Infinitary Lambda Calculi from a Linear Perspective

Ugo Dal Lago

Infinitary Lambda Calculi from a Linear Perspective (Long Version)

Ugo Dal Lago*

Abstract

We introduce a linear infinitary λ-calculus, called $\ell\Lambda_\infty$, in which two exponential modalities are available, the first one being the usual, finitary one, the other being the only construct interpreted coinductively. The obtained calculus embeds the infinitary applicative λ-calculus and is universal for computations over infinite strings. What is particularly interesting about $\ell\Lambda_\infty$, is that the refinement induced by linear logic allows to restrict both modalities so as to get calculi which are *terminating* inductively and *productive* coinductively. We exemplify this idea by analysing a fragment of $\ell\Lambda$ built around the principles of SLL and 4LL. Interestingly, it enjoys confluence, contrarily to what happens in ordinary infinitary λ-calculi.

1 Introduction

The λ-calculus is a widely accepted model of higher-order functional programs—it faithfully captures functions and their evaluation. Through the many extensions introduced in the last forty years, the λ-calculus has also been shown to be a model of imperative features, control [25], etc. Usually, this requires extending the class of terms with new operators, then adapting type systems in such a way that "good" properties (e.g., confluence, termination, etc.), and possibly new ones, hold.

This also happened when potentially infinite structures came into play. Streams and, more generally, coinductive data have found their place in the λ-calculus, following the advent of lazy data structures in functional programming languages like Haskell and ML. By adopting lazy data structures, the programmer has a way to represent infinity by finite means, relying on the fact that data are not completely evaluated, but accessed "on-demand". In presence of infinite structures, the usual termination property takes the form of *productivity* [9]: even if evaluating a stream expression globally takes infinite time, looking for the next symbol in it takes finite time.

All this has indeed been modelled in a variety of ways by enriching λ-calculi and related type theories. One can cite, among the many different works in this area, the ones by Parigot [26], Raffalli [27], Hughes at al. [16], or Abel [1]. Terms are *finite* objects, and the infinite nature of streams is modelled through a staged inspection of so-called thunks. The key ingredient in obtaining productivity and termination is usually represented by sophisticated type systems.

There is also another way of modelling infinite structures in λ-calculi, namely *infinitary rewriting*, where both terms and reduction sequences are not necessarily finite, and as a consequence infinity is somehow internalised into the calculus. Infinitary rewriting has been studied in the context of various concrete rewrite systems, including first-order term rewriting [18], but also systems of higher-order rewriting [20]. In the case of λ-calculus [19], the obtained model is very powerful, but does not satisfy many of the nice properties its finite siblings enjoy, including confluence and finite developments, let alone termination and productivity (which are anyway unexpected in the absence of types).

In this paper, we take a fresh look at infinitary rewriting, through the lenses of linear logic [13]. More specifically, we define an untyped, linear, infinitary λ-calculus called $\ell\Lambda_\infty$, and study its basic

*Università di Bologna & INRIA Sophia Antipolis

properties, how it relates to Λ_∞ [19], and its expressive power. As expected, incepting linearity does not allow *by itself* to solve any of the issues described above: $\ell\Lambda_\infty$ embeds Λ_{001}, a subsystem of Λ_∞, and as such does not enjoy confluence. On the other hand, linearity provides the right tools to *control* infinity: we delineate a simple fragment of $\ell\Lambda_\infty$ which has all the good properties one can expect, including productivity and confluence. Remarkably, this is not achieved through types, but rather by purely structural constraints on the way copying is managed, which is responsible for its bad behaviour. Expressivity, on the other hand, is not sacrificed.

1.1 Linearity vs. Infinity

The crucial rôle linearity has in this paper can be understood without any explicit reference to linear logic as a proof system, but through a series of observations about the usual, finitary, λ-calculus. In any λ-term M, the variable x can occur free more than once, or not at all. If the variable x occurs free exactly once in M *whenever* we form an abstraction $\lambda x.M$, the λ-calculus becomes strongly normalising: at any rewrite step the size of the term strictly decreases. On the other hand, the obtained calculus has a poor expressive power and, *in this form*, is not the model of any reasonable programming language. Let duplication reappear, then, but in controlled form: besides a *linear* abstraction $\lambda x.M$, (which is well formed only if x occurs free exactly once in M) there is also a *nonlinear* abstraction $\lambda!x.M$, which poses no constraints on the number of times x occurs in M. Moreover, and this is the crucial step, interaction with a nonlinear abstraction is restricted: the argument to a nonlinear application must itself be marked as *duplicable* (and *erasable*) for β to fire. This is again implemented by enriching the category of terms with a new construct: given a term M, the *box* $!M$ is a duplicable version of M. Summing up, the language at hand, call it $\ell\Lambda$, is built around applications, linear abstractions, nonlinear abstractions, and boxes. Moreover, β-reduction comes in two flavours:

$$(\lambda x.M)N \to M\{x/N\}; \qquad (\lambda!x.M)!N \to M\{x/N\}.$$

What did we gain by rendering duplicating and erasing operations explicit? Not much, apparently, since pure λ-calculus can be embedded into $\ell\Lambda$ in a very simple way following the so-called Girard's translation [13]: abstractions become nonlinear abstractions, and any application MN is translated into $M!N$. In other words, all arguments can be erased and copied, and we are back to the world of wild, universal, computation. Not everything is lost, however: in any nonlinear abstraction $\lambda!x.M$, x can occur any number of times in M, but at different *depths*: there could be linear occurrences of x, which do not lie in the scope of any box, i.e., at depth 0, but also occurrences of x at greater depths. Restricting the depths at which the bound variable can occur in nonlinear abstractions gives rise to strongly normalising fragments of $\ell\Lambda$. Some of them can be seen, through the Curry-Howard correspondence, as subsystems of linear logic characterising complexity classes. This includes light linear logic [15], elementary linear logic [8] or soft linear logic [21]. As an example, the exponential discipline of elementary linear logic can be formulated as follows in $\ell\Lambda$: for every nonlinear abstraction $\lambda!x.M$ all occurrences of x in M are at depth 1. Noticeably, this gives rise to a characterisation of elementary functions. Similarly, soft linear linear can be seen as the fragment of $\ell\Lambda_\infty$ in which all occurrences of x (if more than one) occur at depth 0 in any nonlinear abstraction $\lambda!x.M$.

But why could all this be useful when rewriting becomes infinitary? Infinitary λ-terms [19] are nothing more than infinite terms defined according to the grammar of the λ-calculus. In other words, one can have a term M such that M is syntactically equal to $\lambda x.xM$. Evaluation can still be modelled through β-reduction. Now, however, reduction sequences of infinite length (sometime) make sense: take $Y(\lambda x.\lambda y.yx)$, where Y is a fixed point combinator. It rewrites in infinitely many steps to the infinite term N such that $N = \lambda y.yN$. Are all infinite reduction sequences acceptable? Not really: an infinite reduction sequence is said to *converge* to the term M only if, informally, any finite approximation to M can be reached along a finite prefix of the sequence. In other words, reduction should be applied deeper and deeper, i.e., the *depth* of the fired redex should tend to infinity. But how could one define the depth of a subterm's occurrence in a given term? There are

many alternatives here, since, informally, one can choose to let the depth increase (or not) when entering the body of an abstraction or any of the two arguments of an application. Indeed, *eight* different calculi can be formed, each with different properties. For example, Λ_{001} is the calculus in which the depth only increases while entering the argument position in applications, while in Λ_{100} the same happens when crossing abstractions. The choice of where the depth increases is crucial not only when defining infinite reduction sequences, but also when defining *terms*, whose category is obtained by completing the set of finite terms with respect to a certain metric. So, not all infinite terms are well-formed. In Λ_{001}, as an example, the term $M = xM$ is well-formed, while $M = \lambda x.M$ is not. In Λ_{100}, the opposite holds. In all the obtained calculi, however, many of the properties one expects are not true: the Complete Developments Theorem (i.e. the infinitary analogue of the Finite Complete Theorem [4]) does not hold and, moreover, confluence fails except if formulated in terms of so-called Böhm reduction [19]. The reason is that Λ_∞ is even wilder than Λ: various forms of infinite computations can happen, but only some of them are benign. Is it that linearity could help in taming all this complexity, similarly to what happens in the finitary case? This paper gives a first positive answer to this question.

1.2 Contributions

The system we will study in the rest of this paper, called $\ell\Lambda_\infty$, is obtained by incepting ideas from infinitary calculi into $\ell\Lambda$. Not one but *two* kinds of boxes are available in $\ell\Lambda_\infty$, and the depth increases only while crossing boxes of the second kind (called *coinductive* boxes), while boxes of the first kind (dubbed *inductive*) leave the depth unchanged. As a consequence, boxes are as usual the only terms which can be duplicated and erased, but they are also responsible for the infinitary nature of the calculus: any term not containing (coinductive) boxes is necessarily finite. Somehow, the depths in the sense of Λ_∞ and of $\ell\Lambda$ coincide.

Besides introducing $\ell\Lambda_\infty$ and proving its basic properties, this paper explores the expressive power of the obtained computational model, showing that it suffers from the same problems affecting Λ_∞, but also that it has precisely the same expressive power as that of Type-2 Turing machines, the reference computational model of so-called computable analysis [31].

The most interesting result among those we give in this paper consists in showing that, indeed, a simple fragment of $\ell\Lambda_\infty$, called $\ell\Lambda_\infty^{\mathsf{4S}}$, is on the one hand flexible enough to encode streams and guarded recursion on them, and on the other guarantees productivity. Remarkably, confluence holds, contrary to what happens for $\ell\Lambda_\infty$. Actually, $\ell\Lambda_\infty^{\mathsf{4S}}$ is defined around the same principles which lead to the definition of light logics. Each kind of box, however, follows a distinct discipline: inductive boxes are handled as in Lafont's SLL [21], while coinductive boxes follow the principles of $\mathsf{4LL}$ [8], hence the name of the calculus. So far, the $\mathsf{4LL}$'s exponential discipline has not been shown to have any computational meaning: now we know that beyond it there is a form of guarded corecursion.

2 $\ell\Lambda_\infty$ and its Basic Properties

In this section, we introduce a linear infinitary λ-calculus called $\ell\Lambda_\infty$, which is the main object of study of this paper. Some of the dynamical properties of $\ell\Lambda_\infty$ will be investigated. Before defining $\ell\Lambda_\infty$, some preliminaries about formal systems with *both* inductive and coinductive rules will be given.

2.1 Mixing Induction and Coinduction in Formal Systems

A formal system over a set of judgments $\mathcal{S}$ is given by a finite set of rules, all of them having one conclusions and an arbitrary finite number of premises. The rules of a *mixed* formal system S are of two kinds: those which are to be interpreted inductively and those which are to be interpreted *co*inductively. To distinguish between the two, inductive rules will be denoted as usual, with a

single line, while coinductive rules will be indicated as follows: $\dfrac{\vdash A_1 \quad \ldots \quad \vdash A_n}{\vdash B}$. Intuitively, any correct derivation in such a system is an *infinite* tree built following inductive and coinductive rules where, however, any *infinite* branch crosses coinductive rule instances *infinitely* often. In other words, there cannot be any infinite branch where, from a certain point on, only inductive rules occur.

