). In Fig. 7, the set of colored occurrences of 8 (resp. of 9) correspond to a thread: the blue ones are negative (they ascend to an abstraction) and the red ones are positive (they ascend to an ax-rule). The colored occurrences of 2, 3, 5 and 7 are all positive. We denote these threads θ_i $(i = 2, 3, 5, 7, 8, 9)$ without ambiguity. If θ is a thread, then $\theta^{\oplus}$ (resp. $\theta^{\ominus}$ is its set of positive (resp. negative) occurrences *e.g.*, $\theta_8^{\ominus} = \{(1^2, 5{\cdot}1{\cdot}8), (1^3, 1{\cdot}5{\cdot}1{\cdot}8), (1^3{\cdot}0, 5{\cdot}1{\cdot}8)\}$. Formally, the definition of ascendance and polar inversion requires the following notation: if P is $\mathtt{S}_{\mathtt{h}}$-derivation typing $t(a) = \lambda x$ and $k \geqslant 2$ is an axiom track assigned to x above a, then $\mathtt{pos}_a^{\lambda}(k)$ denotes the unique position $\alpha \geqslant a{\cdot}0$ such that $\mathtt{tr}^P(\alpha) = k$. Remark that P is implicit in $\mathtt{pos}_a^{\lambda}(k)$. For instance, for $P_{\mathtt{ex}}$ (Fig. 3), $\mathtt{pos}_{\varepsilon}^{\lambda}(9) = 0{\cdot}2$, $\mathtt{pos}_{\varepsilon}^{\lambda}(2) = 0{\cdot}3$. We then set:

- If $t(a) = @$ then $(a, c) \rightarrow_{\mathtt{asc}} (a{\cdot}1, 1{\cdot}c)$.
- If $t(a) = \lambda x$, then $(a, 1{\cdot}c) \rightarrow (a{\cdot}0, c)$ and $(a, k \cdot c) \rightarrow_{\mathtt{pi}}$ $(\mathtt{pos}_a^{\lambda}(k), c)$.

The definition of ascendance and polar inversion are illustrated by Fig. 6. They are almost the same as those in [16] (except that an emptiness constant is not needed), whereas the relation of consumption defined below becomes more involved because of interfaces. In contrast, threads *cannot* be defined in system $\mathscr{R}$: in $\Pi_{\mathtt{ex}}$ which concludes with $\lambda x.x\, x : [\ldots, o, o] \rightarrow o'$, there is no way to ascend from one these two occurrences of o to a unique axiom rule concluding with $x : o$.

We are interested in finding an isomorphism fulfilling Theorem 6 and points (1) and (2) at the beginning of this section. The discussion above explains how to capture point (1) (respecting the typing rules): two edges of a same thread should be relabelled with the same value. In other words:

Observation 1. *Defining an isomorphism from P is about specifying a new value $\mathtt{Val}(\theta)$ for the track of each mutable edge thread θ of P.*

B. Brotherhood and Consumption

We now explain how to capture point (2) of Sec. IV-A *i.e.* we also want to "respect" the interface $(\phi_a)_{a \in \mathtt{supp}_{@}(P)}$ of P (in a sense to be defined) while obtaining a trivial interface (*i.e.* using only identity isomorphisms).

Still in Fig. 7, every colored occurrence of 8 occurs beside a colored occurrence of 9 with the same polarity (each pair of 8 and 9 is nested in the same sequence type): we say that θ_8 and θ_9 are **brother threads**. Likewise, the orange threads θ_3 and θ_5 are brothers, as well as the purple threads θ_2 and θ_7.

We endow now the derivation of Fig. 7 with an interface: in the $\mathtt{app}_{\mathtt{h}}$-rule typing $((\lambda yx.xy)z)(a\, x)$, there are two possible interfaces ϕ_1 from $(8{\cdot}o) \rightarrow (8{\cdot}o, 9{\cdot}o) \rightarrow o'$ to $(3{\cdot}o) \rightarrow (2{\cdot}o, 7{\cdot}o) \rightarrow o'$: one "maps" 8 onto 2 and 9 onto 7 and the other 8 onto 7 and 9 onto 2. We consider the second case (8 into 7,9 into 2). Likewise, at the root $\mathtt{app}_{\mathtt{h}}$-rule, there are two interfaces ϕ_{ε} from $(8{\cdot}o, 9{\cdot}o)$ to $(3{\cdot}o, 5{\cdot}o)$. We assume that ϕ_{ε} maps 8 to 3 and 9 to 5.

