**Theoretical Informatics and Applications** Will be set by the publisher
Informatique Théorique et Applications

# A NON-UNIFORM FINITARY RELATIONAL SEMANTICS OF SYSTEM $T$ *

Lionel Vaux[1]

**Abstract**. We study iteration and recursion operators in the denotational semantics of typed $\lambda$-calculi derived from the multiset relational model of linear logic. Although these operators are defined as fixpoints of typed functionals, we prove them finitary in the sense of Ehrhard's finiteness spaces.

**1991 Mathematics Subject Classification.** 03B70, 03D65, 68Q55.

## 1. Introduction

Since its inception in the late 1960's, denotational semantics has proved to be a valuable tool in the study of programming languages, by recasting the equalities between terms induced by the operational semantics into an more abstract algebraic setting. By the Curry–Howard correspondence between proofs and programs, this concept also provided important contributions to the design of logical systems. Notably, the invention of linear logic by Girard [1] followed from his introduction of coherences spaces as a refinement of Scott's continuous semantics of the $\lambda$-calculus [2], moreover taking into account the property of stability put forward by Berry [3].

The design of coherence spaces was moreover largely based on the ideas previously developed by Girard about a quantitative semantics of the $\lambda$-calculus [4]: the property that the behaviour of a program is specified by its action on finite approximants of its argument can be related with the fact that analytic functions are given by power series. This suggests interpreting types by particular topological vector spaces, and terms by analytic functions between them: Girard proposed

*Keywords and phrases:* Higher order primitive recursion — Denotational semantics

* *This work has been partially funded by the French ANR projet blanc "Curry Howard pour la Concurrence" CHOCO ANR-07-BLAN-0324.*

[1] Institut de Mathématiques de Luminy, CNRS UMR 6206, Marseille, France.

such an interpretation using the notion of normal functor, a categorical analogue of analytic functions.

Applying these ideas in a qualitative setting (*i.e.* forgetting about coefficients in power series) led Girard to the definition of qualitative domains [5], later refined into coherence spaces. These models, together with the so-called graph models introduced by Engeler, Plotkin and Scott [6], share the following basic idea: one may interpret types as sets, which we will call *webs*, possibly with some additional structure (order, coherence, *etc.*); and then one may interpret terms as subsets of the webs of their types, respecting the additional structure (ideals, cliques, *etc.*).

The properties of the interpretations of programs (continuity, stability, *etc.*) are put to use in that they allow the particular morphisms associated with $\lambda$-abstractions to be unambiguously represented by their traces, *i.e.* sets of (input bits/output bit)-pairs. Notice the plural in "input bits": intuitively, a program may need several bits of input information to produce one bit of output. Scott's continuity roughly means that each bit of output requires only finitely many bits of input. Berry's stability indicates that each output bit produced by a deterministic program is characterized by a minimal set of input bits. Girard's linearity specifies those programs which use exactly one input bit per output bit. Such webbed models can be refined as models of linear logic, so that every morphism from $A$ to $B$ becomes a linear morphism from $!A$ to $B$, where $!A$ stands for the type of "chunks" of information (precisely those that are sufficient to obtain one output bit).

The most simple webbed model is the relational one: types are interpreted by sets without any additional structure, and any subset is (potentially) the interpretation of a term. Linear morphisms from $A$ to $B$ are simply relations, *i.e.* subsets of $A \times B$. The exponential modality ! is represented by the finite multiset construction, so that arbitrary morphisms from $A$ to $B$ are relations between finite multisets over $A$ and elements of $B$. Despite its apparent simplicity, the relational model of the $\lambda$-calculus thus obtained has many interesting properties: for instance it is injective on $\beta\eta$-classes [7], and it is closely connected with intersection types and execution time [8].

Ehrhard introduced finiteness spaces [9] by refining this relational model. A finiteness space is a set equipped with a finiteness structure, *i.e.* a particular set of subsets which are said to be finitary; and the model is such that the relational denotation of a proof in linear logic is always a finitary subset of its conclusion. The distinctive property of finiteness spaces is that the intersection of two finitary subsets of dual types is always finite. In the associated model of the simply typed $\lambda$-calculus, this feature allowed Ehrhard to reformulate Girard's quantitative semantics in a standard algebraic setting, where morphisms interpreting typed $\lambda$-terms are analytic functions between the topological vector spaces generated by vectors with finitary supports. More explicitly:

- the vectors of type $A$ are of the form $(v_\alpha)_{\alpha \in |A|}$ where $|A|$ is the web of the associated finiteness space;
- $!A$ is similar to the symmetric algebra on $A$, in particular $|!A|$ is the set of all finite multisets of elements of $|A|$;

- an analytic function from $A$ to $B$ is given by a power series:

$$(f(v))_\beta = \sum_{\overline{\alpha}\in!^!A|} f_{(\overline{\alpha},\beta)} \cdot v^{\overline{\alpha}}$$

  where $v^{\overline{\alpha}}$ denotes $\prod_{i=1}^{n} v_{\alpha_i}$ when $\overline{\alpha} = [\alpha_1, \ldots, \alpha_n]$;
- considering only vectors with finitary supports ensures the convergence of the previous sum, because it has only finitely many non-zero terms.

This provided the semantic foundations of Ehrhard–Regnier's differential $\lambda$-calculus [10] and motivated the general study of a differential extension of linear logic (e.g., [11–17]).

It is worth noticing that finiteness spaces can accommodate typed $\lambda$-calculi only. In particular, we will see the relational semantics of fixpoint combinators is finitary only on empty types: this strengthens a previous remark by Ehrhard. The whole point of the finiteness construction is actually to reject infinite computations, by ensuring the intermediate sets involved in the relational interpretation of a cut are all finite. This contrasts with the purely relational model where the inclusion order defines a cpo on homsets and fixpoints are available at all types — the relational semantics even admits reflexive objects, hence models of the untyped $\lambda$-calculus [18, 19]. Despite this restrictive design, Ehrhard was able to define a finitary interpretation of tail-recursive iteration (Section 3 of [9]): this indicates that the finiteness semantics can accommodate a form of typed recursion. This interpretation, however, is not completely satisfactory: tail recursive iteration is essentially linear, thus it does not provide a type of natural numbers in the associated model of the $\lambda$-calculus.

The main result of the present paper is that finiteness spaces can actually accommodate the standard notion of primitive recursion in the $\lambda$-calculus, Gödel's system $T$: we prove $\underline{\mathrm{Fin}}$ admits a weak natural number object in the sense of [20, 21], and we more generally exhibit a finitary recursion operator for this interpretation of the type of natural numbers. This achievement is twofold:

- Before considering finiteness, we must define a recursion operator in the cartesian closed category deduced from the relational model of linear logic. As we have already stated, Ehrhard's proposition does not match this requirement, which has nothing to do with the finiteness structure: this is essentially due to the fact that the interpretation of natural numbers is *flat* (in the sense of domains). In fact, a similar effect was already noted by Girard in the design of his coherence semantics of system $T$ [22]: his solution was to propose a *lazy* interpretation of natural numbers, where laziness refers to the possibility of pattern matching on non normal terms. We adapt Girard's solution to the purely relational case.
- The second aspect of our work is to establish that this relational semantics is finitary. This is far from immediate because the recursion operator is defined as the union of its finitary approximants, by a fixpoint construction: since the fixpoint operator itself is not a finitary relation, it is necessary to obtain stronger properties of these approximants to deduce finitariness.

Parigot proved by syntactic means that no term in the iterator variant of system $T$ could define a valid predecessor operator, *i.e.* a formal inverse of the successor on Church natural numbers [23]. This implies in particular that the recursion operator of system $T$ cannot be recovered from the sole iterator. A notable outcome of our relational interpretation of system $T$ is that it provides semantic evidence of this gap in expressive power between iterator and recursor (although we did not manage to turn this evidence into a fully new proof).

The present paper takes place in a series of works, aiming to extend the quantitative semantics of the simply typed $\lambda$-calculus in vectorial finiteness spaces to functional programming with base types. This would broaden the scope of the already well developed proof theory of differential linear logic: quantitative semantics provides more precise information on cut elimination, and is thus a better guide in the design of syntax than the plain relational interpretation.

Earlier achievements in this direction include Tasson's extension of the algebraic $\lambda$-calculus [24] with a type of booleans, together with a semantic characterization of total terms which is proved to be complete on boolean functions [16]. The present work contributes an important step by defining precisely the relational framework in which a quantitative semantics of system $T$ could be developed.

Since such quantitative approaches are known to extend the proofs-as-programs paradigm to parallel [25], non deterministic [26], concurrent [13] or even quantum [27] programming features, our hope is that recasting standard datatypes in this setting will bring interesting new ideas on how to integrate such features within functional programming languages.

### 1.1. Structure of the paper

In section 2, we briefly describe two cartesian closed categories: the category $\underline{\mathrm{Rel}}$ of sets and multirelations, and the category $\underline{\mathrm{Fin}}$ of finiteness spaces and finitary multirelations. In section 3, we give an explicit presentation of the relational semantics of typed $\lambda$-calculi in $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$, which we extend to system $T$ in section 4. In section 5, we establish a uniformity property of iteration-definable morphisms, which does not hold for recursion in general. Finally, we discuss possible further developments of this work.

## 2. Sets, Relations and Finiteness Spaces

### 2.1. Notations

If $A$ is a set, we write $\#A$ for the cardinality of $A$, $\mathfrak{P}(A)$ for the powerset of $A$ and $\mathfrak{P}_{\mathrm{f}}(A)$ for the set of all finite subsets of $A$. We identify multisets of elements of $A$ with functions $A \longrightarrow \mathbf{N}$. If $\mu$ is such a multiset, we write $\mathrm{Supp}(\mu)$ for its support set $\{\alpha \in A;\ \mu(\alpha) \neq 0\}$. A finite multiset is a multiset with finite support. We write $A^!$ for the set of all finite multisets of elements of $A$. Whenever $(\alpha_1, \ldots, \alpha_n) \in A^n$, we write $\overline{\alpha} = [\alpha_1, \ldots, \alpha_n]$ for the corresponding finite multiset: $\alpha \in A \mapsto \#\{i;\ \alpha_i = \alpha\}$. We also write $\#[\alpha_1, \ldots, \alpha_n] = n$ for the cardinality of

multisets. The empty multiset is [] and we use the additive notation for multiset union, *i.e.* $\mu + \mu' : \alpha \in A \mapsto \mu(\alpha) + \mu'(\alpha)$.