Formally, the set $\mathbb{C}_{\mathsf{S}}$ of derivable assertions of a mixed formal system S over the set of judgments $\mathcal{S}$ can be seen as the greatest fixpoint of the function $I_{\mathsf{S}} \circ C_{\mathsf{S}}$ where $C_{\mathsf{S}} : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ is the monotone function induced by the the application of a *single* coinductive rule, while I_{S} is the function induced by the application of inductive rules an arbitrary *but finite* number of times. I_{S} can itself be obtained as the least fixpoint of a monotone functional on the space of monotone functions on $\mathcal{P}(\mathcal{S})$. The existence of $\mathbb{C}_{\mathsf{S}}$ can be formally justified by Knaster-Tarski theorem, since all the involved spaces are complete lattices, and the involved functionals are monotone. This is, by the way, very close to the approach from [10]. Let $\mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S})$ be the set of monotone functions on the powerset $\mathcal{P}(\mathcal{S})$. A relation $\sqsubseteq$ on $\mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S})$ can be easily defined by stipulating that $F \sqsubseteq G$ iff for every $X \subseteq \mathcal{S}$ it holds that $F(X) \subseteq G(X)$. The structure $(\mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S}), \sqsubseteq)$ is actually a complete lattice, because:

- The relation $\sqsubseteq$ is a partial order. In particular, antisymmetry is a consequence of function extensionality: if for every X, both $F(X) \subseteq G(X)$ and $G(X) \subseteq F(X)$, then F and G are the same function.
- Given a set of monotone functions $\mathcal{X}$, its lub and sup exist and are the functions $F, G : \mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S})$ such that for every $X \subseteq \mathcal{S}$,

$$F(X) = \bigcap_{H \in \mathcal{X}} H(X); \qquad G(X) = \bigcup_{H \in \mathcal{X}} H(X).$$

  It is easy to verify that F is monotone, that it minorises $\mathcal{X}$, and that it majorises any minoriser of $\mathcal{X}$. Similarly for G.

The function I_{S} is the least fixpoint of the monotone functional $\mathcal{F}$ on $\mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S})$ defined as follows: to every function F, $\mathcal{F}$ associates the function G obtained by feeding F with the argument set X, and then applying one additional instance of inductive rules from S. Since $(\mathcal{P}(\mathcal{S}) \rightsquigarrow \mathcal{P}(\mathcal{S}), \sqsubseteq)$ is a complete lattice, I_{S} is guaranteed to exist.

Formal systems in which *all* rules are either coinductively or inductively interpreted have been studied extensively (see, e.g. [22]). Our constructions, although relatively simple, do not seem to have appeared before at least in this form. The conceptually closest work is the one by Endrullis and coauthors [10].

How could we prove anything about $\mathbb{C}_{\mathsf{S}}$? How should we proceed, as an example, when tyring to prove that a given subset X of $\mathcal{S}$ is included in $\mathbb{C}_{\mathsf{S}}$? Fixed-point theory tells us that the correct way to proceed consists in showing that X is $(I_{\mathsf{S}} \circ C_{\mathsf{S}})$-consistent, namely that $X \subseteq I_{\mathsf{S}}(C_{\mathsf{S}}(X))$. We will frequently apply this proof strategy in the following.

2.2 An Infinitary Linear Lambda Calculus

Preterms are potentially infinite terms built from the following grammar:

$$M, N ::= x \;\Big|\; MM \;\Big|\; \lambda x.M \;\Big|\; \lambda{\downarrow}x.M \;\Big|\; \lambda{\uparrow}x.M \;\Big|\; {\downarrow}M \;\Big|\; {\uparrow}M,$$

where x ranges over a denumerable set $\mathcal{V}$ of variables. $\mathbb{T}$ is the set of preterms. The notion of capture-avoiding substitution of a preterm M for a variable x in anoter preterm N, denoted

$$\frac{}{\downarrow\Theta,\uparrow\Xi,x\vdash x}\ (\mathsf{vl}) \qquad \frac{}{\downarrow\Theta,\uparrow\Xi,\downarrow x\vdash x}\ (\mathsf{vi}) \qquad \frac{}{\downarrow\Theta,\uparrow\Xi,\uparrow x\vdash x}\ (\mathsf{vc}) \qquad \frac{\Gamma,\downarrow\Theta,\uparrow\Xi\vdash M \quad \Delta,\downarrow\Theta,\uparrow\Xi\vdash N}{\Gamma,\Delta,\downarrow\Theta,\uparrow\Xi\vdash MN}\ (\mathsf{a})$$

$$\frac{\Gamma,x\vdash M}{\Gamma\vdash\lambda x.M}\ (\mathsf{ll}) \qquad \frac{\Gamma,\downarrow x\vdash M}{\Gamma\vdash\lambda\downarrow x.M}\ (\mathsf{li}) \qquad \frac{\Gamma,\uparrow x\vdash M}{\Gamma\vdash\lambda\uparrow x.M}\ (\mathsf{lc}) \qquad \frac{\downarrow\Theta,\uparrow\Xi\vdash M}{\downarrow\Theta,\uparrow\Xi\vdash\downarrow M}\ (\mathsf{mi}) \qquad \frac{\downarrow\Theta,\uparrow\Xi\vdash M}{\downarrow\Theta,\uparrow\Xi\vdash\uparrow M}\ (\mathsf{mc})$$

Figure 1: $\ell\Lambda_\infty$: Well-Formation Rules.

$$\dfrac{\dfrac{}{\downarrow y\vdash y}\ (\mathsf{vc}) \quad \dfrac{\pi\triangleright\downarrow y\vdash M}{\downarrow y\vdash\uparrow M}\ (\mathsf{mc})}{\downarrow y\vdash M}\ (\mathsf{a}) \qquad\qquad \dfrac{\rho\triangleright\emptyset\vdash N(\lambda x.x) \quad \dfrac{\dfrac{}{x\vdash x}\ (\mathsf{vl})}{\emptyset\vdash\lambda x.x}\ (\mathsf{ll})}{\emptyset\vdash N(\lambda x.x)}\ (\mathsf{a})$$

Figure 2: $\ell\Lambda_\infty$: Example Derivation Trees π (left) and ρ (right).

$N\{x/M\}$, can be defined, this time by *coinduction*, on the structure of N:

$$\begin{aligned}
(x)\{x/M\} &= M;\\
(y)\{x/M\} &= y;\\
(\lambda x.L)\{y/M\} &= \lambda x.L\{y/M\}; && \text{if } x\notin FV(M)\\
(\lambda\downarrow x.L)\{y/M\} &= \lambda\downarrow x.L\{y/M\}; && \text{if } x\notin FV(M)\\
(\lambda\uparrow x.L)\{y/M\} &= \lambda\uparrow x.L\{y/M\}; && \text{if } x\notin FV(M)\\
(LP)\{y/M\} &= (L\{y/M\})(P\{y/M\});\\
(\downarrow L)\{y/M\} &=\downarrow(L\{y/M\});\\
(\uparrow L)\{y/M\} &=\uparrow(L\{y/M\}).
\end{aligned}$$

Observe that all the equations above are guarded, so this is a well-posed definition. An *inductive* (respectively, *coinductive*) *box* is any preterm in the form $\downarrow M$ (respectively, $\uparrow M$).

Please notice that any (guarded) equation has a unique solution over preterms. As an example, $M=\lambda x.M$, $N=N(\lambda x.x)$, and $M=y\uparrow M$ all have unique solutions. In other words, infinity is everywhere. Only certain preterms, however, will be the objects of this study. To define the class of "good" preterms, simply called *terms*, we now introduce a mixed formal system. An *environment* Γ is simply a set of expressions (called *patterns*) in one the following three forms:

$$p ::= x \ \Big|\ \downarrow x \ \Big|\ \uparrow x,$$

where any variable occurs in at most *one* pattern in Γ. If Γ and Δ are two disjoint environments, then Γ,Δ is their union. An environment is *linear* if it only contains variables. Linear environments are indicated with metavariables like Θ or Ξ. A *term judgment* is an expression in the form $\Gamma\vdash M$ where Γ is an environment and M is a term. A *term* is any preterm M for which a judgment $\Gamma\vdash M$ can be derived by the formal system $\ell\Lambda_\infty$, whose rules are in Figure 1. Please notice that (mc) is coinductive, while all the other rules are inductive. This means that on terms, differently from preterms, not all recursive equations have a solution, anymore. As an example $M=y\uparrow M$ is a term: a derivation π for $\downarrow y\vdash M$ can be found in Figure 2. π is indeed a well-defined derivation because although infinite, any infinite path in it contains infinitely many occurrences of (mc). If we try to proceed in the same way with the preterm $N=N(\lambda x.x)$, we immediately fail: the only candidate derivation ρ looks like the one in Figure 2 and is not well-defined (it contains a "loop" of

inductive rule instances). We write $\mathbb{T}_{\ell\Lambda_\infty}(\Gamma)$ for the set of all preterms M such that $\Gamma \vdash M$. The union of $\mathbb{T}_{\ell\Lambda_\infty}(\Gamma)$ over all environments Γ is denoted simply as $\mathbb{T}_{\ell\Lambda_\infty}$.

Some observations about the rôle of environments are now in order: If $x, \Gamma \vdash M$, then x necessarily occurs free in M, but exactly once and in *linear position* (i.e. not in the scope of any box). If, on the other hand, $\downarrow x, \Gamma \vdash M$, then x can occur free any number of times, even infinitely often, in M. Similarly when $\uparrow x, \Gamma \vdash M$. Observe, in this respect, that inductive and coinductive boxes are actually very permissive: if $\downarrow x, \Gamma \vdash M$, x can even occur in the scope of coinductive boxes, while x can occur in the scope of inductive boxes if $\uparrow x, \Gamma \vdash M$. We claim that this is source of the great expressive power of the calculus, but also of its main defects (e.g. the absence of confluence).

Sometimes it is useful to denote symbols $\downarrow$ and $\uparrow$ in a unified way. To that purpose, let $\mathbb{B}$ be the set $\{0, 1\}$ of binary digits, which is ranged over by metavariables like a or b. $\updownarrow_0$ stands for $\downarrow$, while $\updownarrow_1$ is $\uparrow$. For every $s \in \mathbb{B}^*$, we can define *s-contexts*, ranged over by metavariables like C_s and D_s, as follows, by induction on s:

$$C_\varepsilon ::= [\cdot]; \qquad C_{0\cdot s} ::= \downarrow C_s; \qquad C_{1\cdot s} ::= \uparrow C_s;$$

$$C_s ::= C_s M \mid M C_s \mid \lambda x.C_s \mid \lambda\downarrow x.C_s \mid \lambda\uparrow x.C_s.$$

Given any subset X of $\mathbb{B}^*$, an X-context, sometime denoted as C_X is an s-context where $s \in X$. A *context* C is simply any s-context C_s. For every natural number $n \in \mathbb{N}$, $n^\bullet$ is the set of those strings in $\mathbb{B}^*$ in which 1 occurs precisely n times. For every n, the language $n^\bullet$ is regular.

2.3 Finitary and Infinitary Dynamics

In this section, notions of finitary and infinitary reduction for $\ell\Lambda_\infty$ are given. *Basic reduction* is a binary relation $\mapsto \subseteq \mathbb{T} \times \mathbb{T}$ defined by the following three rules (where, as usual, $M \mapsto N$ stands for $(M, N) \in \mapsto$):

$$(\lambda x.M)N \mapsto M\{x/N\}; \quad (\lambda\downarrow x.M)\downarrow N \mapsto M\{x/N\}; \quad (\lambda\uparrow x.M)\uparrow N \mapsto M\{x/N\}.$$

Basic reduction can be applied in any s-context, giving rise to a *ternary* relation $\rightarrow \subseteq \mathbb{T} \times \mathbb{B}^* \times \mathbb{T}$, simply called *reduction*. That is defined by stipulating that $(M, s, N) \in \rightarrow$ iff there are a s-context C_s and two terms L and P such that $L \mapsto P$, $M = C_s[L]$, and $N = C_s[P]$. In this case, the reduction step is said to occur *at level* s and we write $M \rightarrow_s N$ and $\mathit{level}(M \rightarrow_s N) = s$. We often employ the notation $\rightarrow_X$, i.e., $\rightarrow_X$ is the union of the relations $\rightarrow_s$ for all $s \in X$. If $M \rightarrow_s N$ but we are not interested in the specific s, we simply write $M \rightarrow N$. If $M \rightarrow_{n^\bullet} N$, then reduction is said to occur at depth n.