We associate to the interface the relation of **consumption**: since, intuitively, ϕ_1 associates to the *blue* track 8 (on the left-hand side) the track 2 (on the right hand-side), we say that the thread θ_8 (resp. θ_2) is **left-consumed** (resp. **right-consumed**) at position 1. Moreover, at position 1, the consumed occurrence of 8 is *negative* (blue), so we say that θ_8 is left-consumed negatively, and we write $\theta_8{}^{\ominus}\tilde{\rightarrow}_1{}^{\oplus}\theta_2$ since the consumed occurrence of θ_2 is positive. The notation $\tilde{\rightarrow}_1$ should mention the interface ϕ, but we omit it. Likewise, $\theta_9{}^{\ominus}\tilde{\rightarrow}_1{}^{\oplus}\theta_7$, $\theta_8{}^{\oplus}\tilde{\rightarrow}_{\varepsilon}{}^{\oplus}\theta_3$, $\theta_9{}^{\oplus}\tilde{\rightarrow}_{\varepsilon}{}^{\oplus}\theta_5$. We then say *e.g.*, that θ_9 and θ_7 **face each other** at pos. 1. Formally, $\theta_{\mathtt{L}}\tilde{\rightarrow}_a\theta_{\mathtt{R}}$ if there is $(a \cdot 1, k \cdot c) \in \theta_{\mathtt{L}}$ and $(a \cdot k', c') \in \theta_{\mathtt{R}}$ such[2] that $\phi_a(k \cdot c) = k' \cdot c'$. In an $\mathtt{S}$-derivation, this latter equality is equivalent to $k = k'$ and $c = c'$ (see Sec. 3.4 of [16]).

To obtain Theorem 6, we want a trivial interface up to relabelling the threads. A derivation is trivial when its interfaces are only identities. In the light of Obs. 1, we remark that, to obtain a trivial derivation, we must assign the same *new* track value to *any pair of threads facing each other*. Thus, setting $\tilde{\rightarrow} := \cup_a \tilde{\rightarrow}_a$:

Observation 2. *To obtain a trivial derivation from P, all edge threads such that $\theta_{\mathtt{L}}\tilde{\rightarrow}\theta_{\mathtt{R}}$ should be relabelled with the same values i.e. one must have $\mathtt{Val}(\theta_{\mathtt{L}}) = \mathtt{Val}(\theta_{\mathtt{R}})$ when $\theta_{\mathtt{L}}\tilde{\rightarrow}\theta_{\mathtt{R}}$.*

As a consequence, if two threads θ and θ' are congruent modulo the equivalence closure of $\tilde{\rightarrow}$, they should also be

[2]Note that, when $k, k' \geqslant 2$, $k{\cdot}c \in \mathtt{supp}(\mathtt{L}^P(a))$ and $k'{\cdot}c' \in \mathtt{supp}(\mathtt{R}^P(a))$ *e.g.*, in Fig. 6, $k{\cdot}c \in \mathtt{supp}((S_k)_K)$ and $k'{\cdot}c' \in \mathtt{supp}((S'_k)_{K'})$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{5\vdash x:(8\cdot o)\to(8\cdot o,9\cdot o)\to o'}\,\texttt{ax}\quad\dfrac{}{7\vdash y:o\,[2]}\,\texttt{ax}}{\ldots\vdash x\,y:(8\cdot o,9\cdot o)\to o'}\,\texttt{app}_\texttt{h}}{\ldots\vdash\lambda x.x\,y:(5\cdot(8\cdot o)\to(8\cdot o,9\cdot o)\to o')\to(8\cdot o,9\cdot o)\to o'}\,\texttt{abs}}{\vdash\lambda yx.x\,y:(7\cdot o)\to(5\cdot(8\cdot o)\to(8\cdot o,9\cdot o)\to o')\to(8\cdot o,9\cdot o)\to o'}\,\texttt{abs}\quad\dfrac{}{4\vdash z:o\,[3]}\,\texttt{ax}}{\ldots\vdash(\lambda yx.x\,y)z:(5\cdot(8\cdot o)\to(8\cdot o,9\cdot o)\to o')\to(8\cdot o,9\cdot o)\to o'}\,\texttt{app}_\texttt{h}\quad\dfrac{\dfrac{}{2\vdash a:(\,)\to(3\cdot o)\to(2\cdot o,7\cdot o)\to o'}\,\texttt{ax}}{\ldots\vdash a\,x:(3\cdot o)\to(2\cdot o,7\cdot o)\to o'\,[6]}\,\texttt{app}_\texttt{h}}{\ldots\vdash((\lambda yx.xy)z)(a\,x):(8\cdot o,9\cdot o)\to o'}\,\texttt{app}_\texttt{h}\quad\dfrac{}{4\vdash b:o\,[3]}\,\texttt{ax}\quad\dfrac{}{9\vdash b:o\,[5]}\,\texttt{ax}}{\ldots\vdash(((\lambda yx.xy)z)(a\,x))b:o'}\,\texttt{app}_\texttt{h}$$