### 2.2. Sets and (multi-)relations

Let $f \subseteq A \times B$ be a relation from $A$ to $B$. We write $f^\perp = \{(\beta, \alpha);\ (\alpha, \beta) \in f\}$. For all $a \subseteq A$, we write $f \cdot a$ for the *direct image* of $a$ by $f$:

$$f \cdot a = \{\beta \in B;\ \exists \alpha \in a,\ (\alpha, \beta) \in f\}\,.$$

We call *multirelation* from $A$ to $B$ any relation from $A^!$ to $B$. We define the category $\underline{\mathrm{Rel}}$ of sets and multirelations:

- objects are sets;
- morphisms are multirelations, *i.e.* $\underline{\mathrm{Rel}}(A, B) = \mathfrak{P}\left(A^! \times B\right)$;
- the identity on $A$ is $id_A = \{([\alpha], \alpha);\ \alpha \in A\}$;
- if $f \in \underline{\mathrm{Rel}}(A, B)$ and $g \in \underline{\mathrm{Rel}}(B, C)$ then their composite is $g \circ f = \left\{\left(\sum_{i=1}^n \overline{\alpha}_i, \gamma\right);\ \exists \overline{\beta} = [\beta_1, \ldots, \beta_n] \in B^!,\ (\overline{\beta}, \gamma) \in g \wedge \forall i\ (\overline{\alpha}_i, \beta_i) \in f\right\}$.

This construction can be thought of as follows: $([\alpha_1, \ldots, \alpha_p], \beta) \in f$ represents the instance of $f$ which produces the result $\beta$ by consuming the resources $[\alpha_1, \ldots, \alpha_p]$. Then the definition of composition is quite natural: to produce $\gamma$, $g$ might consume resources $[\beta_1, \ldots, \beta_n]$, each $\beta_i$ being produced by $f$, consuming $\overline{\alpha}_i$; the overall process builds $\gamma$ from $\sum_{i=1}^n \overline{\alpha}_i$.

Let us mention that $\underline{\mathrm{Rel}}$ is the co-Kleisli category of the comonad $(-)^!$ in the relational model of linear logic. Beyond the motivations and general line of work exposed in our introduction, this point is however of marginal interest in the remaining of the paper. The careful reader may nonetheless refer to the addendum of [9] for a formal definition of this model of linear logic.

The category $\underline{\mathrm{Rel}}$ is cartesian closed. The cartesian product is given by the disjoint union of sets $A \uplus B = (\{1\} \times A) \cup (\{2\} \times B)$, with terminal object the empty set $\emptyset$. Projections are $\{([(1, \alpha)], \alpha);\ \alpha \in A\} \in \underline{\mathrm{Rel}}(A \uplus B, A)$ and $\{([(2, \beta)], \beta);\ \beta \in B\} \in \underline{\mathrm{Rel}}(A \uplus B, B)$. If $f \in \underline{\mathrm{Rel}}(C, A)$ and $g \in \underline{\mathrm{Rel}}(C, B)$, pairing is given by: $\langle f, g\rangle = \{(\overline{\gamma}, (1, \alpha));\ (\overline{\gamma}, \alpha) \in f\} \cup \{(\overline{\gamma}, (2, \beta));\ (\overline{\gamma}, \beta) \in g\} \in \underline{\mathrm{Rel}}(C, A \uplus B)$. The unique morphism from $A$ to $\emptyset$ is $\emptyset$. The adjunction for closedness is $\underline{\mathrm{Rel}}(A \uplus B, C) \cong \underline{\mathrm{Rel}}(A, B^! \times C)$ which boils down to the bijection $(A \uplus B)^! \cong A^! \times B^!$: this is a typical feature of cartesian closed categories derived from models of linear logic by the co-Kleisli construction (see, e.g., the comprehensive survey by Melliès [28]).

Moreover, $\underline{\mathrm{Rel}}$ is enriched on cpos: the inclusion of multirelations is a complete partial order.

### 2.3. Finiteness spaces and finitary multirelations

The construction of finiteness spaces follows a well established pattern [29]. It is given by the following notion of orthogonality on subsets $a, a' \subseteq A$: $a \perp a'$ iff $a \cap a'$ is finite. Then one unrolls the familiar definitions associated with

biorthogonal closure. We retain only what is necessary for our exposition: for a detailed presentation, the obvious reference is [9].

Let $\mathfrak{F} \subseteq \mathfrak{P}(A)$ be any set of subsets of $A$. We define the *pre-dual* of $\mathfrak{F}$ in $A$ as $\mathfrak{F}^\perp = \{a' \subseteq A;\ \forall a \in \mathfrak{F},\ a \perp a'\}$. We have the following immediate properties: $\mathfrak{F} \subseteq \mathfrak{F}^{\perp\perp}$ and, if $\mathfrak{G} \subseteq \mathfrak{F}$, $\mathfrak{F}^\perp \subseteq \mathfrak{G}^\perp$. By the last two, we get $\mathfrak{F}^\perp = \mathfrak{F}^{\perp\perp\perp}$. Moreover, $\mathfrak{P}_\mathrm{f}(A) = \mathfrak{P}(A)^\perp \subseteq \mathfrak{F}^\perp$. A *finiteness structure* on $A$ is a set $\mathfrak{F}$ of subsets of $A$ such that $\mathfrak{F}^{\perp\perp} = \mathfrak{F}$. Then a *finiteness space* is a dependent pair $\mathcal{A} = (|\mathcal{A}|, \mathfrak{F}(\mathcal{A}))$ where $|\mathcal{A}|$ is the underlying set, called the *web* of $\mathcal{A}$, and $\mathfrak{F}(\mathcal{A})$ is a finiteness structure on $|\mathcal{A}|$. We write $\mathcal{A}^\perp$ for the *dual* finiteness space: $\left|\mathcal{A}^\perp\right| = |\mathcal{A}|$ and $\mathfrak{F}\left(\mathcal{A}^\perp\right) = \mathfrak{F}(\mathcal{A})^\perp$. The elements of $\mathfrak{F}(\mathcal{A})$ are called the *finitary subsets* of $\mathcal{A}$.

For every set $A$, $(A, \mathfrak{P}_\mathrm{f}(A))$ is a finiteness space and $(A, \mathfrak{P}_\mathrm{f}(A))^\perp = (A, \mathfrak{P}(A))$. In particular, each finite set $A$ is the web of exactly one finiteness space: $(A, \mathfrak{P}_\mathrm{f}(A)) = (A, \mathfrak{P}(A))$. We introduce the empty finiteness space $\mathcal{E} = (\emptyset, \{\emptyset\})$ and the finiteness space of *flat natural numbers* $\mathcal{N} = (\mathbf{N}, \mathfrak{P}_\mathrm{f}(\mathbf{N}))$. If $\mathcal{A}$ and $\mathcal{B}$ are finiteness spaces, we define $\mathcal{A} \,\&\, \mathcal{B}$ and $\mathcal{A} \Rightarrow \mathcal{B}$ as follows:

- $|\mathcal{A} \,\&\, \mathcal{B}| = |\mathcal{A}| \uplus |\mathcal{B}|$ and $\mathfrak{F}(\mathcal{A} \,\&\, \mathcal{B}) = \{a \uplus b;\ a \in \mathfrak{F}(\mathcal{A}) \wedge b \in \mathfrak{F}(\mathcal{B})\}$;
- $|\mathcal{A} \Rightarrow \mathcal{B}| = |\mathcal{A}|^! \times |\mathcal{B}|$ and $f \in \mathfrak{F}(\mathcal{A} \Rightarrow \mathcal{B})$ iff $\forall a \in \mathfrak{F}(\mathcal{A})$, $f \cdot a^! \in \mathfrak{F}(\mathcal{B})$, and $\forall \beta \in |\mathcal{B}|$, $(f^\perp \cdot \{\beta\}) \perp a^!$.

For instance, setting $\mathcal{S} = \{([k], k+1);\ k \in \mathbf{N}\}$, we have $\mathcal{S} \in \mathfrak{F}(\mathcal{N} \Rightarrow \mathcal{N})$. It is easily seen that $\mathcal{A} \,\&\, \mathcal{B}$ is a finiteness space, but the same result for $\mathcal{A} \Rightarrow \mathcal{B}$ is quite technical [9]. We call *finitary multirelations* the elements of $\mathfrak{F}(\mathcal{A} \Rightarrow \mathcal{B})$.

Notice that $\mathfrak{F}(\mathcal{A} \Rightarrow \mathcal{B}) \subseteq \underline{\mathrm{Rel}}(|\mathcal{A}|, |\mathcal{B}|)$. Moreover the identity $id_\mathcal{A} = id_{|\mathcal{A}|}$ is always finitary from $\mathcal{A}$ to itself, and finitary multirelations compose. We write $\underline{\mathrm{Fin}}$ for the category of finiteness spaces with $\underline{\mathrm{Fin}}(\mathcal{A}, \mathcal{B}) = \mathfrak{F}(\mathcal{A} \Rightarrow \mathcal{B})$ and composition defined as in $\underline{\mathrm{Rel}}$. It is cartesian closed with terminal object $\mathcal{E}$, product $- \,\&\, -$ and exponential $- \Rightarrow -$: the definitions of those functors on morphisms, the natural transformations, and the adjunction required for cartesian closedness are exactly the same as for $\underline{\mathrm{Rel}}$, applied to the webs of finiteness spaces and to finitary multirelations. By contrast with $\underline{\mathrm{Rel}}$, $\underline{\mathrm{Fin}}$ is not cpo-enriched: the union of finitary multirelations might not be finitary — see Section 4.2. Again, for reference, the reader may check that $\underline{\mathrm{Fin}}$ is the co-Kleisli category of the exponential comonad in the finitary relational model of linear logic, which is detailed in the Section 1 of [9].

## 3. The Multiset Relational Semantics of Typed λ-Calculi

### 3.1. Typed λ-calculi

In this section, we give an explicit description of the interpretation in $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$ of the basic constructions of typed λ-calculi with products. Type expressions are given by:

$$A, B ::= X \mid A \to B \mid A \times B \mid \top$$

$$\frac{}{\Gamma, x : A, \Delta \vdash x : A}\ (\text{Var}) \qquad \frac{}{\Gamma \vdash \langle\rangle : \top}\ (\text{Unit}) \qquad \frac{a \in \mathfrak{C}_A}{\Gamma \vdash a : A}\ (\text{Const})$$

$$\frac{\Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x\, s : A \to B}\ (\text{Abs}) \qquad \frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash s\, t : B}\ (\text{App})$$

$$\frac{\Gamma \vdash s : A \qquad \Gamma \vdash t : B}{\Gamma \vdash \langle s, t\rangle : A \times B}\ (\text{Pair}) \qquad \frac{\Gamma \vdash s : A_1 \times A_2}{\Gamma \vdash \pi_i s : A_i}\ (\text{Proj}_i)$$

FIGURE 1. Rules of typed $\lambda$-calculi with products

where $X$ ranges over a fixed set $\mathfrak{A}$ of atomic types. Term expressions are given by:

$$s, t ::= x \mid a \mid \lambda x\, s \mid s\, t \mid \langle s, t\rangle \mid \pi_1\, s \mid \pi_2\, s \mid \langle\rangle$$

where $x$ ranges over term variables and $a$ ranges over term constants. To each variable and to each constant, we associate a type: we write $\mathfrak{C}_A$ for the collection of constants of type $A$, and we assume that every type admits a countably infinite set of variables.