Given $X \subseteq \mathbb{B}^*$, a *X-normal form* is any term M such that whenever (M, s, N), it holds that $s \notin X$. The set of all X-normal forms is denoted as $\mathsf{NFs}(X)$. In the just introduced notations, the singleton s is often used in place of $\{s\}$ if this does not cause any ambiguity. A *normal form* is simply a $\mathbb{B}^*$-normal form.

Depths and levels have a different nature: while the depth increases only when entering a coinductive box, the level changes while entering any kind of box, and this is the reason why levels are binary strings rather than natural numbers.

Since $M\{x/N\}$ is well-defined whenever M and N are preterms, reduction is certainly closed as a relation on the space of preterms. That it is also closed on terms is not trivial. First of all, substitution lemmas need to be proved for the three kinds of patterns which possibly appear in environments. The first of these lemmas concerns linear variables:

Lemma 1 (Substitution Lemma, Linear Case)**.** *If* $\Gamma, x, \downarrow\Theta, \uparrow\Xi \vdash M$ *and* $\Delta, \downarrow\Theta, \uparrow\Xi \vdash N$*, then it holds that* $\Gamma, \Delta, \downarrow\Theta, \uparrow\Xi \vdash M\{x/N\}$.

Proof. We can prove that the following subset X of judgments is consistent with $\ell\Lambda_\infty$:

$$\left\{\Gamma, \Delta, \downarrow\Theta, \uparrow\Xi \vdash M\{x/N\} \;\middle|\; \Gamma, x, \downarrow\Theta, \uparrow\Xi \vdash M \in \mathbb{C}_{\ell\Lambda_\infty} \wedge \Delta, \downarrow\Theta, \uparrow\Xi \vdash N \in \mathbb{C}_{\ell\Lambda_\infty}\right\} \cup \mathbb{C}_{\ell\Lambda_\infty}.$$

Suppose that $J = \Gamma, \Delta, \downarrow\Theta, \uparrow\Xi \vdash M\{x/N\}$ is in X. If $J \in \mathbb{C}_{\ell\Lambda_\infty}$, then of course

$$J \in I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(\mathbb{C}_{\ell\Lambda_\infty})) \subseteq I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(X)).$$

Otherwise, we know that $H = \Gamma, x, \downarrow\Theta, \uparrow\Xi \vdash M \in \mathbb{C}_{\ell\Lambda_\infty}$ and, by the fact $\mathbb{C}_{\ell\Lambda_\infty} = I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(\mathbb{C}_{\ell\Lambda_\infty}))$ we can infer that $H \in I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(\mathbb{C}_{\ell\Lambda_\infty}))$, namely that H can be obtained by judgments in $\mathbb{C}_{\ell\Lambda_\infty}$ by applying coinductive rules once, followed by n inductive rules. We prove that $J \in \mathbb{C}_{\ell\Lambda_\infty}$ by induction on n:

- If $n = 0$, then H can be obtained by means of the rule (mc), but this is impossible since the environment in any judgment obtained this way cannot contain any variable, and H actually contains one.
- If $n > 0$, then we distinguish a number of cases, depending on the last inductive rule applied to derive H:
  - If it is (vl), then $M = x$, $M\{x/N\}$ is simply N and Γ does not contain any variable. By the fact that
    $$\Delta, \downarrow\Theta, \uparrow\Xi \vdash N \in \mathbb{C}_{\ell\Lambda_\infty}$$
    it follows, by a Weakening Lemma, that $J \in \mathbb{C}_{\ell\Lambda_\infty}$.
  - It cannot be either (vi) or (vc) or (mi): in all these cases the underlying environment cannot contain variables;
  - If it is either (ll) or (a), then the induction hypothesis yields the thesis immediately;
  - If it is either (li) or (lc), then a Weakening Lemma applied to the induction hypothesis leads to the thesis.

From $J \in \mathbb{C}_{\ell\Lambda_\infty}$, it follows that

$$J \in I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(\mathbb{C}_{\ell\Lambda_\infty})) \subseteq I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(X)),$$

which is the thesis. This concludes the proof. □

A similar result can be given when the substituted variable occurs in the scope of an inductive box:

Lemma 2 (Substitution Lemma, Inductive Case)**.** *If $\Gamma, \downarrow x, \downarrow\Theta, \uparrow\Xi \vdash M$ and $\downarrow\Theta, \uparrow\Xi \vdash N$, then it holds that $\Gamma, \downarrow\Theta, \uparrow\Xi \vdash M\{x/N\}$.*

Proof. The structure of this proof is identical to the one of Lemma 1. □

When the variable is in the scope of a coinductive box, almost nothing changes:

Lemma 3 (Substitution Lemma, Coinductive Case)**.** *If $\Gamma, \downarrow x, \downarrow\Theta, \uparrow\Xi \vdash M$ and $\uparrow\Theta, \uparrow\Xi \vdash N$, then it holds that $\Gamma, \downarrow\Theta, \uparrow\Xi \vdash M\{x/N\}$.*

Proof. The structure of this proof is identical to the one of Lemma 1. □

The following is an analogue of the so-called Subject Reduction Theorem, and is an easy consequence of substitution lemmas:

Proposition 1 (Well-Formedness is Preseved by Reduction)**.** *If $\Gamma \vdash M$ and $M \to N$, then $\Gamma \vdash N$.*

Proof. Let us first of all prove that if $M \mapsto N$, and $\Gamma \vdash M$, then $\Gamma \vdash N$. let us distinguish three cases:

- If M is $(\lambda x.L)P$, then $\Delta, x, \downarrow\Theta, \uparrow\Xi \vdash L$ and $\Sigma, \downarrow\Theta, \uparrow\Xi \vdash P$, where $\Gamma = \Delta, \Sigma, \downarrow\Theta, \uparrow\Xi$. By Lemma 1, one gets that $\Gamma \vdash L\{x/P\}$, which is the thesis.
- If M is $(\lambda\downarrow x.L)\downarrow P$, then $\Delta, \downarrow x, \downarrow\Theta, \uparrow\Xi \vdash L$ and $\downarrow\Theta, \uparrow\Xi \vdash P$, where $\Gamma = \Delta, \downarrow\Theta, \uparrow\Xi$. By Lemma 2, one gets that $\Gamma \vdash L\{x/P\}$, which is the thesis.
- If M is $(\lambda\uparrow x.L)\uparrow P$, then $\Delta, \downarrow\Theta, \uparrow x, \uparrow\Xi \vdash L$ and $\downarrow\Theta, \uparrow\Xi \vdash P$, where $\Gamma = \Delta, \downarrow\Theta, \uparrow\Xi$. By Lemma 3, one gets that $\Gamma \vdash L\{x/P\}$, which is the thesis.

One can then prove that for every context C and for every pair of terms M and N such that $M \mapsto N$, if $\Gamma \vdash C[M]$ then $\Gamma \vdash C[N]$. This is an induction on the structure of C. □

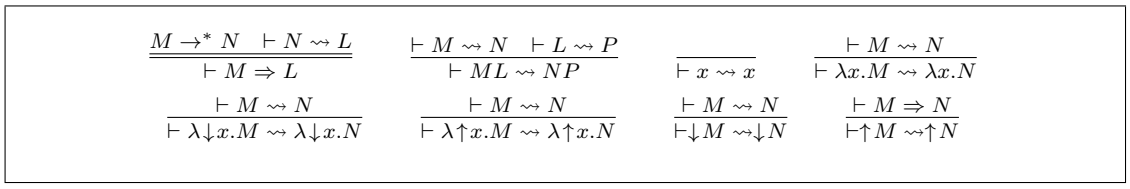

Figure 3: $\ell\Lambda_\infty$: Infinitary Dynamics.

Finitary reduction, as a consequence, is well-defined not only on preterms, but also on terms.

How about *infinite* reduction? Actually, even defining what an infinite reduction sequence *is* requires some care. In this paper, following [10], we define infinitary reduction by way of a mixed formal system (see Section 2). The judgments of this formal system have two forms, namely $\vdash M \Rightarrow N$ and $\vdash M \rightsquigarrow N$, and its rules are in Figure 3. The relation $\Rightarrow$ is the infinitary, coinductively defined, notion of reduction we are looking for. Informally, $\vdash M \Rightarrow N$ is provable (and we write, simply, $M \Rightarrow N$) iff there is a third term L such that M reduces to L in a finite amount of steps, and L itself reduces infinitarily to N where, however, infinitary reduction is applied at depths higher than one. The latter constraint is taken care of by $\rightsquigarrow$.

An infinite reduction sequence, then, can be seen as being decomposed into a finite prefix and finitely many infinite suffixes, each involving subterm occurrences at higher depths. We claim that this corresponds to strongly convergent reduction sequences as defined in [19], although a formal comparison is outside the scope of this paper (see, however, [10]).

What are the main properties of $\Rightarrow$? Is it a confluent notion of reduction? Is it that N is a normal form whenever $M \Rightarrow N$? Actually, the latter question can be easily given a negative answer: take the unique preterm M such that $M = \uparrow(M(II))$, where $I = \lambda x.x$ is the identity combinator. Of course, $\emptyset \vdash M$. We can prove that both $M \Rightarrow N$ and that $M \Rightarrow L$, where

$$N = \uparrow(\uparrow(N(II))I); \qquad L = \uparrow(\uparrow(LI)(II)).$$

(Infact, $\Rightarrow$ is reflexive, see Lemma 22 below.) Neither N nor L is a normal form. It is easy to realise that there is P to which both N and L reduces to, namely $P = \uparrow(\uparrow(PI)I)$. Confluence, however, does not hold in general, as can be easily shown by considering the following two terms M and N:

$$M = K \uparrow N \uparrow K; \qquad N = K \uparrow M \uparrow I;$$

where $K = \lambda\uparrow x.\lambda\uparrow y.x$ and $I = \lambda\uparrow x.x$. If we reduce M at even and at odd depths, we end up at two terms L and P which cannot be joined by $\Rightarrow$, namely the following:

$$L = K \uparrow L \uparrow I; \qquad P = K \uparrow P \uparrow K.$$

The deep reason why this phenomenon happens is an interference between $\rightarrow$ and $\rightsquigarrow$: there are Q and R such that $M \rightarrow Q$ and $M \rightsquigarrow R$, but there is no S such that $Q \rightsquigarrow S$ and $R \rightarrow S$.

2.4 Level-by-Level Reduction

One restriction of $\Rightarrow$ that will be useful in the following is the so called *level-by-level* reduction, which is obtained by constraining reduction to occur at deeper levels only if no redex occurs at outer levels. Formally, let $\rightarrow_{lbl}$ be the restriction of $\rightarrow$ obtained by stipulating that $(M, s, N) \in \rightarrow_{lbl}$ iff there are a s-context C_s and two terms L and P such that $L \mapsto P$, $M = C_s[L]$, and $N = C_s[P]$, *and moreover*, M is t-normal for every prefix t of s. Then one can obtain $\rightsquigarrow_{lbl}$ and $\Rightarrow_{lbl}$ from $\rightarrow_{lbl}$ as we did for $\rightsquigarrow$ and $\Rightarrow$ in Section 2.3 (see Figure 4).