Fig. 7. Two Brother Threads

assigned the same track value *e.g.*, θ_5 and θ_7 must receive the same new label, even if they have no direct relation, because $\theta_9\tilde{\rightarrow}\theta_7$ and $\theta_9\tilde{\rightarrow}\theta_5$.

To prove Theorem 6, we must then prove that we can satisfy Obs. 2: the equalities $\texttt{Val}(\theta_\texttt{L}) = \texttt{Val}(\theta_\texttt{R})$ (for $\theta_\texttt{L}\tilde{\rightarrow}\theta_\texttt{R}$) ensure that in the relabelled derivation $P_\texttt{Val}$, every interface is trivial *i.e.* $P_\texttt{Val}$ is a $\texttt{S}$-derivation. However, the assignment $\texttt{Val}$ must be **consistent** *i.e.* two brother threads should not be reassigned the same value *e.g.*, in Fig. 7, θ_8 and θ_9 (or θ_2 and θ_7) cannot receive the same new track value (*i.e.* $\texttt{Val}(\theta_8) \neq \texttt{Val}(\theta_9)$ must hold).

It is actually not difficult to find a good relabelling $\texttt{Val}$ transforming the derivation of Fig. 7 into a trivial one: we replace the singleton sequence judgment typing typing y (resp. z) with the one concluding with $7\vdash y:o\,[8]$ (the track of the argument edge has been changed from 2 to 8), the one typing z z with the one concluding with $4\vdash z:o\,[7]$ and the right part of the derivation with:

$$\dfrac{\dfrac{\ldots\quad\dfrac{\dfrac{}{2\vdash a:(\,)\to(8\cdot o)\to(8\cdot o,9\cdot o)\to o}\,\texttt{ax}}{\ldots\vdash a\,x:(8\cdot o)\to(8\cdot o,9\cdot o)\to o\,[5]}\,\texttt{app}_\texttt{h}}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots}\quad\dfrac{}{4\vdash b:o\,[8]}\,\texttt{ax}\quad\dfrac{}{9\vdash b:o\,[9]}\,\texttt{ax}}{\ldots}$$

The tracks 3, 2 and 7 labelling inner edges have been changed as well as the tracks 3, 6 and 5 labelling argument edges. No axiom track required to be changed in this example.

If one tries to directly prove that every operable P has a consistent assignment $\texttt{Val}$ such that $\texttt{Val}(\theta_\texttt{L}) = \texttt{Val}(\theta_\texttt{R})$ when $\theta_\texttt{L}\tilde{\rightarrow}\theta_\texttt{R}$, one becomes quickly stuck, for the same reasons as in Sec. II-E. We must then try to find another way. That is why we adapt the method of [16]. In general, we prove (Proposition 7 below) that a consistent assignment $\texttt{Val}$ satisfying Obs.2 exists iff there is no *proof* showing that there are two brother threads that should be given an equal track value. Such a proof would be called a **brother chain** and would have the form $\theta_0\tilde{\leftrightarrow}_{a_0}\theta_1\tilde{\leftrightarrow}_{a_1}\ldots\tilde{\leftrightarrow}_{a_{n-1}}\theta_n$, where θ_0 and θ_n are two brother threads and $\tilde{\leftrightarrow}_a$ is the symmetric closure of $\tilde{\rightarrow}_a$. This would imply that $\texttt{Val}(\theta_0)$ and $\texttt{Val}(\theta_n)$ *must* be equal, which is illicit (θ_0 and θ_n are brothers). In other words, a brother chain corresponds to a proof of contradiction in the first order theory $\mathcal{T}_P$ whose set of constants is $\texttt{Thr}_\texttt{E}(P)$ and whose axioms are $\texttt{Val}(\theta_1) = \texttt{Val}(\theta_2)$ for all θ_1,θ_2 s.t. $\theta_1\tilde{\rightarrow}_a\theta_2$ for some $a\in\texttt{supp}_@(P)$ and $\texttt{Val}(\theta_1)\neq\texttt{Val}(\theta_2)$ for all brother threads θ_1,θ_2. The consistence of $\mathcal{T}_P$ is a *necessary* condition to ensure the existence of a relabelling $\texttt{Val}$ with the expected properties. But is it sufficient? The answer is yes and thus, Proposition 7 below can be interpreted as a **completeness property** for theory $\mathcal{T}_P$ (consistence implies existence). It is pivotal to obtain Theorem 6. The approach closely follows that of [16], although here, threads of mutable *edges* are considered instead on the whole set of *bipositions* (see, *e.g.*, Definition 1,2 and Corollary 1 in [16]).