A typing judgement is an expression $\Gamma \vdash s : A$ derived from the rules in Figure 1 where contexts $\Gamma$ and $\Delta$ range over lists $(x_1 : A_1, \ldots, x_n : A_n)$ of typed variables. Clearly, if a term $s$ is typable, then its type is uniquely determined, say $A$, and then $\Gamma \vdash s : A$ iff $\Gamma$ contains the free variables of $s$. We denote by $s\,[x := t]$ the capture avoiding substitution of $t$ for $x$ in $s$.

The operational semantics of a typed $\lambda$-calculus is given by a congruence $\simeq$ on typed terms: if $s \simeq t$, then $s$ and $t$ have the same type, say $A$; we then write $\Gamma \vdash s \simeq t : A$ for any suitable $\Gamma$. In general, we will give $\simeq$ as the reflexive, symmetric and transitive closure of a contextual relation $>$ on typed terms. We define $>_0$ as the least one such that: $\pi_i\langle s_1, s_2\rangle >_0 s_i$ and $(\lambda x\, s)\, t >_0 s\,[x := t]$ (with the obvious assumptions ensuring typability), and we write $\simeq_0$ for the corresponding equivalence.

Pure typed $\lambda$-calculi are those with no additional constant or conversion rule: fix a set $\mathfrak{A}$ of atomic types, and write $\Lambda_0^{\mathfrak{A}}$ for the calculus where $\mathfrak{C}_A = \emptyset$ for all $A$, and $s \simeq t$ iff $s \simeq_0 t$.

### 3.2. Relational interpretation and finiteness property

Assume a set $[\![X]\!]$ is given for each base type $X$; then we interpret type constructions by $[\![A \to B]\!] = [\![A]\!]^! \times [\![B]\!]$, $[\![A \times B]\!] = [\![A]\!] \uplus [\![B]\!]$ and $[\![\top]\!] = \emptyset$. Further assume that with every constant $a \in \mathfrak{C}_A$ is associated a subset $[\![a]\!] \subseteq [\![A]\!]$. The relational semantics of a derivable typing judgement

$$x_1 : A_1, \ldots, x_n : A_n \vdash s : A$$

will be a relation

$$[\![s]\!]_{x_1:A_1,\ldots,x_n:A_n} \subseteq [\![A_1]\!]^! \times \cdots \times [\![A_n]\!]^! \times [\![A]\!]\,.$$

$$\frac{}{\Gamma^{[]}, x^{[\alpha]} : A, \Delta^{[]} \vdash x^{\alpha} : A}\ [\![\text{Var}]\!] \qquad \frac{a \in \mathfrak{C}_A \quad \alpha \in [\![a]\!]}{\Gamma^{[]} \vdash a^{\alpha} : A}\ [\![\text{Const}]\!]$$

$$\frac{\Gamma, x^{\overline{\alpha}} : A \vdash s^{\beta} : B}{\Gamma \vdash \lambda x\, s^{(\overline{\alpha},\beta)} : A \to B}\ [\![\text{Abs}]\!]$$

$$\frac{\Gamma_0 \vdash s^{([\alpha_1,\ldots,\alpha_k],\beta)} : A \to B \qquad \Gamma_1 \vdash t^{\alpha_1} : A \quad \cdots \quad \Gamma_k \vdash t^{\alpha_k} : A}{\sum_{j=0}^{k} \Gamma_j \vdash s\, t^{\beta} : B}\ [\![\text{App}]\!]$$

$$\frac{\Gamma \vdash s_i^{\alpha} : A_i}{\Gamma \vdash \langle s_1, s_2\rangle^{(i,\alpha)} : A_1 \times A_2}\ [\![\text{Pair}_i]\!] \qquad \frac{\Gamma \vdash s^{(i,\alpha)} : A_1 \times A_2}{\Gamma \vdash \pi_i s^{\alpha} : A_i}\ [\![\text{Proj}_i]\!]$$

FIGURE 2. Computing the elements of the relational semantics

We first introduce the deductive system of Figure 2, which is a straightforward adaptation of de Carvalho's system $R$ [18] to the simply typed case. In this system, derivable judgements are semantic annotations of typing judgements:

$$x_1^{\overline{\alpha}_1} : A_1, \ldots, x_n^{\overline{\alpha}_n} : A_n \vdash s^{\alpha} : A$$

stands for

$$(\overline{\alpha}_1, \ldots, \overline{\alpha}_n, \alpha) \in [\![s]\!]_{x_1:A_1,\ldots,x_n:A_n}\,.$$

In rules [[Var]] and [[Const]], $\Gamma^{[]}$ denotes an annotated context of the form $x_1^{[]} : A_1, \ldots, x_n^{[]} : A_n$. In rule [[App]], the sum of annotated contexts is defined pointwise:

$$\left(x_1^{\overline{\alpha}_1} : A_1, \ldots, x_n^{\overline{\alpha}_n} : A_n\right) + \left(x_1^{\overline{\alpha}'_1} : A_1, \ldots, x_n^{\overline{\alpha}'_n} : A_n\right) = \left(x_1^{\overline{\alpha}''_1} : A_1, \ldots, x_n^{\overline{\alpha}''_n} : A_n\right)$$

where $\overline{\alpha}''_i = \overline{\alpha}_i + \overline{\alpha}'_i$ for all $i$. The semantics of a term is then simply obtained as the set of its annotations:

$$[\![s]\!]_{x_1:A_1,\ldots,x_n:A_n} = \left\{(\overline{\alpha}_1, \ldots, \overline{\alpha}_n, \alpha);\ x_1^{\overline{\alpha}_1} : A_1, \ldots, x_n^{\overline{\alpha}_n} : A_n \vdash s^{\alpha} : A\right\}.$$

Notice there is no rule for $\langle\rangle$ in Figure 2, hence $[\![\langle\rangle]\!]_{\Gamma} = \emptyset$ for all $\Gamma$.

**Theorem 3.1** (Invariance)**.** *If* $\Gamma \vdash s \simeq_0 t : A$ *then* $[\![s]\!]_{\Gamma} = [\![t]\!]_{\Gamma}$.

*Proof.* We followed the standard interpretation of typed $\lambda$-calculi in cartesian closed categories, in the particular case of Rel. A direct proof is also easy, first proving a substitution lemma: if $\Gamma_0, x : A^{[\alpha_1,\ldots,\alpha_k]}, \Delta_0 \vdash s^{\beta} : B$, and, for all $j \in \{1, \ldots, k\}$, $\Gamma_j, \Delta_j \vdash t^{\alpha_j} : A$, then $\sum_{j=0}^{k} \Gamma_j, \sum_{j=0}^{k} \Delta_j \vdash s\,[x := t]^{\beta} : B$. □

The relational interpretation also defines a semantics in Fin: assume a finiteness structure $\mathfrak{F}(X)$ is given for each atomic type $X$, so that $X^* = ([\![X]\!], \mathfrak{F}(X))$ is a finiteness space, and let $(A \to B)^* = A^* \Rightarrow B^*$, $(A \times B)^* = A^* \,\&\, B^*$ and $\top^* = \mathcal{E}$. Then, further assuming that, for all $a \in \mathfrak{C}_A$, $[\![a]\!] \in \mathfrak{F}(A^*)$, we obtain:

**Theorem 3.2** (Finiteness)**.** *If* $x_1 : A_1, \ldots, x_n : A_n \vdash s : A$ *then* $[\![s]\!]_{x_1:A_1,\ldots,x_n:A_n} \in \mathfrak{F}(A_1^* \Rightarrow \cdots \Rightarrow A_n^* \Rightarrow A^*)$.

*Proof.* This is a straightforward consequence of the fact that the cartesian closed structure of $\underline{\mathrm{Fin}}$ is given by the same morphisms as in $\underline{\mathrm{Rel}}$. A direct proof is also possible, by induction on typing derivations. □

### 3.3. On the relations denoted by $\lambda$-terms

We have just shown that $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$ model $\simeq_0$ in pure typed $\lambda$-calculi. Be aware that if we introduce no atomic type, then the semantics is actually trivial: in $\Lambda_0^{\emptyset}$, all types and terms are interpreted by $\emptyset$.

By contrast, we can consider the internal language $\Lambda_{\underline{\mathrm{Rel}}}$ of $\underline{\mathrm{Rel}}$ in which all relations can be described: fix $\mathfrak{A}$ as the collection of all sets (or a fixed set of sets) and $\mathfrak{C}_A = \mathfrak{P}(|\!|A|\!|)$. Then let $s \simeq_{\underline{\mathrm{Rel}}} t$ iff $|\!|s|\!|_\Gamma = |\!|t|\!|_\Gamma$, for any suitable $\Gamma$. The point in defining such a language is to enable very natural notations for relations: in general, we will identify closed terms in $\Lambda_{\underline{\mathrm{Rel}}}$ with the relations they denote in the empty context. For instance, we write $id_A = \lambda x\, x$ with $x$ of type $A$; and if $f \in \underline{\mathrm{Rel}}(A,B)$ and $g \in \underline{\mathrm{Rel}}(B,C)$, we have $g \circ f = \lambda x\,(g\,(f\,x))$. Similarly, the internal language $\Lambda_{\underline{\mathrm{Fin}}}$ of $\underline{\mathrm{Fin}}$, where $\mathfrak{A}$ is the collection of all finiteness spaces and $\mathfrak{C}_A = \mathfrak{F}(A^*)$, allows us to denote conveniently all finitary relations.

Before we address the main subject of the paper, system $T$, let's just review some easy examples of usual $\lambda$-calculus constructions that can be modelled in $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$. First, being cartesian closed categories, they are actually models of pure typed $\lambda$-calculi with extensionality, surjective pairing and terminal object: Theorem 3.1 still holds if we add the reductions $\lambda x\,(u\,x) >_0 u$, $\langle \pi_1\, s, \pi_2\, s\rangle >_0 s$ and $v >_0 \langle\rangle$ as soon as $x$ is not free in $u$ and $v$ has type $\top$ (although, in that case, $>_0$ is no longer confluent).