Clearly, if $M \Rightarrow_{lbl} N$, then $M \Rightarrow N$. Moreover, $\Rightarrow_{lbl}$, contrarily to $\Rightarrow$, is confluent, simply because $\rightarrow_{lbl}$ satisfies a diamond-property. This is not surprising, and has been already observed

$$\frac{M \to^*_{lbl} N \quad \vdash N \rightsquigarrow_{lbl} L \quad N \in \mathsf{NFs}(0^\bullet)}{\vdash M \Rightarrow_{lbl} L} \qquad \frac{\vdash M \rightsquigarrow_{lbl} N \quad \vdash L \rightsquigarrow_{lbl} P}{\vdash ML \rightsquigarrow_{lbl} NP} \qquad \frac{}{\vdash x \rightsquigarrow_{lbl} x} \qquad \frac{\vdash M \rightsquigarrow_{lbl} N}{\vdash \lambda x.M \rightsquigarrow_{lbl} \lambda x.N}$$

$$\frac{\vdash M \rightsquigarrow_{lbl} N}{\vdash \lambda{\downarrow}x.M \rightsquigarrow_{lbl} \lambda{\downarrow}x.N} \qquad \frac{\vdash M \rightsquigarrow_{lbl} N}{\vdash \lambda{\uparrow}x.M \rightsquigarrow_{lbl} \lambda{\uparrow}x.N} \qquad \frac{\vdash M \rightsquigarrow_{lbl} N}{\vdash{\downarrow}M \rightsquigarrow_{lbl}{\downarrow}N} \qquad \frac{\vdash M \Rightarrow_{lbl} N}{\vdash{\uparrow}M \rightsquigarrow_{lbl}{\uparrow}N}$$

Figure 4: $\ell\Lambda_\infty$: Level-by-Level Infinitary Dynamics.

$$\frac{}{\Gamma, x \vdash_{abc} x}\ (\mathsf{v}) \qquad \frac{\Gamma \vdash^{\mathbf{F}}_{abc} M \quad \Gamma \vdash^{\mathbf{A}}_{abc} N}{\Gamma \vdash_{abc} MN}\ (\mathsf{a}) \qquad \frac{x, \Gamma \vdash^{\mathbf{L}}_{abc} M}{\Gamma \vdash_{abc} \lambda x.M}\ (\mathsf{l})$$

$$\frac{\Gamma \vdash_{0bc} M}{\Gamma \vdash^{\mathbf{A}}_{0bc} M}\ (\mathsf{ai}) \qquad \frac{\Gamma \vdash_{1bc} M}{\Gamma \vdash^{\mathbf{A}}_{1bc} M}\ (\mathsf{ac}) \qquad \frac{\Gamma \vdash_{a0c} M}{\Gamma \vdash^{\mathbf{F}}_{a0c} M}\ (\mathsf{fi}) \qquad \frac{\Gamma \vdash_{a1c} M}{\Gamma \vdash^{\mathbf{F}}_{a1c} M}\ (\mathsf{fc}) \qquad \frac{\Gamma \vdash_{ab0} M}{\Gamma \vdash^{\mathbf{L}}_{ab0} M}\ (\mathsf{li}) \qquad \frac{\Gamma \vdash_{ab1} M}{\Gamma \vdash^{\mathbf{L}}_{ab1} M}\ (\mathsf{lc})$$

Figure 5: Λ_∞: Well-Formation Rules.

in the realm of finitary rewriting [28]. Moreover, level-by-level is effective: only a finite portion of M needs to be inspected in order to check if a given redex occuring in M can be fired (or to find one if M contains one). Indeed, it will used to define what it means for a term in $\ell\Lambda_\infty$ to compute a function, which is the main topic of the following section.

3 On the Expressive Power of $\ell\Lambda_\infty$

The just introduced calculus $\ell\Lambda_\infty$ can be seen as a refinement of Λ_∞ obtained by giving a first-order status to depths, i.e., by introducing a specific construct which makes the depth to increase when crossing it. In this section, we will give an interesting result about the absolute expressive power of the introduced calculus: not only functions on finite strings can be expressed, but also functions on *infinite* strings. Before doing that, we will investigate on the possibility to embed existing infinitary λ-calculi from the literature.

3.1 Embedding Λ_∞

Some introductory words about Λ_∞ are now in order (see [19] or [17] for more details). Originally Λ_∞ has been defined based on completing the space of λ-terms with respect to a metric. Here we reformulate the calculus differently, based on coinduction.

In Λ_∞, there are many choices as to where the underlying depth can increase. Indeed, *eight* different calculi can be defined. More specifically, for every $a, b, c \in \mathbb{B}$, Λ_{abc} is obtained by stipulating that:

- the depth increases while crossing abstractions iff $a = 1$;
- the depth increases when going through the first argument of an application iff $b = 1$;
- the depth increases when entering the second argument of an application iff $c = 1$.

Formally, one can define terms of Λ_{abc} as those (finite or infinite) λ-terms M such that $\Gamma \vdash_{abc} M$ is derivable through the rules in Figure 5. Finite and infinite reduction sequences can be defined exactly as we have just done for $\ell\Lambda_\infty$. The obtained calculi have a very rich and elegant mathematical theory. Not much is known, however, about whether Λ_∞ can be tailored as to guarantee key properties of programs working on streams, like productivity.

Let us now show how Λ_{000} and Λ_{001} can indeed be embedded into $\ell\Lambda_\infty$. For every binary digit

a, the map $\langle\cdot\rangle_a$ from the space of terms of Λ_{00a} into the space of preterms is defined as follows:

$$\begin{aligned}\langle x\rangle_a &= x;\\ \langle MN\rangle_a &= \langle M\rangle_a \updownarrow_a \langle N\rangle_a;\\ \langle \lambda x.M\rangle_a &= \lambda \updownarrow_a x.\langle M\rangle_a;\end{aligned}$$

where the expression $\updownarrow_a M$ is defined to be $\downarrow M$ if $a = 0$ and $\uparrow M$ if $a = 1$. Please observe that $\langle\cdot\rangle_a$ is defined by coinduction on the space of terms of Λ_{00a}, which contains possibly infinite objects. By the way, $\langle\cdot\rangle_a$ can be seen as Girard's embedding of intuitionistic logic into linear logic where, however, the kind of boxes and abstractions we use depends on a: we go inductive if $a = 0$ and coinductive otherwise.

First of all, preterms obtained via the embedding are actuall(depending on a) in the environment:

Lemma 4. *For every $M \in \Lambda_{00a}$, it holds that $\updownarrow_a FV(M) \vdash \langle M\rangle_a$.*

Proof. We proceed by showing that the following set of judgments X is consistent with $\ell\Lambda_\infty$:

$$\left\{\updownarrow_a FV(M) \vdash \langle M\rangle_a \;\middle|\; M \in \Lambda_{00a}\right\}.$$

Suppose that $\updownarrow_a FV(M) \vdash \langle M\rangle_a$, where $M \in \Lambda_{00a}$. This implies that $M = G[N_1, \ldots, N_n]$, where $N_1, \ldots, N_n \in \Lambda_{00a}$. The fact that $M \in I_{\ell\Lambda_\infty}(C_{\ell\Lambda_\infty}(X))$ can be proved by induction on G, with different cases depending on the value of a. □

From a dynamical point of view, this embedding is *perfect*: not only basic reduction in Λ_∞ can be simulated in $\ell\Lambda_\infty$, but any reduction we do in $\langle M\rangle_a$ can be traced back to a reduction happening in M.

Lemma 5 (Perfect Simulation)**.** *For every $M \in \Lambda_{00a}$, if $M \to_n N$, then $\langle M\rangle_a \to_{n\bullet} \langle N\rangle_a$. Moreover, for every $M \in \Lambda_{00a}$, if $\langle M\rangle_a \to_{n\bullet} N$, then there is L such that $M \to_n L$ and $\langle L\rangle_a \equiv N$.*

Proof. Just consider how a redex $(\lambda x.M)N$ in Λ_{00a} is translated: it becomes $(\lambda \updownarrow_a x.\langle M\rangle_a) \updownarrow_a \langle N\rangle_a$. As can be easily proved, for every a and for every $M, N \in \Lambda_{00a}$,

$$(\langle M\rangle_a)\{x/\langle N\rangle_a\} = \langle M\{x/N\}\rangle_a.$$

This means $\ell\Lambda_\infty$ correctly simulates Λ_{00a}. For the converse, just observe that the only redexes in $\langle M\rangle_a$ are those corresponding to redexes from M. □

One may wonder whether Λ_{001} is the only (non-degenerate) dialect of Λ_∞ which can be simulated in $\ell\Lambda_\infty$. Actually, besides the perfect embedding we have just given there is also an *imperfect* embedding of systems in the form Λ_{a0b} (where a and b are binary digits) into $\ell\Lambda_\infty$:

$$\begin{aligned}\langle x\rangle_{ab} &= x;\\ \langle MN\rangle_{ab} &= (\lambda \updownarrow_a x.x)(\langle M\rangle_{ab} \updownarrow_b \langle N\rangle_{ab});\\ \langle \lambda x.M\rangle_{ab} &= \lambda \updownarrow_b x.\, \updownarrow_b \langle M\rangle_{ab}.\end{aligned}$$

This is a variation on the so-called call-by-value embedding of intuitionistic logic into linear logic (i.e. the embedding induced by the map $(A \to B)^\bullet = !(A^\bullet) \multimap !(B^\bullet)$, see [23]). Please notice, however, that variables occur nonlinearly in the environment, while the term itself is never a box, contrarily to the usual call-by-value embedding (where at least values are translated into boxes). As expected:

Lemma 6. *For every $M \in \Lambda_{a0b}$, it holds that $\updownarrow_b FV(M) \vdash \langle M\rangle_{ab}$.*

As can be easily realised, any β step in Λ_∞ can be simulated by *two* reduction steps in $\ell\Lambda_\infty$. This makes the simulation imperfect:

Lemma 7 (Imperfect Simulation)**.** *For every $M \in \Lambda_{a0b}$, if $M \to_n N$, then $\langle M\rangle_{ab} \to^2_{n\bullet} \langle N\rangle_{ab}$.*

Lemma 8. *Moreover, for every $M \in \Lambda_{a0b}$, if $\langle M\rangle_{ab} \to_{n\bullet} N$, then there is L such that $M \to_n L$ and $\langle L\rangle_a \equiv N$.*

3.2 $\ell\Lambda_\infty$ as a Stream Programming Language

One of the challenges which lead to the introduction of infinitary rewriting systems is, as we argued in the Introduction, the possibility to inject infinity (as arising in lazy data structures such as streams) into formalisms like the λ-calculus. In this section, we show that, indeed, $\ell\Lambda_\infty$ can not only express terms of any free (co)algebras, but also any effective function on them. Moreover, anything $\ell\Lambda_\infty$ can compute can also be computed by Type-2 Turing machines. To the author's knowledge, this is the first time this is done for any system of infinitary rewriting (some partial results, however, can be found in [5]).

3.3 Signatures and Free (Co)algebras

A *signature* Φ is a set of function symbols, each with an associated *arity*. Function symbols will be denoted with metavariables like $\mathtt{f}$ or $\mathtt{g}$. In this paper, we are concerned with *finite* signatures, only. Sometimes, a signature has a very simple structure: an *alphabet signature* is a signature whose function symbols all have arity 1, except for a single nullary symbol, denoted ε. Given an alphabet Σ, Σ^* (Σ^ω, respectively) denotes the set of finite (infinite, respectively) words over Σ. Σ^∞ is simply $\Sigma^* \cup \Sigma^\omega$. For every alphabet Σ, there is a corresponding alphabet signature Φ_Σ. Given a signature (or an alphabet), one usually needs to define the set of terms built according to the algebra itself. Indeed, the *free algebra* $\mathbb{FA}(\Phi)$ induced by a signature Φ is the set of all finite terms built from function symbols in Φ, i.e., all terms *inductively* defined from the following production (where n is the arity of $\mathtt{f}$):

$$t ::= \mathtt{f}(t_1, \ldots, t_n). \tag{1}$$

There is another canonical way of building terms from signatures, however. One can interpret the production above *coinductively*, getting a space of finite *and infinite* terms: the *free coalgebra* $\mathbb{FC}(\Phi)$ induced by a signature Φ is the set of all finite *and infinite* terms built from function symbols in Φ, following (1). Notice that Σ^* is isomorphic to $\mathbb{FA}(\Phi_\Sigma)$, while Σ^∞ is isomorphic to $\mathbb{FC}(\Phi_\Sigma)$. We often elide the underlying isomorphisms, confusing strings and terms.