Proposition 7. *If $\mathcal{T}_P$ is consistent, i.e. there are no brother chains w.r.t. an operable derivation P, then there is a trivial derivation P_* isomorphic to P.*

Proof sketch. Let $\sim_\phi$ be the equivalence closure of $\cup_{a\in\texttt{supp}_@(P)}\tilde{\leftrightarrow}_a$ and q be the canonical map from $\texttt{E}(P)$ to $\texttt{E}(P)_\phi = \texttt{E}(P)/\sim_\phi$. Since $\texttt{E}(P)$ is countable, let ι be an injection from $\texttt{E}(P)_\phi$ to $\mathbb{N}\setminus\{0,1\}$. One checks that the relabelling $\iota\circ q$ induces an isomorphism of operable derivations from P to a certain derivation P_* which is indeed correctly defined and trivial (full details in Sec. B.2. of [15]). $\square$

To develop the final stages of this theorem, we also need the following lemmas:

Lemma 8 (Uniqueness of Consumption)**.** *Let P be an operable derivation, $\circledast\in\{\oplus,\ominus\}$ and $\theta\in\texttt{Thr}_\texttt{E}(P)$. Then, there is a most one θ' such that ($\theta^\circledast\tilde{\rightarrow}\theta'$ or $\theta^\circledast\tilde{\leftarrow}\theta'$).*

We define the **applicative depth** of a thread θ as the maximal applicative depth of a judgment in which it occurs *e.g.*, in Fig. 7, $\texttt{ad}(\theta_8) = \texttt{ad}(\theta_9) = 0$ and $\texttt{ad}(\theta_7) = 1$. Consumption causes an increase of applicative depth, provided the left thread is consumed positively:

Lemma 9 (Monotonicity)**.** *If $\theta_\texttt{L}\overset{\oplus}{\tilde{\rightarrow}}\theta_\texttt{R}$, then $\texttt{ad}(\theta_\texttt{L}) < \texttt{ad}(\theta_\texttt{R})$.*

One may easily understand why Lemma 9 is not valid with $\theta_\texttt{L}^\ominus$ instead of $\theta_\texttt{L}^\oplus$: a λx at pos. a may "call" an occurrence of x at a' whose applicative depth is greater than that of a. Thus, the negative occurrence of a thread θ does not have *a priori* maximal applicative depth in θ (this is not the case in Fig. 7).

C. Collapsing Brother Threads

We now prove the main theorem: according to Prop. 7, in order to prove Theorem 6, we must prove that brother chains *do not* exist. For that, we assume *ad absurdum* that there is a brother chain $\theta_0 \overset{\leftharpoonup}{\leftrightarrow}_{a_0} \theta_1 \overset{\leftharpoonup}{\leftrightarrow}_{a_1} \ldots \overset{\leftharpoonup}{\leftrightarrow}_{a_{n-1}} \theta_n$ associated to an operable derivation P, whose interface ϕ is omitted from the notations. Details can be found in Sec. 13.5 of [15].

Argument 1 (normal brother chains do not exist): Using Lemma 8, it is easy to prove that, if no thread is left-consumed negatively in the chain (in that case, we say that the chain is a **normal brother chain**), then it is of the form $\theta_0{}^{\oplus} \overset{\leftharpoonup}{\rightarrow}_{a_0} \theta_1{}^{\oplus} \overset{\leftharpoonup}{\rightarrow}_{a_1} \ldots \overset{\leftharpoonup}{\rightarrow}_{a_{n-1}} \theta_n$ By Lemma 9, this entails that $\mathtt{ad}(a_0) < \mathtt{ad}(a_1) < \mathtt{ad}(a_2) \ldots < \mathtt{ad}(a_n)$ Thus, θ_0 and θ_n cannot be brother, because two brother threads have the same applicative depth. Contradiction.