We can also extend the language with particular base types and constants. For instance, we can introduce base type $\mathsf{Bool}$ together with constants $\mathsf{T}$ and $\mathsf{F}$ of type $\mathsf{Bool}$, and $\mathsf{D}_A$ of type $\mathsf{Bool} \to A \to A \to A$, with the additional reductions $\mathsf{D}\,\mathsf{T}\,s\,t > s$ and $\mathsf{D}\,\mathsf{F}\,s\,t > t$ (we will in general omit the type subscript of such constants and keep the obvious hypotheses on typability implicit) and fix interpretations as follows: let $\mathcal{B}$ a finiteness space with a two element web $|\mathcal{B}| = \{tt, ff\}$; then let $\mathsf{Bool}^* = \mathcal{B}$, $|\!|\mathsf{T}|\!| = \{tt\}$, $|\!|\mathsf{F}|\!| = \{ff\}$ and $|\!|\mathsf{D}_A|\!| = \mathcal{D}_A = \{([tt], [\alpha], [], \alpha);\ \alpha \in |A^*|\} \cup \{([ff], [], [\alpha], \alpha);\ \alpha \in |A^*|\}$. That these interpretations are finitary should be clear. Then one retains that $\Gamma \vdash s \simeq t : A$ implies $|\!|s|\!|_\Gamma = |\!|t|\!|_\Gamma$. But the essential point is that this semantics is internal: in $\Lambda_{\underline{\mathrm{Rel}}}$ and $\Lambda_{\underline{\mathrm{Fin}}}$, $x : A, y : A \vdash \mathcal{D}_A\,\mathcal{T}\,x\,y \simeq x : A$ and $x : A, y : A \vdash \mathcal{D}_A\,\mathcal{F}\,x\,y \simeq y : A$. This means that booleans interact as expected with all relations of appropriate types and not just with those derived from $\lambda$-terms.

Much more structure can be revealed in $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$, with computational counterparts in $\Lambda_{\underline{\mathrm{Rel}}}$ and $\Lambda_{\underline{\mathrm{Fin}}}$. Most importantly, the semantics of the $\lambda$-calculus in $\underline{\mathrm{Fin}}$ lead Ehrhard and Regnier to the definition of the differential $\lambda$-calculus in [10]. One can present this calculus in the current framework by introducing, for all types $A$ and $B$, a new constant $\mathsf{Diff}_{A,B}$ of type $(A \to B) \to A \to A \to B$ subject to a new reduction rule $(\mathsf{Diff}\,\lambda x\, s)\,t > \lambda x\,\left(\frac{\partial\, s}{\partial\, x} \cdot t\right)$ where $\frac{\partial\, s}{\partial\, x} \cdot t$ is the linearized version of substitution defined in [10].

### 3.4. System $T$

The main contribution of the present paper is to establish that Fin models Gödel's system $T$, which can be presented in various ways. The iterator version of system $T$ is the typed $\lambda$-calculus with an atomic type $\mathsf{Nat}$ of natural numbers, and constants $\mathsf{O}$ of type $\mathsf{Nat}$ (corresponding to zero), $\mathsf{S}$ of type $\mathsf{Nat} \to \mathsf{Nat}$ (corresponding to successor) and for every type $A$, $\mathsf{I}_A$ of type $\mathsf{Nat} \to (A \to A) \to A \to A$, subject to the following additional conversions: $\mathsf{I}\,\mathsf{O}\,u\,v > v$ and $\mathsf{I}\,(\mathsf{S}\,t)\,u\,v > u\,(\mathsf{I}\,t\,u\,v)$. The recursor variant is similar, but the iterator is replaced with $\mathsf{R}_A$ of type $\mathsf{Nat} \to (\mathsf{Nat} \to A \to A) \to A \to A$ subject to conversions $\mathsf{R}\,\mathsf{O}\,u\,v > v$ and $\mathsf{R}\,(\mathsf{S}\,t)\,u\,v > u\,t\,(\mathsf{R}\,t\,u\,v)$. Yet another possible system is obtained with tail recursive iteration: take $\mathsf{J}_A$ of type $\mathsf{Nat} \to (A \to A) \to A \to A$ and let $\mathsf{J}\,\mathsf{O}\,u\,v > v$ and $\mathsf{J}\,(\mathsf{S}\,t)\,u\,v > \mathsf{J}\,t\,u\,(u\,v)$.

These systems allow us to represent exactly the same functions on the set of natural numbers, where the number $n$ is denoted by $\mathsf{S}^n\,\mathsf{O}$: this is the consequence of a normalization theorem (see [22]). Notice in particular that, when applied to a canonical integer, $\mathsf{I}$ and $\mathsf{J}$ coincide: $\mathsf{I}\,(\mathsf{S}^n\,\mathsf{O})uv \simeq \mathsf{J}\,(\mathsf{S}^n\,\mathsf{O})uv \simeq u^n\,v$; this does not hold for all terms of type $\mathsf{Nat}$, however (consider, e.g., a variable of type $\mathsf{Nat}$). Moreover, we can define a recursor using iteration and products with the standard encoding $\mathsf{rec} = \lambda x\,\lambda y\,\lambda z\,\pi_1\,(\mathsf{I}\,x\,(\lambda w\,\langle y\,(\pi_2\,w)\,(\pi_1\,w), \mathsf{S}\,(\pi_2\,w)\rangle)\,\langle z, \mathsf{O}\rangle)$, and we get $\mathsf{rec}\,(\mathsf{S}^n\,\mathsf{O})uv \simeq \mathsf{R}\,(\mathsf{S}^n\,\mathsf{O})uv$: the idea is to reconstruct the integer argument on the fly. But this encoding is valid only for ground terms of type $\mathsf{Nat}$: $\mathsf{rec}\,(\mathsf{S}\,t)\,u\,v \simeq u\,t\,(\mathsf{rec}\,t\,u\,v)$ holds only if we suppose $t$ is of the form $\mathsf{S}^n\,\mathsf{O}$, or reduces to such a term (notice we could as well use $\mathsf{J}$ in the definition of $\mathsf{rec}$). By contrast, the encoding of the iterator by $\mathsf{iter} = \lambda x\,\lambda y\,\lambda z\,(\mathsf{R}\,x\,(\lambda x'\,y)\,z)$ is extensionally valid: $\mathsf{iter}\,\mathsf{O}\,u\,v \simeq v$ and $\mathsf{iter}\,(\mathsf{S}\,t)\;u\,v \simeq u\,(\mathsf{iter}\,t\,u\,v)$ for all $t, u, v$.

The fact that one direction of the encoding holds only on ground terms indicate that the algorithmic properties of both systems may differ. And these differences will appear in the semantics (see Section 5). Also, recall the discussion in our introduction: the tail recursive variant of iterator, $\mathsf{J}$ subject to $\mathsf{J}\,(\mathsf{S}\,t)\,u\,v > \mathsf{J}\,t\,u\,(u\,v)$, uses its integer argument linearly. This enabled Ehrhard to define a semantics of iteration, with $\mathsf{Nat}^* = \mathcal{N} = (\mathbf{N}, \mathfrak{P}_{\mathrm{f}}(\mathbf{N}))$, $[\![\mathsf{O}]\!] = \mathcal{O} = \{0\}$ and $[\![\mathsf{S}]\!] = \mathcal{S} = \{([n], n+1);\ n \in \mathbf{N}\}$. Such an interpretation of natural numbers, however, fails to provide a semantics of $\mathsf{I}$ or $\mathsf{R}$, in Rel or Fin.

**Lemma 3.3.** *Assume $[\![\mathsf{Nat}]\!] = |\mathcal{N}|$, $[\![\mathsf{O}]\!] = \mathcal{O}$ and $[\![\mathsf{S}]\!] = \mathcal{S}$, and let $A$ be any type such that $[\![A]\!] \neq \emptyset$. Then there is no $\mathcal{I}_A \subseteq [\![\mathsf{Nat} \to (A \to A) \to A \to A]\!]$ such that, setting $[\![\mathsf{I}_A]\!] = \mathcal{I}_A$, we obtain $[\![\mathsf{I}\,\mathsf{O}\,u\,v]\!]_\Gamma = [\![v]\!]_\Gamma$ and $[\![\mathsf{I}\,(\mathsf{S}\,t)\;u\,v]\!]_\Gamma = [\![u\,(\mathsf{I}\,t\,u\,v)]\!]_\Gamma$ as soon as $\Gamma \vdash t : \mathsf{Nat}$, $\Gamma \vdash u : A \to A$ and $\Gamma \vdash v : A$.*

*Proof.* By contradiction, assume the above equations hold. We can then derive the following in $\Lambda_{\mathrm{Rel}}$: $\lambda x\,\lambda y\,(\mathcal{I}\,\mathcal{O}\;(\lambda z\,x)\;y) \simeq \lambda x\,\lambda y\,y$ and, observing that $\emptyset \subseteq \mathbf{N}$, $\lambda x\,\lambda y\,(\mathcal{I}\;(\mathcal{S}\,\emptyset)\;(\lambda z\,x)\;y) \simeq \lambda x\,\lambda y\,x$. Since moreover $\mathcal{S}\,\emptyset \simeq \emptyset$, and the semantics is monotonic for relation inclusion, we obtain: $\lambda x\,\lambda y\,(\mathcal{I}\;(\mathcal{S}\,\emptyset)\;(\lambda z\,x)\;y) \simeq \lambda x\,\lambda y\,(\mathcal{I}\,\emptyset\;(\lambda z\,x)\;y) \subseteq \lambda x\,\lambda y\,(\mathcal{I}\,\mathcal{O}\;(\lambda z\,x)\;y)$. We conclude that $[\![\lambda x\,\lambda y\,x]\!]_{x:A,y:A} \subseteq [\![\lambda x\,\lambda y\,y]\!]_{x:A,y:A}$ for every type $A$, which obviously fails as soon as $[\![A]\!] \neq \emptyset$. □

# 4. A finitary relational interpretation of primitive recursion

## 4.1. Lazy natural numbers

We say a multirelation from $A$ to $B$ is:

- *linear* if it contains only elements of the form $([\alpha], \beta)$;
- *affine* if it contains only elements of the form $([\alpha], \beta)$ or $([], \beta)$;
- *strict* if it contains no element of the form $([], \beta)$.

In the proof of Lemma 3.3, $\mathcal{S}\,\emptyset = \emptyset$ holds because $\mathcal{S}$ is linear, hence strict. This reflects the general fact that, if $s$ is a strict multirelation from $A$ to $B$ then, for all $t \in \underline{\mathrm{Rel}}(B, C)$, $([], \gamma)$ in $t \circ s$ iff $([], \gamma) \in t$. Such a phenomenon was also noted by Girard in his interpretation of system $T$ in coherence spaces [22]. His evidence that there was no interpretation of the iteration operator using the linear successor relied on a coherence argument. The previous lemma is stronger: it holds in any webbed model as soon as the interpretation of successor is strict.