3.4 Representing (Infinitary) Terms in $\ell\Lambda_\infty$

. There are many number systems which work well in the finitary λ-calculus. One of them is the well-known system of Church numerals, in which $n \in \mathbb{N}$ is represented by $\lambda x.\lambda y.x^n y$. We here adopt another scheme, attributed to Scott [30]: this allows to make the relation between depths and computation more explicit. Let $\Phi = \{\mathtt{f}_1, \ldots, \mathtt{f}_n\}$ and suppose that symbols in Φ can be totally ordered in such a way that $\mathtt{f}_n \leq \mathtt{f}_m$ iff $n \leq m$. Terms of the free algebra $\mathbb{FA}(\Phi)$ can be encoded as terms of $\ell\Lambda_\infty$ as follows

$$\langle \mathtt{f}_m(t_1, \ldots, t_p)\rangle^{\mathbb{A}}_\Phi = \lambda\!\downarrow\! x_1.\cdots.\lambda\!\downarrow\! x_n.x_m \downarrow\langle t_1\rangle^{\mathbb{A}}_\Phi \cdots \downarrow\langle t_p\rangle^{\mathbb{A}}_\Phi.$$

Similarly for terms in the free coalgebra $\mathbb{FC}(\Phi)$:

$$\langle \mathtt{f}_m(t_1, \ldots, t_p)\rangle^{\mathbb{C}}_\Phi = \lambda\!\downarrow\! x_1.\cdots.\lambda\!\downarrow\! x_n.x_m \uparrow\langle t_1\rangle^{\mathbb{C}}_\Phi \cdots \uparrow\langle t_p\rangle^{\mathbb{C}}_\Phi.$$

Given a string $s \in \Sigma^*$, the term $\langle s\rangle^{\mathbb{A}}_{\Phi_\Sigma}$ is denoted simply as $\langle s\rangle_*$. Similarly, if $s \in \Sigma^\omega$, $\langle s\rangle_\omega$ indicates $\langle s\rangle^{\mathbb{C}}_{\Phi_\Sigma}$. Please observe how $\langle\cdot\rangle^{\mathbb{A}}_\Phi$ differs from $\langle\cdot\rangle^{\mathbb{C}}_\Phi$: in the first case the encoding of subterms are wrapped in an inductive box, while in the second case the enclosing box is coinductive. This very much reflects the spirit of our calculus: in $\langle t\rangle^{\mathbb{C}}_\Phi$ the depth increases whenever entering a subterm, while in $\langle s\rangle^{\mathbb{A}}_\Phi$, the depth *never* increases.

3.5 Universality

The question now is: given the encoding in the last paragraph, which *functions* can we represent in $\ell\Lambda_\infty$? If domain and codomain are *free algebras*, a satisfactory answer easily comes from the universality of ordinary λ-calculus with respect to computability on finite structures: the

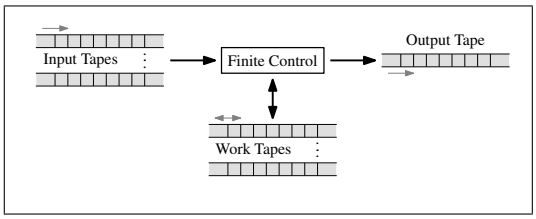

Figure 6: The Structure of a Type-2 Turing Machine

class of functions at hand coincides with the effectively computable ones. If, on the other hand, functions handling or returning terms from free coalgebras are of interest, the question is much more interesting.

The expressive power of $\ell\Lambda_\infty$ actually coincides with the one of Type-2 Turing machines: these are mild generalisations of ordinary Turing machines obtained by allowing inputs and outputs to be not-necessarily-finite strings. Such a machine consists of finitely many, initially blank work tapes, finitely many one-way input tapes and a single one-way output tape. Noticeably, input tapes initially contain not-necessarily-finite strings, while the output tape is sometime supposed to be filled with an infinite string. See [31] for more details and Figure 6 for a graphical representation of the structure of any Type-2 Turing machine: black arrows represent the data flow, whereas grey arrows represent the possible direction of the head in the various tapes.

We now need to properly formalise *when* a given function on possibly infinite strings can be represented by a term in $\ell\Lambda_\infty$. To that purpose, let $\mathbb{S}$ be the set $\{*, \omega\}$, where the two elements of $\mathbb{S}$ are considered merely as *symbols*, with no internal structure. Objects in $\mathbb{S}$ are indicated with metavariables like $\mathfrak{a}$ or $\mathfrak{b}$. A partial function f from $\Sigma^{\mathfrak{a}_1} \times \cdots \times \Sigma^{\mathfrak{a}_n}$ to $\Sigma^{\mathfrak{b}}$ is said to be *representable* in $\ell\Lambda_\infty$ iff there is a finite term M_f such that for every $s_1 \in \Sigma^{\mathfrak{a}_1}, \ldots, s_n \in \Sigma^{\mathfrak{a}_n}$ it holds that

$$M_f\langle s_1\rangle_{\mathfrak{a}_1} \cdots \langle s_n\rangle_{\mathfrak{a}_n} \Rightarrow_{lbl} \langle f(s_1, \ldots, s_n)\rangle_{\mathfrak{b}}$$

if $f(s_1, \ldots, s_n)$ is defined, while $M_f\langle s_1\rangle_{\mathfrak{a}_1} \cdots \langle s_n\rangle_{\mathfrak{a}_n}$ has no normal form otherwise. Notice the use of level-by-level reduction. Noticeably:

Theorem 1 (Universality)**.** *The class of functions which are representable in $\ell\Lambda_\infty$ coincides with the class of functions computable by Type-2 Turing machines.*

Proof. This proof relies on a standard encoding of Turing machines into $\ell\Lambda_\infty$. Rather than describing the encoding in detail, we now give some observations, that together should convince the reader that the encoding is indeed possible:

- First of all, inductive and coinductive fixed point combinators are both available in $\ell\Lambda_\infty$. Indeed, let M_a be the following term:

  $$M_a = \lambda\downarrow x.\lambda\downarrow y.y\updownarrow_a((x\downarrow x)\downarrow y)$$

  Then Y_a is just $M_a\downarrow M_a$. Observe that $Y_a\downarrow M \Rightarrow M\updownarrow_a(Y_a\downarrow M)$.
- Moreover, observe that the encoding of free (co)algebras described above not only provides an elegant way to represent *terms*, but also allows to very easily define efficient combinators for selection. For example, given the alphabet $\Sigma = \{0, 1\}$, the algebra $\mathbb{FA}(\Sigma)$ of binary strings corresponds to the term

  $$M = \lambda x.\lambda\downarrow y_0.\lambda\downarrow y_1.\lambda\downarrow y_\varepsilon.x\downarrow y_0\downarrow y_1\downarrow y_\varepsilon.$$

  Please observe that

  $$M\langle 0(s)\rangle^{\mathbb{A}}_\Sigma\downarrow N_0\downarrow N_1\downarrow N_\varepsilon \Rightarrow N_0\langle s\rangle^{\mathbb{A}}_\Sigma;$$
  $$M\langle 1(s)\rangle^{\mathbb{A}}_\Sigma\downarrow N_0\downarrow N_1\downarrow N_\varepsilon \Rightarrow N_1\langle s\rangle^{\mathbb{A}}_\Sigma;$$
  $$M\langle \varepsilon\rangle^{\mathbb{A}}_\Sigma\downarrow N_0\downarrow N_1\downarrow N_\varepsilon \Rightarrow N_\varepsilon\langle s\rangle^{\mathbb{A}}_\Sigma.$$

This can be generalised to an arbitrary (co)algebra.

- Tuples can be represented easily as follows:

$$\lambda x.x \downarrow M_1 \ldots \downarrow M_n.$$

- A configuration of a Type-2 Turing machine working on the alphabet Σ, with states in the set Q, and having n input tapes and m working tapes can be seen as the following $n+3m+1$-tuple:

$$(s_1, \ldots, s_n, t_1^l, a_1, t_1^r, \ldots, t_m^l, a_m, t_m^r, q)$$

  where s_i is the not-yet-read portion of the i-th input tape, t_i^l (respectively, t_i^r) is the portion of the i-th working tape to the left (respectively, right) of the head, a_i is the symbol currently under the i-th head, and q is the current state. All these $n + 3m + 1$ objects can be seen as elements of appropriate (co)algebras:
  - $s_1, \ldots, s_n$ are (finite or infinite, depending on the underlying machine) strings;
  - $a_1, \ldots, a_m, q$ are all elements of finite sets;
  - $t_1^l, t_1^r, \ldots, t_m^l, t_m^r$ are finite strings;

  As a consequence, all of them can be encoded in Scott-style. Moreover, the availability of selectors and tuples makes it easy to write a term encoding the transition function of the encoded machine. Please notice that the output tape is *not* part of the configuration above. If a character is produced in output (i.e. whenever the head of the output tape moves right), the rest of the computation will take place "inside" the encoding of a (possibly infinite) string.
- One needs to be careful when handling final states: if the machine reaches a final state *even if* it is meant to produce infinite strings in output, the encoding λ-term should anyway diverge [31].

Putting the ingredients above together, one gets for every machine $\mathcal{M}$ a term $M_{\mathcal{M}}$ which computes the same function as $\mathcal{M}$. The fact that anything computable by a finite term M is also computable by a Type-2 Turing machine is quite easy, since level-by-level reduction is effective and, moreover, a normal form in the sense of level-by-level reduction is reached iff it is reached applying surface reduction (in the sense of inductive boxes) *at each depth*. □

4 Taming Infinity: $\ell\Lambda_\infty^{\mathsf{4S}}$

As we have seen in the last sections, $\ell\Lambda_\infty$ is a very powerful model: not only it is universal as a way to compute over streams, but also comes with an extremely liberal infinitary dynamics for which, unfortunately, confluence does not hold. If this paper finished here, this work would then be rather inconclusive: $\ell\Lambda_\infty$ suffers from the same kind of defects which its nonlinear sibling Λ_∞ has.

In this section, however, we will define a restriction of $\ell\Lambda_\infty$, called $\ell\Lambda_\infty^{\mathsf{4S}}$, which thanks to a careful management of boxes in the style of light logics [15, 8, 21], allows to keep infinity under control, and to get results which are impossible to achieve in Λ_∞.

Actually, the notion of a preterm remains unaltered. What changes is how *terms* are defined. First of all, patterns are generalised by two new productions $p ::= \#x \mid \updownarrow x$, which make the notion of an environment slightly more general: it can now contain variables in *five* different forms. Judgments have the usual shape, namely $\Gamma \vdash M$ where Γ is an environment and M is a term. Metavariables like Υ or Π stand for environments where the only allowed patterns are either variables or the ones in the form $\downarrow x$. Rules of $\ell\Lambda_\infty^{\mathsf{4S}}$ are quite different than the ones of $\ell\Lambda_\infty$, and can be found in Figure 7. The meaning well-formation rules induce on variable occurring in environments is more complicated than for $\ell\Lambda_\infty$. Suppose that $\Gamma \vdash M$. Then:

1. If $x \in \Gamma$ then, as usual, x occurs once in M, and outside of any box;
2. If $\#x \in \Gamma$ then x can occur any number of times in M, but all these occurrences are in *linear* position, i.e., outside the scope of *any* box;
3. If $\downarrow x \in \Gamma$ then x occurs exactly once in M, and in the scope of exactly one (inductive) box;
4. If $\uparrow x \in \Gamma$, then x occurs any number of times in M, with the only proviso that any such occurrence of x must be in the scope of *at least* one coinductive box.

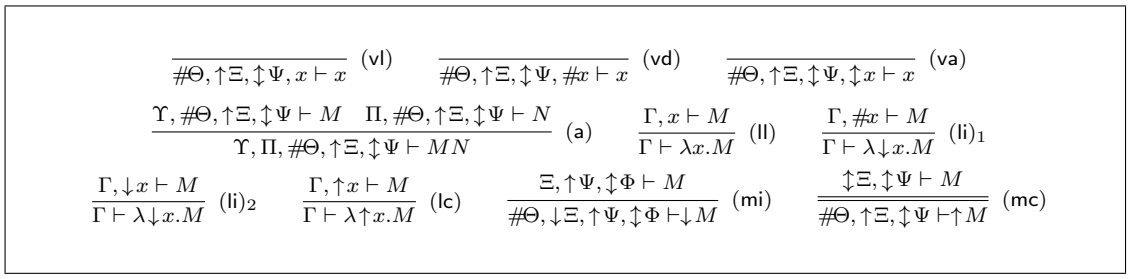

Figure 7: $\ell\Lambda^{\mathsf{4S}}_{\infty}$: Well-Formation Rules.