Argument 2 (if they existed, brother chains could be normalized): Argument 1 suggests that we reduce to the case of *normal* brother chains, whose non-existence has been established thanks to Lemma 9. Let us then endeavour to "normalize" chains when some negative left-consumption. This happens to be possible, but this requires first that we first define a notion of residuation, for mutable edges and then for edge threads. Residuation for edges is mostly similar to that for bipositions (sketched in Sec. III-B). To define residuation for threads, we need to prove that edge residuation is compatible with $\asymp$ (*i.e.* with ascendance and polar inversion), meaning that $\mathtt{e}_1 \asymp \mathtt{e}_2$ implies $\mathtt{Res}_b(\mathtt{e}_1) \asymp \mathtt{Res}_b(\mathtt{e}_2)$. We must also check that $\overset{\leftharpoonup}{\rightarrow}$ is stable by residuation. See Sec. 13.5 and B.4 of [15] for the details.

How do we then escape the case $\theta_{\mathtt{L}}{}^{\ominus} \overset{\leftharpoonup}{\rightarrow}_a \theta_{\mathtt{R}}$? Indeed, when $\theta_{\mathtt{L}}{}^{\ominus} \overset{\leftharpoonup}{\rightarrow}_a \theta_{\mathtt{R}}$, either $t|_a$ is a redex and in that case, $\mathtt{Res}_b(\theta_{\mathtt{L}}) = \mathtt{Res}_b(\theta_{\mathtt{R}})$ (with $b = \overline{a}$) or $t|_a$ is not a redex and in that case, the negative occurrences of $\theta_{\mathtt{L}}$ ascend to a λx in $2\,k$ steps, visiting k @-nodes and k λy-nodes. An example of this case is represented in Fig. 8 with $k = 3$. The sequence type in which $\theta_{\mathtt{L}}$ is nested is colored in blue and by lack of space, we write $\lambda 1, \lambda 2, \lambda 3$ instead of $\lambda x_1, \lambda x_2, \lambda x_3$ and $(*)$ for matterless sequence types. The sequence $(S_k)_{k \in K}$ and, in particular, $\theta_{\mathtt{L}}{}^{\ominus}$ are "called" by λx. After 3 reduction steps, the residual of $\theta_{\mathtt{L}}$ is nested in $(S'_k)_{k \in K'}$ in the left-hand side of redex, which corresponds to the first case. After a fourth reduction step, $\theta_{\mathtt{L}}$ has the same residual as $\theta_{\mathtt{R}}$.

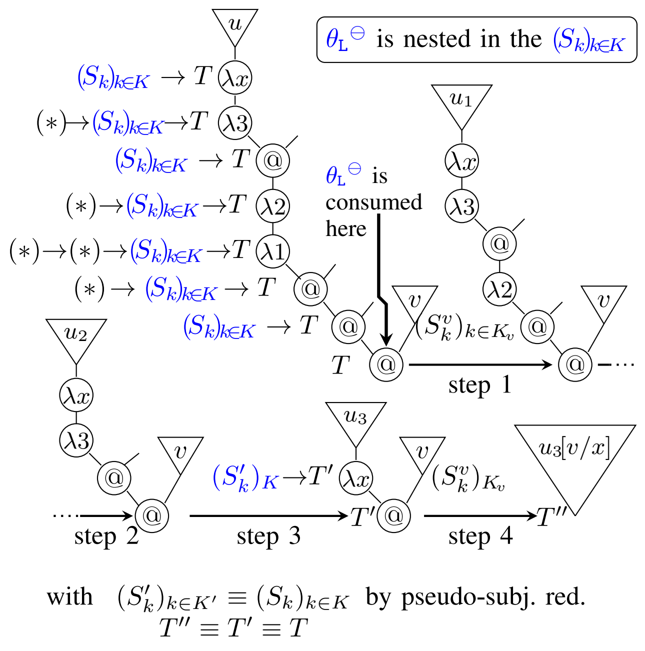

$\theta_{\mathtt{L}}{}^{\ominus}$ is nested in the $(S_k)_{k\in K}$
$(S_k)_{k\in K} \to T$
$(*)\to(S_k)_{k\in K}\to T$
$(S_k)_{k\in K} \to T$
$(*)\to(S_k)_{k\in K}\to T$
$(*)\to(*)\to(S_k)_{k\in K}\to T$
$(*)\to(S_k)_{k\in K}\to T$
$(S_k)_{k\in K} \to T$
$\theta_{\mathtt{L}}{}^{\ominus}$ is consumed here
$(S^v_k)_{k\in K_v}$
step 1
step 2
step 3
step 4
$(S'_k)_K \to T'$
$(S^v_k)_{K_v}$
$u_3[v/x]$
with $(S'_k)_{k\in K'} \equiv (S_k)_{k\in K}$ by pseudo-subj. red.
$T'' \equiv T' \equiv T$