In short, strict morphisms cannot produce anything *ex nihilo*; but the successor of any natural number should be marked as non-zero, for the iterator to distinguish between both cases. Hence the successor should not be strict: similarly to Girard's solution, we will interpret $\mathsf{Nat}$ by so-called *lazy* natural numbers. Let $\mathcal{N}_l = (|\mathcal{N}_l|, \mathfrak{P}_{\mathrm{f}}(|\mathcal{N}_l|))$ be such that $|\mathcal{N}_l| = \mathbf{N} \cup \mathbf{N}^{>}$, where $\mathbf{N}^{>}$ is just a disjoint copy of $\mathbf{N}$. The elements of $\mathbf{N}^{>}$ are denoted by $k^{>}$, for $k \in \mathbf{N}$: $k^{>}$ represents a partial number, not fully determined but *strictly greater than* $k$. If $\nu \in |\mathcal{N}_l|$, we define $\nu^+$ as $k+1$ if $\nu = k$ and $(k+1)^{>}$ if $\nu = k^{>}$. Then we set $\mathcal{S}_l = \{([], 0^{>})\} \cup \{([\nu], \nu^+);\ \nu \in |\mathcal{N}_l|\}$, which is affine. Notice that $\mathcal{O} \in \mathfrak{F}(\mathcal{N}_l)$ and $\mathcal{S}_l \in \mathfrak{F}(\mathcal{N}_l \Rightarrow \mathcal{N}_l)$.

We will show that these allow us to provide an interpretation of recursion, hence iteration, in system $T$: for all finiteness space $\mathcal{A}$, there exists

$$\mathcal{R}_{\mathcal{A}} \in \mathfrak{F}(\mathcal{N}_l \Rightarrow (\mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A})$$

such that, in $\Lambda_{\underline{\mathrm{Fin}}}$,

$$y : \mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}, z : \mathcal{A} \vdash \mathcal{R}\,\mathcal{O}\,y\,z \simeq z : \mathcal{A}$$

and

$$x : \mathcal{N}_l, y : \mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}, z : \mathcal{A} \vdash \mathcal{R}\,(\mathcal{S}_l\,x)\,y\,z \simeq y\,x\,(\mathcal{R}\,x\,y\,z) : \mathcal{A}.$$

## 4.2. Fixpoints

For all finiteness space $\mathcal{A}$, write $Rec[\mathcal{A}] = \mathcal{N}_l \Rightarrow (\mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}$. We want to introduce a recursion operator $\mathcal{R}_{\mathcal{A}} \in \mathfrak{F}(Rec[\mathcal{A}])$ intuitively subject to the following definition:

$$\mathcal{R}\,t\,u\,v = \text{match } t \text{ with } \left\{ \begin{array}{rcl} \mathsf{O} & \mapsto & v \\ \mathsf{S}\,t' & \mapsto & u\,t'\,(\mathcal{R}\,t'\,u\,v) \end{array} \right. .$$

This definition is recursive, and a natural method to obtain such an operator is as the fixpoint of

$$\mathcal{S}tep = \lambda\mathcal{X}\,\lambda x\,\lambda y\,\lambda z\left(\text{match } x \text{ with } \left\{ \begin{array}{lcl} \mathsf{O} & \mapsto & z \\ \mathsf{S}\,x' & \mapsto & y\,x'\,(\mathcal{X}\,x'\,y\,z) \end{array} \right.\right).$$

The cartesian closed category $\underline{\mathrm{Rel}}$ is cpo-enriched, the order on morphisms being inclusion. Hence it has fixpoints at all types: for all set $A$ and $f \in \underline{\mathrm{Rel}}(A,A)$, the least fixpoint of $f$ is $\bigcup_{k\geq 0} f^k\,\emptyset$, which is an increasing union. The least fixpoint operator is itself definable as the supremum of its approximants, $\mathcal{F}ix_A = \bigcup_{k\geq 0}\mathcal{F}ix_A^{(k)}$, where $\mathcal{F}ix_A^{(0)} = \emptyset$ and $\mathcal{F}ix_A^{(k+1)} = \lambda f\left(f\left(\mathcal{F}ix_A^{(k)}\,f\right)\right)$. More explicitly, $\mathcal{F}ix_A^{(k+1)} = \left\{([[\alpha_1,\ldots,\alpha_n],\alpha) + \sum_{i=1}^n \overline{\varphi}_i, \alpha)\,;\ \forall i,\ (\overline{\varphi}_i,\alpha_i) \in \mathcal{F}ix_A^{(k)}\right\}$: the reader may check that this equation holds by inspecting all the possible annotations of $\lambda f\,(f\,(a\,f))$ by the rules of Figure 2, where $a$ is any constant of type $(A\to A)\to A$. Intuitively, the pair $([\alpha_1,\ldots,\alpha_n],\alpha)$ corresponds to the head instance of $f$ in $f\,(a\,f)$, while each $\overline{\varphi}_i$ holds the instances of $f$ used by $a$ in order to produce $\alpha_i$ in the recursive call $(a\,f)$.

Notice that these approximants are finitary: if $\mathcal{A}$ is a finiteness space then, for all $k$, $\mathcal{F}ix_{\mathcal{A}}^{(k)} = \mathcal{F}ix_{|\mathcal{A}|}^{(k)} \in \mathfrak{F}\,((\mathcal{A}\Rightarrow\mathcal{A})\Rightarrow\mathcal{A})$. The fixpoint, however, is not finitary in general: for instance $\mathcal{F}ix\,\mathcal{S}_l = \mathbf{N}^{>} \notin \mathfrak{F}\,(\mathcal{N}_l)$ hence $\mathcal{F}ix \notin \mathfrak{F}\,((\mathcal{N}_l\Rightarrow\mathcal{N}_l)\Rightarrow\mathcal{N}_l)$. In fact $\mathcal{F}ix \in \mathfrak{F}\,((\mathcal{A}\Rightarrow\mathcal{A})\Rightarrow\mathcal{A})$ only for the empty type $\mathcal{A} = \mathcal{E}$:

**Lemma 4.1.** *If $|\mathcal{A}| \neq \emptyset$, then $\mathcal{F}ix \notin \mathfrak{F}\,((\mathcal{A}\Rightarrow\mathcal{A})\Rightarrow\mathcal{A})$.*

*Proof.* Let $\alpha \in |\mathcal{A}|$ and $f = \{([],\alpha)\}\cup\{([\alpha],\alpha)\} \in \mathfrak{P}_{\mathrm{f}}\,(\mathcal{A}\Rightarrow\mathcal{A}) \subseteq \mathfrak{F}\,(\mathcal{A}\Rightarrow\mathcal{A})$. Observe that $([([],\alpha)],\alpha) \in \mathcal{F}ix^{(1)}$, $([([],\alpha),([\alpha],\alpha)],\alpha) \in \mathcal{F}ix^{(2)}$, and more generally $([([],\alpha)] + n\,[([\alpha],\alpha)],\alpha) \in \mathcal{F}ix^{(n+1)}$. Hence $f^! \cap \left(\mathcal{F}ix^{\perp}\cdot\{\alpha\}\right)$ is infinite although $f \in \mathfrak{F}\,(\mathcal{A}\Rightarrow\mathcal{A})$, which contradicts the definition of finitary multirelations. □

This result indicates that the finitary semantics refuses infinite computations and will not accommodate general recursion. It is thus very natural to investigate the nature of the algorithms that can be studied in a finitary setting, hence our interest in typed recursion and system $T$. So we proceed in two steps: we first introduce the finitary approximants $\mathcal{R}_{\mathcal{A}}^{(k)} \in \mathfrak{F}\,(Rec\,[\mathcal{A}])$ by $\mathcal{R}_{\mathcal{A}}^{(k)} = \mathcal{S}tep_{\mathcal{A}}^k\,\emptyset$, then we prove $\mathcal{R}_{\mathcal{A}} = \bigcup_{k\geq 0}\mathcal{R}_{\mathcal{A}}^{(k)} \in \mathfrak{F}\,(Rec\,[\mathcal{A}])$.

On a side note, recall that coherence spaces accommodate fixpoints without difficulty [22]: in particular, the coherence semantics of system $T$ is very similar to the relational interpretation we give in the following. By contrast, the finiteness property is completely new. We will moreover show in section 5 that, despite its similarity with the coherence semantics, the relational interpretation itself brings new light on the nature of iteration and recursion, which would not be accessible in the uniform setting of coherence spaces.

### 4.3. Pattern matching on lazy natural numbers

We introduce a finitary operator $\mathcal{C}ase$, intuitively defined as:

$$\mathcal{C}ase\, t\, u\, v = \text{match } t \text{ with } \left\{ \begin{array}{lcl} \mathsf{O} & \mapsto & v \\ \mathsf{S}\, t' & \mapsto & u\, t' \end{array} \right. .$$

More formally:

**Definition 4.2.** *If $\overline{\nu} = [\nu_1, \ldots, \nu_k] \in |\mathcal{N}_l|^!$, we write $\overline{\nu}^+ = \left[\nu_1^+, \ldots, \nu_n^+\right]$. Then for all set $A$, let*

$$\begin{array}{rcl} \mathcal{C}ase_A & = & \{([0], [], [\alpha], \alpha);\ \alpha \in A\} \\ & \cup & \left\{([0^{>}] + \overline{\nu}^+, [(\overline{\nu}, \alpha)], [], \alpha);\ \overline{\nu} \in |\mathcal{N}_l|^! \wedge \alpha \in A\right\}. \end{array}$$

**Lemma 4.3.** *The operator $\mathcal{C}ase$ performs pattern matching on natural numbers:*

$$y : \mathcal{N}_l \Rightarrow \mathcal{A}, z : \mathcal{A} \vdash \mathcal{C}ase\, \mathcal{O}\, y\, z \simeq z : \mathcal{A}$$

*and*

$$x : \mathcal{N}_l, y : \mathcal{N}_l \Rightarrow \mathcal{A}, z : \mathcal{A} \vdash \mathcal{C}ase\, (\mathcal{S}_l\, x)\, y\, z \simeq y\, x : \mathcal{A}.$$

*Moreover, pattern matching is finitary:*

$$\mathcal{C}ase_{\mathcal{A}} = \mathcal{C}ase_{|\mathcal{A}|} \in \mathfrak{F}\left(\mathcal{N}_l \Rightarrow (\mathcal{N}_l \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}\right).$$

*Proof.* That the equations hold is by now a routine exercise. To prove that $\mathcal{C}ase$ is finitary, we check the definition of $\mathfrak{F}(- \Rightarrow -)$. For the first direction: for all $n \in \mathfrak{F}(\mathcal{N}_l)$, $\mathcal{C}ase\, n \subseteq \{([], [\alpha], \alpha);\ \alpha \in |\mathcal{A}|\} \cup \left\{([(\overline{\nu}, \alpha)], [], \alpha);\ \overline{\nu}^+ \in n^! \wedge \alpha \in |\mathcal{A}|\right\}$; hence, setting $n' = \{\nu;\ \nu^+ \in n\} \in \mathfrak{F}(\mathcal{N}_l)$, we obtain $\mathcal{C}ase\, n \subseteq (\lambda y\, \lambda z\, z) \cup (\lambda y\, \lambda z\, (y\, n'))$, and we conclude since the union of two finitary subsets is finitary. In the reverse direction, we prove that, for all $\gamma \in |(\mathcal{N}_l \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}|$, setting $N' = \mathcal{C}ase^{\perp} \cdot \{\gamma\}$, $n^! \cap N'$ is finite: this is immediate because $N'$ has at most one element (by the definition of $\mathcal{C}ase$). □

### 4.4. A relational recursor

We introduce the relation $\mathcal{R}$ as the fixpoint of $\mathcal{S}tep$.