5. Finally, if $\updownarrow x \in \Gamma$, then x occurs any number of times in M, in any possible position.

Conditions 1. to 3. are reminiscent of the ones of Lafont's soft linear logic. Analogously, Condition 4. is very much in the style of 4LL as described by Danos and Joinet [8]: $\uparrow$ is morally a functor for which contraction, weakening and digging are available, but which does not support dereliction. We will come back to the consequences of the exponential discipline below in this section. Please observe that any variable x marked as $\#x$ or $\downarrow x$ cannot occur in the scope of coinductive boxes. The pattern $\updownarrow x$ has only a merely technical role.

If Γ and Δ are environments, we write $\Gamma \prec \Delta$ iff Δ can be obtained from Γ by replacing *some* patterns in the form $\downarrow x$ with $\#x$. Well-formation is preserved by reduction and, as for $\ell\Lambda_{\infty}$, a proof of this fact requires a number of substitution lemmas:

Lemma 9 (Substitution Lemma, Linear Case)**.** *If* $\Upsilon, x, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash M$ *and* $\Pi, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash N$*, then* $\Upsilon, \Pi, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash M\{x/N\}$*.*

Lemma 10 (Substitution Lemma, First Inductive Case)**.** *If* $\Upsilon, \#\Theta, \#x, \uparrow\Xi, \updownarrow\Psi \vdash M$ *and* $\Phi, \uparrow\Xi, \updownarrow\Psi \vdash N$*, then* $\Upsilon, \#\Theta, \#\Phi, \uparrow\Xi, \updownarrow\Psi \vdash M\{x/N\}$*.*

Lemma 11 (Substitution Lemma, Second Inductive Case)**.** *If* $\Upsilon, \#\Theta, \downarrow x, \uparrow \Xi, \updownarrow \Psi \vdash M$ *and* $\Phi, \uparrow\Xi, \updownarrow\Psi \vdash N$*, then* $\Upsilon, \#\Theta, \downarrow\Phi, \uparrow\Xi, \updownarrow\Psi \vdash M\{x/N\}$*.*

Lemma 12 (Substitution Lemma, Coinductive Case)**.** *If* $\Upsilon, \#\Theta, \uparrow\Xi, \uparrow x, \updownarrow\Psi \vdash M$ *and* $\updownarrow\Xi, \updownarrow\Psi \vdash N$*, then* $\Upsilon, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash M\{x/N\}$*.*

Lemma 13 (Substitution Lemma, Arbitrary Case)**.** *If* $\Upsilon, \#\Theta, \uparrow\Xi, \updownarrow\Psi, \updownarrow x \vdash M$ *and* $\updownarrow\Psi \vdash N$*, then* $\Upsilon, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash M\{x/N\}$*.*

Altogether, the lemmas above imply

Proposition 2 (Well-Formedness is Preseved by Reduction)**.** *If* $\Gamma \vdash M$ *and* $M \to N$*, then* $\Delta \vdash N$ *where* $\Gamma \prec \Delta$*.*

Please observe that the underlying environment can indeed change during reduction, but in a very peculiar way: variables occurring in a $\downarrow$-pattern can later move to a $\#$-pattern.

Classes $\mathbb{T}_{\ell\Lambda^{\mathsf{4S}}_{\infty}}$ and $\mathbb{T}_{\ell\Lambda^{\mathsf{4S}}_{\infty}}(\Gamma)$ (where Γ is an environment) are defined in the natural way, as in $\ell\Lambda_{\infty}$.

4.1 The Fundamental Lemma

It is now time to show *why* $\ell\Lambda^{\mathsf{4S}}_{\infty}$ is a computationally well-behaved object. In this section we will prove a crucial result, namely that reduction is strongly normalising *at each depth*. Before embarking on the proof of this result, let us spend some time to understand why this is the case, giving some necessary definitions along the way.

For any term $M \in \mathbb{T}_{\ell\Lambda^{\mathsf{4S}}_{\infty}}$ let us define the *size* $||M||_n$ of M *at depth* n as the number of occurrences of any symbol at depth n inside M. Observe that $||M||_n$ is well-defined *only* because

$$
\begin{aligned}
||x||_0 &= 1;\\
||\downarrow M||_0 &= ||M||_0 + 1;\\
||\uparrow M||_0 &= 0;\\
||MN||_0 &= ||M||_0 + ||N||_0 + 1;\\
||\lambda x.M||_0 &= ||\lambda\downarrow x.M||_0\\
&= ||\lambda\uparrow x.M||_0 = ||M||_0 + 1;\\
\\
||x||_{m+1} &= 0;\\
||\downarrow M||_{m+1} &= ||M||_{m+1};\\
||\uparrow M||_{m+1} &= ||M||_m;\\
||MN||_{m+1} &= ||M||_{m+1} + ||N||_{m+1};\\
||\lambda x.M||_{m+1} &= ||\lambda\downarrow x.M||_{m+1}\\
&= ||\lambda\uparrow x.M||_{m+1} = ||M||_{m+1}.
\end{aligned}
$$

Figure 8: Parametrised Sizes of Preterms: Equations.

M is assumed to be a term and not just a preterm. Formally, $||M||_n$ is any natural number satisfying the equations in Figure 8, and the following result holds:

Lemma 14. *For every term M and for every natural number $m \in \mathbb{N}$ there is a unique natural number n such that $||M||_m = n$.*

Proof. The fact that for each M and for each m there is *one* natural number satisfying the equations in Figure 8 can be proved by induction on m:

- If $m = 0$, then since M is a term, $\Gamma \vdash M$ is an element of the set $I_{\ell\Lambda^{4S}_\infty}(C_{\ell\Lambda^{4S}_\infty}(\mathbb{C}_{\ell\Lambda^{4S}_\infty}))$. Then, let us perform another induction on the (finite) number of inductive rules used to obtain $\Gamma \vdash M$ from something in $C_{\ell\Lambda^{4S}_\infty}(\mathbb{C}_{\ell\Lambda^{4S}_\infty})$:
  - If the last rule is (vl), (vd) or (va), then $||M||_0 = 1$ by definition;
  - If the last rule is (a), then $M = NL$, there are $||N||_0$ and $||L||_0$, and $||M||_0 = ||N||_0 + ||L||_0 + 1$;
  - If the last rule is either (ll), $(\mathsf{li})_1$, $(\mathsf{li})_2$, (lc) or (mi), then we can proceed like in the previous case, by the inductive hypothesis;
  - If the last rule is (mc), then $||M||_0 = 0$ by definition.
- If $m \geq 1$, then again, since M is a term, $\Gamma \vdash M$ is an element of the set $I_{\ell\Lambda^{4S}_\infty}(C_{\ell\Lambda^{4S}_\infty}(\mathbb{C}_{\ell\Lambda^{4S}_\infty}))$. Then, let us perform an induction on the (finite) number of inductive rules used to get $\Gamma \vdash M$ from something in $C_{\ell\Lambda^{4S}_\infty}(\mathbb{C}_{\ell\Lambda^{4S}_\infty})$. The only interesting case is $M = \uparrow N$, since in all the other cases we can proceed exactly as in the case $m = 0$. From the fact that $\Gamma \vdash M \in I_{\ell\Lambda^{4S}_\infty}(C_{\ell\Lambda^{4S}_\infty}(\mathbb{C}_{\ell\Lambda^{4S}_\infty}))$, it follows that there is Δ such that $\Delta \vdash N \in \mathbb{C}_{\ell\Lambda^{4S}_\infty}$, i.e. N itself is a term. But then we can apply the inductive hypothesis and obtain that $||N||_{m-1}$ exists. It is now clear that $||M||_m = ||N||_{m-1}$ exists.

As for uniqueness, it can be proved by observing that the equations from figure 8 can be oriented so as to get a confluent rewrite system for which, then, the Church-Rosser property holds. This concludes the proof. □

Now, suppose that a term M is such that $M \to_{n\bullet} P$. The term M, then, must be in the form $C_{n\bullet}[N]$ where N is a redex whose reduct is L, and P is just $C_{n\bullet}[L]$. The question is: how does any $||C_{n\bullet}[N]||_m$ relate to the corresponding $||C_{n\bullet}[L]||_m$? Some interesting observations follow:

- If $m < n$ then $||C_{n\bullet}[L]||_m$ equals $||C_{n\bullet}[N]||_m$, since reduction does not affect the size at lower levels;

$$
\begin{aligned}
W_0^n(x) &= 1;\\
W_0^n(\downarrow M) &= n \cdot W_0^n(M);\\
W_0^n(\uparrow M) &= 0;\\
W_0^n(MN) &= W_0^n(M) + W_0^n(N);\\
W_0^n(\lambda x.M) &= W_0^n(\lambda\downarrow x.M)\\
&= W_0^n(\lambda\uparrow x.M) = W_0^n(M) + 1;
\end{aligned}
$$

$$
\begin{aligned}
W_{m+1}^n(x) &= 0;\\
W_{m+1}^n(\downarrow M) &= W_{m+1}^n(M);\\
W_{m+1}^n(\uparrow M) &= W_m^n(M);\\
W_{m+1}^n(MN) &= W_{m+1}^n(M) + W_{m+1}^n(N);\\
W_{m+1}^n(\lambda x.M) &= W_{m+1}^n(\lambda\downarrow x.M)\\
&= W_{m+1}^n(\lambda\uparrow x.M) = W_{m+1}^n(M).
\end{aligned}
$$

Figure 9: Parametrised Weights of Preterms: Equations.

- If $m > n$ then of course $||C_{n\bullet}[L]||_m$ can be much bigger than $||C_{n\bullet}[N]||_m$, simply because symbol occurrences at depth m can be duplicated as an effect of substitution;
- Finally, if $m = n$, then $p = ||C_{n\bullet}[L]||_m$ can again be bigger than $r = ||C_{n\bullet}[N]||_m$, but in a very controlled way. More specifically,
  - if N is a *linear* redex, then $r < p$ because the function body has exactly one free occurrence of the bound variable;
  - if N is an *inductive* redex, then r can indeed by bigger than p, but in that case the involved inductive box has disappeared.
  - if N is a *coinductive* redex, then $r < p$ because the involved coinductive box can actually be copied many times, but all the various copies will be found at depths strictly bigger than $n = m$.

The informal argument above can be formalised by way of an appropriate notion of weight, generalising the argument by Lafont [21] to the more general setting we work in here.

Given $n, m \in \mathbb{N}$ and a term M, the *n-weight* $W_m^n(M)$ of M at depth m is any natural number satisfying the rules from Figure 9.

Lemma 15. *For every term M and for every natural numbers $n, m \in \mathbb{N}$, there is a unique natural number p such that $W_m^n(M) = p$.*

Proof. This can be proved to hold in exactly the same way as we did for the size in Lemma 14. □

Similarly, one can define the duplicability factor of M at depth m, $D_m(M)$: take the rules in Figure 10 and prove they uniquely define a natural number for every term, in the same way as we have just done for the weight ($NFO(x, M)$ is the number of free occurrences of x in the term M, itself a well-defined concept when M is a term). Given a term M, the weight of M at depth n is simply $W_n(M) = W_n^{D_n(M)}(M)$.