Fig. 8. Collapsing a Redex Tower

In general, the method that we have just described would allow us, starting from a brother chain, to obtain a normal brother chain in a *finite* number of steps. Since normal brother chains do not exist by Argument 1, this proves that brother chains do not exist at all. Thus, by Proposition 7, there is a trivial derivation P_* which is isomorphic to P. This concludes the proof of Theorem 6, which entails Theorem 2.

V. Conclusion and Perspectives

We introduced a sequential type system $\mathtt{S}_{\mathtt{op}}$, which allows encoding reduction paths as in other rigid type systems. The main results of this article (Theorems 2, 6 along with Lemma 5) state that every (possibly infinitary) multiset-based derivation is the collapse of a trivial derivation (which means sequential and permutation-free) and that, moreover, one can directly represent in the trivial setting every reduction path in the multiset or operable setting, the latter meaning rigid with isomorphisms allowing us to encode reduction choices. This proves that the *trivial* rigid framework is as expressive as the non-trivial one and advocates in favor of replacing multiset intersection with sequential intersection, since it is no more complicated to use sequences (system $\mathtt{S}$ does not use a permutation rule or type isomorphisms) than multisets, while having extra-features (parsing, pointing).

Fiore *et al.* [7] introduced the generalized species of structures to interpret the λ-calculus, which have been later enriched by Tsukada *et al.* [13] intro *weighted* generalized species. The fact that sequential intersection is easier to handle than list intersection (subject reduction without permutation) as in [12] suggests a notion of *sequential* (generalized) species of structures to interpret the λ-calculus, which would replace the list comonad $\mathbb{P}$ with the sequence functor $\mathbb{S}$ on small categories $\mathcal{C}$ defined by:

- The objects of $\mathbb{S}(\mathcal{C})$ are sequences $(c_k)_{k \in K}$ with *e.g.*, $K \subseteq \mathbb{N} \setminus \{0,1\}$ and c_k object of $\mathcal{C}$ for all $k \in K$.
- The morphisms of $\mathbb{S}(\mathcal{C})$ are $(\rho, (\phi)_{k \in K} : (c_k)_K \to (c'_k)_{K'})$ with $\rho : K \to K'$ set-bijection and, for all $k \in K$, ϕ_k $\mathcal{C}$-morphism from c_k to $c'_{\rho(k)}$.

This adaptation should demand rethinking some constructions because for instance, $\mathbb{S}$ is not a comonad. We leave this for the future work.

References

[1] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In CSL 2007, Lausanne, pages 298–312.

[2] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. MSCS, 2017.

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

[4] Daniel de Carvalho. Sémantique de la logique linéaire et temps de calcul. PhD thesis, Université Aix-Marseille, November 2007.

[5] Thomas Ehrhard and Laurent Regnier. Uniformity and the taylor expansion of ordinary lambda-terms. Theor. Comput. Sci., 403(2-3):347–372, 2008.

[6] Jörg Endrullis, Helle Hvid Hansen, Dimitri Hendriks, Andrew Polonsky, and Alexandra Silva. A coinductive framework for infinitary rewriting and equational reasoning. In RTA 2015, 2015, Warsaw.

[7] Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel. The cartesian closed bicategory of generalised species of structures. Journal of The London Mathematical Society-second Series - J LONDON MATH SOC-SECOND SER, 77:203–220, 11 2007.

[8] Philippa Gardner. Discovering needed reductions using type theory. In TACS 94, Sendai.

[9] Jean-Yves Girard. Linear logic. Theoretical Computer Science, 50:1–102, 1987.

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

[11] C.-H. Luke Ong and Takeshi Tsukada. Two-level game semantics, intersection types, and recursion schemes. In ICALP 2012, Warwick.

[12] Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Generalised species of rigid resource terms. In LICS, Reykjavik, 2017.

[13] Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Species, profunctors and taylor expansion weighted by SMCC: A unified framework for modelling nondeterministic, probabilistic and quantum programs. In LICS, Oxford, 2018.

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

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

[16] Pierre Vial. Every λ-term is meaningful in the infinitary relational model. In LICS, Oxford, July 9-12, 2018.