**Definition 4.4.** *Fix a set $A$. Let*

$$\mathcal{S}tep_A = \lambda\mathcal{X}\, \lambda x\, \lambda y\, \lambda z\, (\mathcal{C}ase_A\, x\, (\lambda x'\, (y\, x'\, (\mathcal{X}\, x'\, y\, z)))\, z)$$

*and, for all $k \in \mathbf{N}$, let $\mathcal{R}_A^{(k)} = \mathcal{S}tep_A^k\, \emptyset$. Then we define $\mathcal{R}_A = \bigcup_{k \geq 0} \mathcal{R}_A^{(k)}$, and fix $[\![\mathsf{R}]\!] = \mathcal{R}$.*

**Lemma 4.5.** *For all finiteness space $\mathcal{A}$, $\mathcal{S}tep_{\mathcal{A}} = \mathcal{S}tep_{|\mathcal{A}|} \in \mathfrak{F}(Rec[\mathcal{A}] \Rightarrow Rec[\mathcal{A}])$ and, for all $k$, $\mathcal{R}_{\mathcal{A}}^{(k)} = \mathcal{R}_{|\mathcal{A}|}^{(k)} \in \mathfrak{F}(Rec[\mathcal{A}])$. Moreover, we have: $\mathcal{R}_{\mathcal{A}}^{(0)} = \emptyset$ and*

$$\begin{array}{rcl}\mathcal{R}_{\mathcal{A}}^{(k+1)} & = & \{([0], [], [\alpha], \alpha);\ \alpha \in |\mathcal{A}|\} \quad \cup \\ & & \Big\{ \left([0^{>}] + \sum_{i=0}^{n} \overline{\nu}_i^{+}, [(\overline{\nu}_0, [\alpha_1, \ldots, \alpha_n], \alpha)] + \sum_{i=1}^{n} \overline{\varphi}_i, \sum_{i=1}^{n} \overline{\alpha}_i, \alpha\right); \\ & & \qquad \forall i,\ (\overline{\nu}_i, \overline{\varphi}_i, \overline{\alpha}_i, \alpha_i) \in \mathcal{R}_{\mathcal{A}}^{(k)} \Big\}.\end{array}$$

*Proof.* The finiteness of the approximants follows from Theorem 3.2. The explicit description of $\mathcal{R}_{\mathcal{A}}^{(k)}$ is a direct application of the definition of the relational semantics. □

**Theorem 4.6** (Correctness)**.** *For all suitable $\Gamma$ and $\Delta$, $[\![\mathsf{R}\,\mathsf{O}\,y\,z]\!]_{\Gamma} = [\![z]\!]_{\Gamma}$ and $[\![\mathsf{R}\,(\mathsf{S}\,x)\,y\,z]\!]_{\Delta} = [\![y\,x\,(\mathsf{R}\,x\,y\,z)]\!]_{\Delta}$.*

*Proof.* This follows directly from Lemma 4.3 and the fact that $\mathcal{R} = \mathcal{S}tep\,\mathcal{R}$. □

## 4.5. Finiteness

It only remains to prove $\mathcal{R}$ is finitary. Following the definition of $(- \Rightarrow -)$, we proceed in two steps: the image of a finitary subset of $\mathcal{N}_l$ is finitary; conversely, the preimage of a singleton is "anti-finitary".

**Definition 4.7.** *If $n \in \mathfrak{F}(\mathcal{N}_l)$, we set $\max(n) = \max\{k;\ k \in n \vee k^{>} \in n\}$, with the convention $\max(\emptyset) = 0$. Then if $\overline{\nu} \in |\mathcal{N}_l|^{!}$ we set $\max(\overline{\nu}) = \max(\mathrm{Supp}(\nu))$, and if $N \subseteq n^{!}$ for some $n \in \mathfrak{F}(\mathcal{N}_l)$, $\max(N) = \max\left(\bigcup_{\overline{\nu} \in N} \mathrm{Supp}(\overline{\nu})\right)$.*

**Lemma 4.8.** *For all $\gamma = (\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{R}_{\mathcal{A}}$, $\gamma \in \mathcal{R}_{\mathcal{A}}^{(\max(\overline{\nu})+1)}$.*

*Proof.* By induction on $\max(\overline{\nu})$, using Lemma 4.5. □

**Lemma 4.9.** *If $n \in \mathfrak{F}(\mathcal{N}_l)$, then $\mathcal{R}_{\mathcal{A}}\,n \in \mathfrak{F}((\mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A})$.*

*Proof.* The previous Lemma entails $\mathcal{R}_{\mathcal{A}}\,n = \mathcal{R}_{\mathcal{A}}^{(\max(n)+1)}\,n$. We conclude recalling that $\mathcal{R}_{\mathcal{A}}^{(\max(n)+1)} \in \mathfrak{F}(Rec[\mathcal{A}])$. □

**Definition 4.10.** *For all $\overline{\varphi} = [(\overline{\nu}_1, \overline{\alpha}_1, \alpha_1), \ldots, (\overline{\nu}_k, \overline{\alpha}_k, \alpha_k)] \in |\mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}|^{!}$, let $\Diamond\overline{\varphi} = \sum_{j=1}^{k} \#\,\overline{\nu}_j$.*

**Lemma 4.11.** *If $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{R}_{\mathcal{A}}$, then $\#\,\overline{\nu} = \#\,\overline{\alpha} + \#\,\overline{\varphi} + \Diamond\overline{\varphi}$.*

*Proof.* Using Lemma 4.5, the result is proved for all $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{R}_{\mathcal{A}}^{(k)}$, by induction on $k$. □

**Theorem 4.12** (The recursion operator is finitary)**.** $\mathcal{R}_{\mathcal{A}} \in \mathfrak{F}(Rec[\mathcal{A}])$.

*Proof.* By Lemma 4.9, we are left to prove that, for all $n \in \mathfrak{F}(\mathcal{N}_l)$ and $\gamma = (\overline{\varphi}, \overline{\alpha}, \alpha) \in |(\mathcal{N}_l \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}|$, $N = n^{!} \cap \left(\mathcal{R}^{\perp} \cdot \{\gamma\}\right)$ is finite. But by Lemma 4.11,

$$N \subseteq \left\{\overline{\nu} \in |\mathcal{N}_l|^{!};\ \#\,\overline{\nu} = \#\,\overline{\alpha} + \#\,\overline{\varphi} + \Diamond\overline{\varphi} \wedge \max(\overline{\nu}) \leq \max(n)\right\}$$

which is finite. □

**Remark 4.13.** *We keep calling $\mathcal{R}$ "the" recursion operator, but notice such an operator is not unique in* Rel *or* Fin*: let $\mathcal{C}ase'_A = \{([0,0],[],[\alpha],\alpha);\ \alpha \in A\} \cup \left\{\left([0^{>}] + \overline{\nu}^{+}, [(\overline{\nu},\alpha)],[],\alpha\right);\ \overline{\nu} \in |\mathcal{N}_l|^{!} \wedge \alpha \in A\right\}$, for instance; this variant of matching operator behaves exactly like $\mathcal{C}ase$, and one can reproduce our construction of the recursor based on that. This is to be related with the fact that neither* Rel *nor* Fin *admit coproducts: in other terms, there is no canonical way to implement sum types and pattern matching in these categories.*

## 5. About iteration

### 5.1. A weak natural number object

We have just provided a semantics of system $T$ with recursor. Now let $\mathcal{I}_A = \lambda x\,\lambda y\,\lambda z\,(\mathcal{R}_A\,x\,(\lambda x'\,y)\,z)$ for all set $A$. By Theorem 4.12, $\mathcal{I}_{\mathcal{A}} = \mathcal{I}_{|\mathcal{A}|} \in \mathfrak{F}\,(Iter\,[\mathcal{A}])$, where $Iter\,[\mathcal{A}] = \mathcal{N}_l \Rightarrow (\mathcal{A} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{A}$. Moreover, by Theorem 4.6 this defines an iteration operator and we obtain that the triple $(|\mathcal{N}_l|, \mathcal{O}, \mathcal{S}_l)$, resp. $(\mathcal{N}_l, \mathcal{O}, \mathcal{S}_l)$, is a weak natural number object [20,21] in the cartesian closed category Rel, resp. Fin. Indeed:

**Lemma 5.1.** *For all $f \in \underline{\mathrm{Fin}}(\mathcal{A},\mathcal{A})$ and all $a \in \mathfrak{F}\,(\mathcal{A})$, there exists $h \in \underline{\mathrm{Fin}}(\mathcal{N}_l,\mathcal{A})$ such that $h\,\mathcal{O} = a$ and $h \circ \mathcal{S}_l = f \circ h$.*

*Proof.* Take $h = \lambda x\,(\mathcal{I}\,x\,f\,a)$. □

According to Remark 4.13, there is no hope of finding a strong natural number object in these categories, *i.e.* to require $h$ to be unique in the above lemma.

We could also have introduced $\mathcal{I}$ by a construction similar to that of $\mathcal{R}$:

**Definition 5.2.** *Let*

$$\begin{array}{rcl} \mathcal{I}t\mathcal{S}tep & = & \lambda\mathcal{X}\,\lambda x\,\lambda y\,\lambda z\,(\mathcal{C}ase\,x\,(\lambda x'\,(y\,(\mathcal{X}\,x'\,y\,z)))\,z) \\ & \in & \mathfrak{F}\,(Iter \Rightarrow Iter) \end{array}$$

*and, for all $k \in \mathbf{N}$, $\mathcal{I}^{(k)} = \mathcal{I}t\mathcal{S}tep^{k}\,\emptyset \in \mathfrak{F}\,(Iter)$.*

**Lemma 5.3.** *The relations $\mathcal{I}^{(k)}_{\mathcal{A}}$ are given by: $\mathcal{I}^{(0)}_{\mathcal{A}} = \emptyset$ and*

$$\mathcal{I}^{(k+1)}_{\mathcal{A}} = \{([0],[],[\alpha],\alpha);\ \alpha \in |\mathcal{A}|\} \cup \left\{ \begin{array}{r} \left([0^{>}] + \sum_{i=1}^{n} \overline{\nu}_i^{+}, [([\alpha_1,\ldots,\alpha_n],\alpha)] + \sum_{i=1}^{n} \overline{\varphi}_i, \sum_{i=1}^{n} \overline{\alpha}_i, \alpha\right); \\ \forall i,\ (\overline{\nu}_i,\overline{\varphi}_i,\overline{\alpha}_i,\alpha_i) \in \mathcal{I}^{(k)}_{\mathcal{A}} \end{array} \right\}.$$

*Proof.* Again, routine exercise. □

**Lemma 5.4.** *We have $\bigcup_{k\geq 0} \mathcal{I}^{(k)}_{\mathcal{A}} = \mathcal{I}_{\mathcal{A}}$.*

*Proof.* Check that, for all $k$, $\mathcal{I}^{(k)} = \lambda x\,\lambda y\,\lambda z\,\left(\mathcal{R}^{(k)}\,x\,(\lambda x'\,y)\,z\right)$. □

### 5.2. Uniformity of iteration

We now develop a semantic investigation of the gap in expressive power between iteration and recursion, which was formally established by Parigot [23].

One distinctive feature of the (pure or finitary) relational model is non-uniformity: if $a, a' \in \mathfrak{F}(\mathcal{A})$ then $a \cup a' \in \mathfrak{F}(\mathcal{A})$; and in the construction of $a^!$, there is no restriction on the elements of the multisets we consider. It is very different from the setting of coherence spaces for instance. But we can show the iterator only considers uniform sets of lazy numbers, in the following sense: if $k \in \mathbf{N}$, we define $\underline{k} = \mathcal{S}_l^k \mathcal{O} = \{l^>;\ l < k\} \cup \{k\} \in \mathfrak{F}(\mathcal{N}_l)$; we say $n \subseteq |\mathcal{N}_l|$ is *uniform* if $n \subseteq \underline{k}$ for some $k$. Notice that, in the coherence space of lazy natural numbers used by Girard in [22] to interpret system $T$, the sets $\underline{k}$ are the finite maximal cliques: coherence is given by $k \subset l$ iff $k = l$, $k \subset l^>$ iff $k > l$ and $k^> \subset l^>$ for all $k, l$. The only infinite maximal clique is $\mathbf{N}^>$ which is not finitary (recall this is the fixpoint of $\mathcal{S}_l$). We prove $\mathcal{I}$ considers only uniform sets of lazy numbers.

Let $\mathcal{S}tage_{\mathcal{A}}^{(0)} = \{([0], [], [\alpha], \alpha);\ \alpha \in |\mathcal{A}|\}$ and, for all $k \in \mathbf{N}$,

$$\begin{array}{rcl} \mathcal{S}tage_{\mathcal{A}}^{(k+1)} & = & \{([0], [], [\alpha], \alpha);\ \alpha \in |\mathcal{A}|\} \cup \\ & & \left\{ \begin{array}{r} \left([0^>] + \sum_{i=1}^n \overline{\nu}_i^+, [([\alpha_1, \ldots, \alpha_n], \alpha)] + \sum_{i=1}^n \overline{\varphi}_i, \sum_{i=1}^n \overline{\alpha}_i, \alpha\right); \\ \forall i,\ (\overline{\nu}_i, \overline{\varphi}_i, \overline{\alpha}_i, \alpha_i) \in \mathcal{S}tage_{\mathcal{A}}^{(k)} \end{array} \right\}. \end{array}$$

Setting $\mathcal{S}tage_{\mathcal{A}}^{(-1)} = \emptyset$, one can check that $\mathcal{S}tage_{\mathcal{A}}^{(k-1)} \subseteq \mathcal{I}_{\mathcal{A}}^{(k)} \subseteq \mathcal{S}tage_{\mathcal{A}}^{(k)}$ for all $k$, and then $\mathcal{I}_{\mathcal{A}} = \bigcup_{k \geq 0} \mathcal{S}tage_{\mathcal{A}}^{(k)}$.

**Lemma 5.5.** *If $\mathcal{A} \neq \mathcal{E}$ then, for all $k \in \mathbf{N}$,*

$$\bigcup \left\{ \mathrm{Supp}(\overline{\nu});\ \exists (\overline{\varphi}, \overline{\alpha}, \alpha),\ (\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{S}tage_{\mathcal{A}}^{(k)} \setminus \mathcal{S}tage_{\mathcal{A}}^{(k-1)} \right\} = \underline{k}.$$

*Proof.* The first inclusion ($\subseteq$) is easy by induction on $\underline{k}$. For the reverse ($\supseteq$), check that $([0^>, 1^>, \ldots, (k-1)^>, k], (k-1)[([\alpha], \alpha)], [\alpha], \alpha) \in \mathcal{S}tage_{\mathcal{A}}^{(k)} \setminus \mathcal{S}tage_{\mathcal{A}}^{(k-1)}$ for all $\alpha \in |\mathcal{A}|$. □

As a consequence, for all $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{I}$, $\mathrm{Supp}(\overline{\nu})$ is uniform. Of course, no such property holds for $\mathcal{R}$, because

$$\mathcal{R}_{\mathcal{A}}^{(1)} \supseteq \left\{ \left( \left[0^>\right] + \overline{\nu}^+, [(\overline{\nu}, [], \alpha)], [], \alpha \right);\ \alpha \in |\mathcal{A}| \wedge \overline{\nu} \in |\mathcal{N}_l|^! \right\}.$$

Actually, this non-uniformity of recursion is not related with our specific choice of the interpretation of $\mathsf{R}$. Rather, it follows from the fact that, in the equation $\mathsf{R}\,(\mathsf{S}\,t)\,u\,v \simeq u\,t\,(\mathsf{R}\,t\,u\,v)$, $u$ can take arbitrary arguments. Let us make this remark precise:

**Lemma 5.6.** *Fix a set $A \neq \emptyset$, $\alpha \in A$ and let $r$ be a relation of type $|\mathcal{N}_l| \to (|\mathcal{N}_l| \to A \to A) \to A \to A$, such that $r\,(\mathcal{S}_l\,t)\,u\,v \simeq u\,t\,(r\,t\,u\,v)$ for all terms $t$, $u$ and $v$ of suitable type in $\Lambda_{\mathrm{Rel}}$. Then, for all $\overline{\nu} \in |\mathcal{N}_l|^!$, $r$ admits an element of the form $(\overline{\nu}^+ + [0^>, \ldots, 0^>], \overline{\phi}, [], \alpha) \in r$.*

*Proof.* Fix $\alpha \in A$ and let $f = \{(\overline{\nu}, [\,], \alpha)\}$, which is a singleton relation of type $|\mathcal{N}_l| \to A \to A$. Then, if $x$ is a variable of type $|\mathcal{N}_l|$, $r\ (\mathcal{S}_l\, x)\ f\, \emptyset \simeq f\, x\ (r\, x\, f\, \emptyset)$ holds in $\Lambda_{\underline{\mathrm{Rel}}}$. Now observe that, by the definition of $f$, $[\![f\, x\ (r\, x\, f\, \emptyset)]\!]_{x:|\mathcal{N}_l|} = \{(\overline{\nu}, \alpha)\}$. Hence $x^{\overline{\nu}} : |\mathcal{N}_l| \vdash (r\ (\mathcal{S}_l\, x)\ f\, \emptyset)^{\alpha} : A$. By inspecting the rules of Figure 2, we necessarily have $x^{\overline{\nu}} : |\mathcal{N}_l| \vdash (r\ (\mathcal{S}_l\, x)\ f)^{([],\alpha)} : A \to A$. By the same argument, we obtain $x^{\overline{\nu}} : |\mathcal{N}_l| \vdash (r\ (\mathcal{S}_l\, x))^{(\overline{\varphi},[],\alpha)} : (|\mathcal{N}_l| \to A \to A) \to A \to A$ for some $\overline{\varphi} \in f^!$. Finally, there exist finite multisets $[\mu_1, \ldots, \mu_k], \overline{\rho}_1, \ldots, \overline{\rho}_k \in |\mathcal{N}_l|^!$ such that $x^{[]} : |\mathcal{N}_l| \vdash r^{([\mu_1,\ldots,\mu_k],\overline{\varphi},[],\alpha)} : |\mathcal{N}_l| \to (|\mathcal{N}_l| \to A \to A) \to A \to A$ and $x^{\overline{\rho}_j} : |\mathcal{N}_l| \vdash (\mathcal{S}_l\, x)^{\mu_j} : |\mathcal{N}_l|$ for all $j$, with moreover $\sum_j \overline{\rho}_j = \overline{\nu}$. Since $(\overline{\rho}_j, \mu_j) \in \mathcal{S}_l$ for all $j$, we conclude that $[\mu_1, \ldots, \mu_k]$ is of the form $\overline{\nu}^+ + [0^>, \ldots, 0^>]$. □

The contrast between Lemma 5.5 and Lemma 5.6 reflects a gap in expressive power between iteration and recursion. Indeed, from Lemma 5.5, we can deduce the following result:

**Lemma 5.7.** *Consider the iterator variant of system $T$, and fix the interpretations $[\![\mathsf{Nat}]\!] = \mathcal{N}_l$, $[\![\mathsf{O}]\!] = \mathcal{O}$, $[\![\mathsf{S}]\!] = \mathcal{S}_l$ and $[\![\mathsf{I}]\!] = \mathcal{I}$. Fix a variable $x$ of type $\mathsf{Nat}$. Let $s$ be a term such that:*

*(i) all occurrences of $x$ are of the form $\mathsf{I}\, x$;*
*(ii) no occurrence of $\mathsf{I}\, x$ is inside the argument of an application.*

*Then, if some judgement $\Gamma, x^{\overline{\nu}} : \mathsf{Nat} \vdash s^{\alpha} : A$ is derivable,* $\mathrm{Supp}\,(\overline{\nu})$ *is uniform.*

*Proof.* We prove the result by induction on $s$, the only interesting case being that of an application:

- if $s = \mathsf{I}\, x$ then $A$ is of the form $(B \to B) \to B$, we check that $\Gamma, x^{\overline{\nu}} : \mathsf{Nat} \vdash \mathsf{I}\, x^{(\overline{\phi},\beta)} : (B \to B) \to B$ implies $(\overline{\nu}, \overline{\phi}, \alpha) \in \mathcal{I}$, and we conclude by Lemma 5.5;
- otherwise, $s = u\, v$ and $x$ does not occur in $v$, hence $\Gamma, x^{\overline{\nu}} : \mathsf{Nat} \vdash s^{\alpha} : A$ implies some judgment $\Gamma_0, x^{\overline{\nu}} : \mathsf{Nat} \vdash u^{([\gamma_1,\ldots,\gamma_k],\alpha)} : C \to A$ can be derived, and we conclude by induction hypothesis applied to $u$, which necessarily satisfies (i) and (ii).

□

**Corollary 5.8.** *No term of the form $\lambda x\, s$, with $s$ as in the previous lemma is a valid implemention of recursion.*

Notice that this already rules out the usual naïve attempt to implement recursion using only iteration and products (recall the term $\mathsf{rec}$ from section 3.4). We can moreover drop the hypothesis (ii) in Lemma 5.7 by considering the following oriented version of coherence on $|\mathcal{N}_l|$: we say $\mu$ is less defined than $\nu$ and write $\mu \prec \nu$ if there are $m < n \in \mathbf{N}$ such that $\mu = m^>$ and either $\nu = n^>$ or $\nu = n$. The coherence relation $\frown$ is the reflexive and symmetric closure of this partial order. We then obtain the following property for $\mathcal{I}$:

**Lemma 5.9.** *For all $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathcal{I}$,* $\mathrm{Supp}\,(\overline{\nu})$ *is downwards closed for $\prec$,* i.e. *$\mu \prec \nu \in \mathrm{Supp}\,(\overline{\nu})$ implies $\mu \in \mathrm{Supp}\,(\overline{\nu})$.*

*Proof.* By Lemma 5.3, it is sufficient to prove this result for $\mathcal{I}^{(k)}$, for all $k \in \mathbf{N}$. First notice that any union of downwards closed subsets of $|\mathcal{N}_l|$ is downwards closed, and that if $n \subseteq |\mathcal{N}_l|$ is downwards closed then so is $\{0^{>}\} \cup \{\nu^{+};\ \nu \in n\}$. We then conclude by a straightforward induction on $k$. □

Again, such a property fails for any implementation of recursion, by Lemma 5.6. Moreover:

**Lemma 5.10.** *With the same hypotheses as in Lemma 5.7, consider a term $t$ such that all occurrences of $x$ in $t$ are of the form $\mathsf{I}\,x$. Then, if some judgement $\Gamma, x^{\overline{\nu}} : \mathsf{Nat} \vdash t^{\alpha} : A$ is derivable,* $\mathrm{Supp}\,(\overline{\nu})$ *is downwards closed for $\prec$.*

*Proof.* Again, the proof is by induction on $t$, with the only interesting case being that of an application:

- if $t = \mathsf{I}\,x$, we conclude directly by the previous lemma;
- otherwise, $t = u\,v$ and $\Gamma, x^{\overline{\nu}} : \mathsf{Nat} \vdash t^{\alpha} : A$ implies that we can derive judgements of the form $\Gamma_0, x^{\overline{\nu}_0} : \mathsf{Nat} \vdash u^{([\beta_1,\ldots,\beta_k],\alpha)} : B \to A$ and $\Gamma_j, x^{\overline{\nu}_j} : \mathsf{Nat} \vdash v^{\beta_j} : B$ for $j \in \{1,\ldots,k\}$ where $\overline{\nu} = \sum_{j=0}^{k} \overline{\nu}_j$; we conclude by induction hypothesis applied to $u$ and $v$, together with the fact that unions of downwards closed subsets are downwards closed.

□

**Corollary 5.11.** *No term of the form $\lambda x\, t$, with $t$ as in the previous lemma is a valid implemention of recursion.*

This concluding result is not fully satisfactory, because there remains the condition that $x$ occurs only as the first argument of $\mathsf{I}$. It is still unclear to us whether our semantic argument can be refined in order to avoid this hypothesis. Notice for instance that $[\![\mathsf{I}\,x\,\emptyset\,x]\!]_{x:|\mathcal{N}_l|} = \{([0,\nu],\nu);\ \nu \in |\mathcal{N}_l|\}$ where $[0,\nu]$ need not be uniform nor downwards closed for $\prec$, hence those simple properties are not sufficient to discriminate between iterator-based terms and recursion operators.

On the other hand, the fact that no term in the iterator variant of system $T$ defines a recursion operator follows straightforwardly from Parigot's work on the encodings of system $T$ in higher order intuitionistic logic [23]: Parigot's results entail that no predecessor can be derived from the iterator. A predecessor is a term $p$ such that $p\,\mathsf{O} \simeq \mathsf{O}$ and $p\,(\mathsf{S}\,t) \simeq t$ for all $t$ of type $\mathsf{Nat}$. Such a predecessor is trivially obtained from the recursor: consider $p = \lambda x\,(\mathsf{R}\,x\ (\lambda y\,y)\ \mathsf{O})$. Hence Parigot's results subsume both Corollaries 5.8 and 5.11.

Anyway, the results of this section showed how the relational interpretation of system $T$ reflected this well known computational distinction between iteration and recursion: Lemmas 5.5 and 5.10 on the one hand, and Lemma 5.6 on the other hand, provide a semantic insight into the nature of this gap, if not a new proof of it.

## Related and future work

Our success in defining a semantics of system $T$ in $\underline{\mathrm{Fin}}$ immediately poses the question of its generalization to other datatypes. The cpo-enriched structure of $\underline{\mathrm{Rel}}$ allowed for an abstract description of datatypes [30, 31] as particular functors which are monotonic for relation inclusion. In particular, this provided the basis of a categorical account of container types [32]. In such a setting, it is natural to define recursive datatypes, such as lists or trees, as the least fixpoints of particular Scott-continuous functors. The question is then how to transport these constructions to the finitary setting.

In recent work with Tasson [33], we study two orders on finiteness spaces derived from set inclusion: the most restrictive one, finiteness extension, was used by Ehrhard to provide an interpretation of second order linear logic in an unpublished preliminary version of [9]; the largest one, finiteness inclusion, is a cpo on finiteness spaces. We study various notions of continuity for functors in finiteness spaces, and relate them with the existence of fixpoints for these functors. A striking feature of this development is that we are led to consider the properties of functors w.r.t. both orders simultaneously: continuity for finiteness inclusion, and monotonicity for finiteness extension. We prove in particular that every functor obtained by applying a very generic construction on finiteness spaces satisfies these properties, and admits a least fixpoint for finiteness inclusion.

We then apply this result to describe a relational semantics of functional programming with recursive datatypes, which generalizes the results of the present paper: the fixpoints of algebraic datatype functors define recursive datatypes with finitary constructors, destructors and iterators. This paves the way towards a quantitative semantics of typed recursion, according to the following roadmap: first find a correct quantitative semantics for constructors *vs* destructors (zero and successor *vs* pattern matching in the case of system $T$); then check that the coefficients involved in the computation of the fixpoint defining iterators remain finite. We may first apply this strategy to system $T$. If it proves fruitful, we would then try to generalize it to recursive algebraic datatypes.

Notice that another standard approach to the semantics of datatypes is to consider the impredicative encoding of inductive types, e.g. in system $F$. Such a semantics in finiteness spaces can be derived from that of second order linear logic. This is based on a class of functors which, in particular, are monotonic for the finiteness extension order. Notice however that this does not provide a denotational semantics *stricto sensu*: in general, the interpretation decreases under cut elimination. Moreover, the possibility of defining a quantitative semantics in this setting is not clear.

## References

[1] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
[2] Dana S. Scott. Data types as lattices. *SIAM J. Comput.*, 5(3):522–587, 1976.

[3] Gérard Berry. Stable models of typed lambda-calculi. In Giorgio Ausiello and Corrado Böhm, editors, *ICALP*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 1978.
[4] Jean-Yves Girard. Normal functors, power series and lambda-calculus. *Annals of Pure and Applied Logic*, 37(2):129–177, 1988.
[5] Jean-Yves Girard. The system f of variable types, fifteen years later. *Theor. Comput. Sci.*, 45(2):159–192, 1986.
[6] Chantal Berline. Graph models of lambda-calculus at work, and variations. *Mathematical Structures in Computer Science*, 16(2):185–221, 2006.
[7] Richard Statman. Completeness, invariance and lambda-definability. *J. Symb. Log.*, 47(1):17–26, 1982.
[8] Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Thèse de doctorat, Université Aix-Marseille II, 2007.
[9] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.
[10] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309:1–41, 2003.
[11] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Electr. Notes Theor. Comput. Sci.*, 123:35–74, 2005.
[12] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine's machine and the Taylor expansion of $\lambda$-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *CiE*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2006.
[13] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. *Information and Computation*, 208(6):606–633, 2010.
[14] Paolo Tranquilli. Intuitionistic differential nets and lambda-calculus. *Theor. Comput. Sci.*, 412(20):1979–1997, 2011.
[15] Lionel Vaux. Differential linear logic and polarization. In Curien [34], pages 371–385.
[16] Christine Tasson. Algebraic totality, towards completeness. In Curien [34], pages 325–340.
[17] Michele Pagani and Christine Tasson. The inverse Taylor expansion problem in linear logic. In *LICS*, 2009.
[18] Daniel de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. Technical report, 2008. Rapport de recherche INRIA n° 6638.
[19] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 298–312. Springer Berlin, 2007.
[20] Marie-France Thibault. Pre-recursive categories. *Journal of Pure and Applied Algebra*, 24:79–93, 1982.
[21] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*. Cambridge University Press, New York, NY, USA, 1988.
[22] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. CUP, Cambridge, 1989.
[23] Michel Parigot. On the representation of data in lambda-calculus. In Egon Börger, Hans Kleine Büning, and Michael M. Richter, editors, *CSL*, volume 440 of *Lecture Notes in Computer Science*, pages 309–321. Springer, 1989.
[24] Lionel Vaux. The algebraic lambda calculus. *Mathematical Structures in Computer Science*, 19(5):1029–1059, 2009.
[25] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational model of a parallel and non-deterministic lambda-calculus. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009.
[26] Vincent Danos and Thomas Ehrhard. On probabilistic coherence spaces. Technical report, 2008.
[27] Pablo Arrighi and Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In Andrei Voronkov, editor, *Rewriting Techniques and Applications, 19th*

*International Conference, RTA-2008*, volume 5117 of *Lecture Notes in Computer Science*, pages 17–31, Hagenberg, Austria, July 15–17, 2008. Springer.

[28] Pierre-Louis Curien, Hugo Herbelin, Jean-Louis Krivine, and Paul-André Melliès. *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et synthèses*, pages 1–196. Société Mathématique de France, 2009.

[29] Martin Hyland and Andrea Schalk. Glueing and orthogonality for models of linear logic. *Theor. Comput. Sci.*, 294(1/2):183–231, 2003.

[30] Roland Carl Backhouse and Paul F. Hoogendijk. Generic properties of datatypes. In Roland Carl Backhouse and Jeremy Gibbons, editors, *Generic Programming*, volume 2793 of *Lecture Notes in Computer Science*, pages 97–132. Springer, 2003.

[31] Roland Carl Backhouse, Peter J. de Bruin, Paul F. Hoogendijk, Grant Malcolm, Ed Voermans, and Jaap van der Woude. Polynomial relators. In Maurice Nivat, Charles Rattray, Teodor Rus, and Giuseppe Scollo, editors, *AMAST*, Workshops in Computing, pages 303–326. Springer, 1991.

[32] Paul Hoogendijk and Oege De Moor. Container types categorically. *J. Funct. Program.*, 10:191–225, March 2000.

[33] Christine Tasson and Lionel Vaux. Transport of finiteness structures and applications. to appear in Mathematical Structures in Computer Science, 2011.

[34] Pierre-Louis Curien, editor. *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings*, volume 5608 of *Lecture Notes in Computer Science*. Springer, 2009.

Communicated by (The editor will be set by the publisher).
November 21, 2011.