The calculus $\ell\Lambda^{\mathsf{4S}}_\infty$ is designed in such a way that the duplicability factor never increases:

Lemma 16. *If $M \in \mathbb{T}_{\ell\Lambda^{\mathsf{4S}}_\infty}$ and $M \to N$, then $D_m(M) \geq D_m(N)$ for every m.*

Proof. A formal proof could be given. We prefer, however, to give a more intuitive one here. Observe that:

$$
\begin{aligned}
D_0(x) &= 1;\\
D_0(\downarrow M) &= D_0(M);\\
D_0(\uparrow M) &= 1;\\
D_0(MN) &= \max\{D_0(M), D_0(N)\};\\
D_0(\lambda x.M) &= D_0(\lambda\uparrow x.M) = D_0(M);\\
D_0(\lambda\downarrow x.M) &= \max\{NFO(x, M), D_0(M)\}.
\end{aligned}
$$

$$
\begin{aligned}
D_{m+1}(x) &= 1;\\
D_{m+1}(\downarrow M) &= D_{m+1}(M);\\
D_{m+1}(\uparrow M) &= D_m(M);\\
D_{m+1}(MN) &= \max\{D_{m+1}(M), D_{m+1}(N)\};\\
D_{m+1}(\lambda x.M) &= D_{m+1}(\lambda\downarrow x.M)\\
&= D_{m+1}(\lambda\uparrow x.M) = D_{m+1}(M).
\end{aligned}
$$

Figure 10: Parametrised Duplicability Factor of Preterms: Equations.

- If $\Gamma, x \vdash M$ or $\Gamma, \downarrow x \vdash M$, then the variable x occurs free exactly once in M, in the first case outside the scope of any box, in the second case in the scope of exactly one inductive box.
- If $\Gamma, \#x \vdash M$, then x occurs free more than once in M, all the occurrences being outside the scope of any box.
- The duplicability factor at level n of M is nothing more than the maximum, over all abstractions $\lambda\downarrow x.N$ at level n in M, of the number of free occurrences of x in N. Observe that by the well-formation rules in Figure 7, the variable x must be marked as $\#x$ or as $\downarrow x$ for any such N. If it is marked as $\downarrow x$, however, it occurs once in N.
- Now, consider the substitution lemmas 9, 10, 11, 12, and 13. In all the five cases, one realises that:
  1. for every n, $D_n(M\{x/N\}) \leq \max\{D_n(M), D_n(N)\}$, because every abstraction occurring in $M\{x/N\}$ also occurs in either M or N, and substitution is capture-avoiding.
  2. If in the judgment $\Gamma \vdash M\{x/N\}$ one gets as a result of the substitution lemma there is $\#y \in \Gamma$, then $NFO(y, M\{x/N\})$ cannot be too big: there must be some z such that z is marked as $\#z$ in one (or both) of the provable judgments existing by hypothesis, but $NFO(z, M) + NFO(z, N)$ majorises $NFO(y, M\{x/N\})$. Why? Simply because the only case in which $NFO(y, M\{x/N\})$ could potentially grow bigger is the one in which x is marked as $\#x$ in the judgment for M. In that case, however, the variables which are free in N are all linear (or marked as $\uparrow z$ or $\updownarrow z$).

This concludes the proof. □

Moreover, and this is the crucial point, $W_n(M)$ is guaranteed to strictly decrease whenever $M \to_{n\bullet} N$:

Lemma 17. *Suppose that $M \in \mathbb{T}_{\ell\Lambda^{\mathsf{as}}_{\infty}}$ and that $M \to_{n\bullet} N$. Then $W_n(M) > W_n(N)$. Moreover, $W_m(M) = W_m(N)$ whenever $m < n$.*

Proof. We first of all need to prove the following variations on the substitution lemmas:

1. If $\Upsilon, x, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash M$ and $\Pi, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash N$, then for every $n \geq \max\{D_0(M), D_0(N)\}$ it holds that $W^n_0(M\{x/N\}) \leq W^n_0(M) + W^n_0(N)$.
2. If $\Upsilon, \#\Theta, \#x, \uparrow\Xi, \updownarrow\Psi \vdash M$ and $\Phi, \uparrow\Xi, \updownarrow\Psi \vdash N$, then for every $n \geq \max\{D_0(M), D_0(N)\}$ it holds that $W^n_0(M\{x/N\}) \leq W^n_0(M) + NFO(x, M) \cdot W^n_0(N)$.

3. If $\Upsilon, \#\Theta, \downarrow x, \uparrow \Xi, \updownarrow \Psi \vdash M$ and $\Phi, \uparrow \Xi, \updownarrow \Psi \vdash N$, then for every $n \geq \max\{D_0(M), D_0(N)\}$ it holds that $W_0^n(M\{x/N\}) \leq W_0^n(M) + n \cdot W_0^n(N)$.
4. If $\Upsilon, \#\Theta, \uparrow \Xi, \updownarrow \Psi, \updownarrow x \vdash M$ and $\updownarrow \Psi \vdash N$, then for every $n \geq \max\{D_0(M), D_0(N)\}$ it holds that $W_0^n(M\{x/N\}) \leq W_0^n(M)$.
5. If $\Upsilon, \#\Theta, \uparrow \Xi, \uparrow x, \updownarrow \Psi \vdash M$ and $\updownarrow \Xi, \updownarrow \Psi \vdash N$, then for every $n \geq \max\{D_0(M), D_0(N)\}$ it holds that $W_0^n(M\{x/N\}) \leq W_0^n(M)$.

All the statements above can be proved, as usual, by induction on the (finite) number of inductive well-formation rules which are necessary to prove the judgment about M from something in $C_{\ell\Lambda^{4S}_{\infty}}(\mathbb{C}_{\ell\Lambda^{4S}_{\infty}})$. As an example, let us consider some inductive cases on Point 2., which is one of the most interesting:

- If M is proved well-formed by
$$\frac{}{\#\Theta, \uparrow \Xi, \updownarrow \Psi, \#x \vdash x}\ (\mathsf{vd})$$
then $M\{x/N\} = N$ and
$$\begin{aligned} W_0^n(M\{x/N\}) &= W_0^n(N) \\ &= 1 \cdot W_0^n(N) = NFO(x, M) \cdot W_0^n(N) \\ &\leq W_0^n(M) + NFO(x, M) \cdot W_0^n(N). \end{aligned}$$
- If M is proved well-formed by
$$\frac{\Upsilon, \#\Theta, \#x, \uparrow \Xi, \updownarrow \Psi \vdash L \quad \Pi, \#\Theta, \#x, \uparrow \Xi, \updownarrow \Psi \vdash P}{\Upsilon, \Pi, \#\Theta, \#x, \uparrow \Xi, \updownarrow \Psi \vdash LP}\ (\mathsf{a})$$
then $M\{x/N\} = (L\{x/N\})(P\{x/N\})$ and, by inductive hypothesis, we have
$$\begin{aligned} W_0^n(L\{x/N\}) &\leq W_0^n(L) + NFO(x, L) \cdot W_0^n(N); \\ W_0^n(P\{x/N\}) &\leq W_0^n(P) + NFO(x, P) \cdot W_0^n(N). \end{aligned}$$
But then:
$$\begin{aligned} W_0^n(M\{x/N\}) &= W_0^n(L\{x/N\}) + W_0^n(P\{x/N\}) \\ &\leq (W_0^n(L) + NFO(x, L) \cdot W_0^n(N)) \\ &\quad + (W_0^n(P) + NFO(x, P) \cdot W_0^n(N)) \\ &= (W_0^n(L) + W_0^n(P)) \\ &\quad + (NFO(x, L) + NFO(x, P)) \cdot W_0^n(N) \\ &= W_0^n(M) + NFO(x, M) \cdot W_0^n(N). \end{aligned}$$
- If M is proved well-formed by
$$\frac{\Xi, \uparrow \Psi, \updownarrow \Phi \vdash M}{\#\Theta, \#x, \downarrow \Xi, \uparrow \Psi, \updownarrow \Phi \vdash \downarrow M}\ (\mathsf{mi})$$
then x does not occur free in M and, as a consequence $M\{x/N\} = M$. The thesis easily follows.

With the five lemmas above in our hands, it is possible to prove that if $M \mapsto N$, then $W_0(M) > W_0(N)$. Let's proceed by cases depending on how $M \mapsto N$ is derived:

- If $M = (\lambda x.L)P$, then $\Upsilon, x, \#\Theta, \uparrow \Xi, \updownarrow \Psi \vdash L$ and $\Pi, \#\Theta, \uparrow \Xi, \updownarrow \Psi \vdash P$. We can apply Point 1. (and Lemma 16) obtaining
$$\begin{aligned} W_0(M) = W_0^{D_0(M)}(M) &= W_0^{D_0(M)}(L) + W_0^{D_0(M)}(P) + 1 \\ &> W_0^{D_0(M)}(L) + W_0^{D_0(M)}(P) \geq W_0^{D_0(M)}(L\{x/P\}) \\ &= W_0^{D_0(M)}(N) \geq W_0^{D_0(N)}(N) = W_0(N). \end{aligned}$$
- If $M = (\lambda \downarrow x.L) \downarrow P$, then we can distinguish two sub-cases:

- If $\Upsilon, \#x, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash L$ and $\Phi, \uparrow\Xi, \updownarrow\Psi \vdash P$, then we can apply Point 2. (and Lemma 16) obtaining

$$\begin{aligned} W_0(M) &= W_0^{D_0(M)}(M) \\ &= W_0^{D_0(M)}(L) + D_0(M) \cdot W_0^{D_0(M)}(P) + 1 \\ &> W_0^{D_0(M)}(L) + D_0(M) \cdot W_0^{D_0(M)}(P) \\ &\geq W_0^{D_0(M)}(L) + NFO(x, L) \cdot W_0^{D_0(M)}(P) \\ &\geq W_0^{D_0(M)}(L\{x/P\}) \\ &= W_0^{D_0(M)}(N) \geq W_0^{D_0(N)}(N) \\ &= W_0(N). \end{aligned}$$

- If $\Upsilon, \downarrow x, \#\Theta, \uparrow\Xi, \updownarrow\Psi \vdash L$ and $\Phi, \uparrow\Xi, \updownarrow\Psi \vdash P$, then we can apply Point 3. (and Lemma 16) obtaining

$$\begin{aligned} W_0(M) &= W_0^{D_0(M)}(M) \\ &= W_0^{D_0(M)}(L) + D_0(M) \cdot W_0^{D_0(M)}(P) + 1 \\ &> W_0^{D_0(M)}(L) + D_0(M) \cdot W_0^{D_0(M)}(P) \\ &\geq W_0^{D_0(M)}(L\{x/P\}) \\ &= W_0^{D_0(M)}(N) \geq W_0^{D_0(N)}(N) = W_0(N). \end{aligned}$$

- If $M = (\lambda\uparrow x.L)\uparrow P$, then $\Upsilon, \#\Theta, \uparrow x, \uparrow\Xi, \updownarrow\Psi \vdash L$ and $\uparrow\Xi, \updownarrow\Psi \vdash P$. We can apply Point 5. (and Lemma 16) obtaining

$$\begin{aligned} W_0(M) &= W_0^{D_0(M)}(M) = W_0^{D_0(M)}(L) + 1 \\ &> W_0^{D_0(M)}(L) \geq W_0^{D_0(M)}(L\{x/P\}) \\ &= W_0^{D_0(M)}(N) \geq W_0^{D_0(N)}(N) = W_0(N). \end{aligned}$$

Now, remember that $M \to_{n\bullet} N$ iff there are a n-context $C_{n\bullet}$ and two terms L and P such that $L \mapsto P$, $M = C_{n\bullet}[L]$, and $N = C_{n\bullet}[P]$. The thesis can be proved by induction on $C_{n\bullet}$. □

The Fundamental Lemma easily follows:

Proposition 3 (Fundamental Lemma)**.** *For every natural number $n \in \mathbb{N}$, the relation $\to_{n\bullet}$ is strongly normalising.*

4.2 Normalisation and Confluence

The Fundamental Lemma has a number of interesting consequences, which make the dynamics of $\ell\Lambda^{4S}_{\infty}$ definitely better-behaved than that of $\ell\Lambda_{\infty}$. The first such result we give is a Weak-Normalisation Theorem:

Theorem 2 (Normalisation)**.** *For every term $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$ there is a normal form $N \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$ such that $M \Rightarrow N$.*

Proof. This is an immediate consequence of the Fundamental Lemma: for every term M, first reduce (by $\to$) all redexes at depth 0, obtaining N, and then normalise all the subterms of N at depth 1 (by $\rightsquigarrow$). Conclude by observing that reduction at higher depths does not influence lower depths. □

The way the two modalities interact in $\ell\Lambda^{4S}_{\infty}$ has effects which go beyond normalisation. More specifically, the two relations $\to$ and $\rightsquigarrow$ do not interfere like in $\ell\Lambda_{\infty}$, and as a consequence, we get a Confluence Theorem:

Theorem 3 (Strong Confluence)**.** *If $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$, $M \Rightarrow N$ and $M \Rightarrow L$, then there is $P \in \ell\Lambda^{4S}_{\infty}$ such that $N \Rightarrow P$ and $L \Rightarrow P$.*

The path to confluence requires some auxiliary lemmas:

Lemma 18. *If $\Upsilon, \#\Theta, \uparrow\Xi, \updownarrow x, \updownarrow\Psi \vdash M$ and $\updownarrow\Psi \vdash N$, $M \Rightarrow L$, and $N \Rightarrow P$, then $M\{x/N\} \Rightarrow L\{x/P\}$.*

Proof. This is a coinduction on M. □

Lemma 19. *If $\Upsilon, \#\Theta, \uparrow\Xi, \uparrow x, \updownarrow\Psi \vdash M$ and $\updownarrow\Xi, \updownarrow\Psi \vdash N$, $M \rightsquigarrow L$, and $N \Rightarrow P$, then $M\{x/N\} \rightsquigarrow L\{x/P\}$.*

Proof. This is again a coinduction on the structure of M, exploiting Lemma 18 □

Lemma 20 (Noninterference)**.** *If $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$, $M \rightarrow_{0\bullet} N$ and $M \rightsquigarrow L$, then there is $P \in \ell\Lambda^{4S}_{\infty}$ such that $N \rightsquigarrow P$ and $L \rightarrow_{0\bullet} P$.*

Proof. By coinduction on the structure of M. Some interesting cases:

- If $M = QR$ and $Q \rightarrow_{0\bullet} S$, then $N = SR$. By definition, $Q \rightsquigarrow X$ and $R \rightsquigarrow Y$, where $L = XY$. By induction hypothesis, there is Z such that $S \rightsquigarrow Z$ and $X \rightarrow_{0\bullet} Z$. The term we are looking for, then, is just $P = ZY$. Indeed, $N = SR \rightsquigarrow ZY$ and, other other hand, $L = XY \rightarrow_{0\bullet} ZY$.
- If $M = (\lambda\uparrow x.Q)\uparrow R$ and $N = Q\{x/R\}$, then L is in the form $(\lambda\uparrow x.X)\uparrow Y$ where $Q \rightsquigarrow X$ and $R \Rightarrow Y$, and then we can apply Lemma 18 obtaining that $N \rightsquigarrow P = X\{x/Y\}$. On the other hand, $L \rightarrow_{0\bullet} P$.

□

But there is even more: $\rightarrow_{0\bullet}$ and $\rightsquigarrow$ commute.

Lemma 21 (Postponement)**.** *If $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$, $M \rightsquigarrow N \rightarrow_{0\bullet} L$, then there is $P \in \ell\Lambda^{4S}_{\infty}$ such that $M \rightarrow_{0\bullet} P \rightsquigarrow L$.*

Proof. Again a coinduction on the structure of M. Some interesting cases:

- We can exclude the case in which $M = \uparrow Q$, because in that case also N would be a coinductive boxes, and coinductive boxes are $\rightarrow_{0\bullet}$-normal forms.
- If $M = (\lambda\uparrow x.Q)\uparrow R$, $N = (\lambda\uparrow x.S)\uparrow X$ (where $Q \rightsquigarrow S$ and $R \Rightarrow X$) and $L = S\{x/X\}$, then Lemma 18 ensures that $P = Q\{x/R\}$ is such that $P \rightsquigarrow L$.

□

One-step reduction is not in general confluent in infinitary λ-calculi. However, $\rightarrow_{0\bullet}$ indeed is:

Proposition 4. *If $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$, $M \rightarrow^*_{0\bullet} N$, and $M \rightarrow^*_{0\bullet} L$, then there is P such that $N \rightarrow^*_{0\bullet} P$ and $L \rightarrow^*_{0\bullet} P$.*

Proof. This can be proved with standard techniques, keeping in mind that in an inductive abstraction $\lambda\downarrow x.M$, the variable x occurs finitely many times in M. □

The last two lemmas are of a techincal nature, but can be proved by relatively simple arguments:

Lemma 22. *Both $\rightsquigarrow$ and $\Rightarrow$ are reflexive.*

Proof. Easy. □

Lemma 23. *If $M \in \mathbb{T}_{\ell\Lambda^{4S}_{\infty}}$ and $M \rightarrow_{n\bullet} N$ (where $n \geq 1$), then $M \rightsquigarrow N$.*

Proof. Easy, given Lemma 22 □

We are finally able to prove the Confluence Theorem:

Proof of Theorem 3. We will show how to associate a term $P = f(\alpha)$ to any pair in the form $\alpha = (M \Rightarrow N, M \Rightarrow L)$ or in the form $\alpha = (M \rightsquigarrow N, M \rightsquigarrow L)$. The function f is defined by coinduction on the structure of the two proofs in α. This will be done in such a way that in the first case $N \Rightarrow f(\alpha)$ and $L \Rightarrow f(\alpha)$, while in the second case $N \rightsquigarrow f(\alpha)$ and $L \rightsquigarrow f(\alpha)$. If α is $(M \Rightarrow N, M \Rightarrow L)$, then by definition, $M \to^* Q \rightsquigarrow N$ and $M \to^* R \rightsquigarrow L$. Exploiting Lemma 23, Lemma 22, and Lemma 21, one obtains that there exist S and X such that $M \to^*_{0\bullet} S \rightsquigarrow N$ and $M \to^*_{0\bullet} X \rightsquigarrow L$. By Proposition 4, one obtains that there is Y with $S \to^*_{0\bullet} Y$ and $X \to_{0\bullet} Y$. By repeated application of Lemma 20 and Lemma 22, one can conclude there are Z and W such that $N \to^*_{0\bullet} Z$, $Y \rightsquigarrow Z$, $Y \rightsquigarrow W$ and $L \to^*_{0\bullet} W$. Now, let $f(\alpha)$ be just $f(Y \rightsquigarrow Z, Y \rightsquigarrow W)$. If, on the other hand, α is $(M \rightsquigarrow N, M \rightsquigarrow L)$, we can define f by induction on the proof of the two statements where, however, we are only interested in the last thunk of inductive rule instances. This is done in a natural way. As an example, if M is an application PQ, then clearly N is RS and L is XY, where $P \rightsquigarrow R$, $P \rightsquigarrow X$ $Q \rightsquigarrow S$, and $Q \rightsquigarrow Y$; moreover, $f(\alpha)$ is the term $f(P \rightsquigarrow R, P \rightsquigarrow X) f(Q \rightsquigarrow S, Q \rightsquigarrow Y)$. Notice how the function f is well defined, being a guarded recursive function on sets defined as greatest fixed points. □

Confluence and Weak Normalisation together imply that normal forms are unique:

Corollary 1 (Uniqueness of Normal Forms)**.** *Every term $M \in \mathbb{T}_{\ell\Lambda^{\mathsf{4S}}_\infty}$ has a unique normal form.*

Strangely enough, even if every term M has a normal form N, it is not guaranteed to reduce to it in every reduction order, simply because one can certainly choose to "skip" certain depths while normalising. In this sense, level-by-level sequences are normalising: they are not allowed to go to depth $n > m$ if there is a redex at depth m.

4.3 Expressive Power

At this point, one may wonder whether $\ell\Lambda^{\mathsf{4S}}_\infty$ is well-behaved simply because its expressive power is simply too low. Although at present we are not able to characterise the class of functions which can be represented in it, we can already give some interesting observations on its expressive power.

First of all, let us observe that the inductive fragment of $\ell\Lambda^{\mathsf{4S}}_\infty$ (i.e. the subsystem obtained by dropping coinductive boxes) is complete for polynomial time computable functions on finite strings, although inputs need to be represented as Church numerals for the result to hold: this is a consequence of polytime completeness for SLL [21].

About the "coinductive" expressive power of $\ell\Lambda^{\mathsf{4S}}_\infty$, we note that a form of guarded recursion can indeed be expressed, thanks to the very liberal exponential discipline of $\mathsf{4LL}$. Consider the term $M = \lambda\uparrow x.\lambda y.\lambda\uparrow z.y \uparrow (xxz \uparrow z)$ and define X to be $M \uparrow M$. One can easily verify that for any (closed) term N, the term $XN \uparrow N$ reduces in three steps to $N \uparrow (XN \uparrow N)$. In other words, then, X is indeed a fixed point combinator which however requires the argument functional to be applied to it twice.

The two observations above, taken together, mean that $\ell\Lambda^{\mathsf{4S}}_\infty$ is, at least, capable of expressing all functions from $(\mathbb{B}^*)^\infty$ to $(\mathbb{B}^*)^\infty$ such that for each n, the string at position n in the output stream can be computed in polynomial time from the string at position n in the input stream. Whether one could go (substantially) beyond this is an interesting problem that we leave for further work. One cannot, however, go too far, since the $\mathsf{4LL}$ exponential discipline imposes that all typable stream functions are causal, i.e., for each n, the value of the output at position n only depends on the input positions *up to* n, at least if one encodes streams as in Section 3.2.

5 Further Developments

We see this work only as a first step towards understanding how linear logic can be useful in taming the complexity of infinitary rewriting in the context of the λ-calculus. There are at least three different promising research directions that the results in this paper implicitly suggest. All of them are left for future work, and are outside the scope of this paper.

Semantics It would be interesting to generalise those semantic frameworks which work well for ordinary linear logic and λ-calculi to $\ell\Lambda_\infty$. One example is the so-called relational model of linear logic, in which formulas are interpreted as sets and morphisms are interpreted as binary relations. Noticeably, the exponential modality is interpreted by forming power multisets. Since the only kind of infinite regression we have in $\ell\Lambda_\infty$ is the one induced by coinductive boxes, it seems that the relation model should be adaptable to the calculus described here. Similarly, game semantics [2] and the geometry of interaction [14] seem to be well-suited to model infinitary rewriting.

Types The calculus $\ell\Lambda_\infty$ is untyped. Incepting types into it would first of all be a way to ensure the absence of deadlocks (consider, as an example, the term $(\lambda{\downarrow}x.x)({\uparrow}M)$). The natural candidate for a framework in which to develop a theory of types for $\ell\Lambda_\infty$ is the one of recursive types, given their inherent relation with infinite computations. Another challenge could be adapting linear dependent types [7] to an infinitary setting.

Implicit Complexity One of the most interesting applications of the linearisation of Λ_∞ as described here could come from implicit complexity, whose aim is characterising complexity classes by logical systems and programming languages without any reference to machine models nor to combinatorial concepts (e.g. polynomials). We think, in particular, that subsystems of $\ell\Lambda_\infty$ would be ideal candidates for characterising, e.g. type-2 polynomial time operators. This, however, would require a finer exponential discipline, e.g. an inductive-coinductive generalisation of the bounded exponential modality [12].

6 Related Work

Although this is arguably the first paper explicitly combining ideas coming from infinitary rewriting with resource-consciousness in the sense of linear logic, some works which are closely related to ours, but having different goals, have recently appeared.

First of all, one should mention Terui's work on computational ludics [29]: there, designs (i.e. the ludics' counterpart to proofs) are meant to both capture syntax (proofs) and semantics (functions), and are thus infinitary in nature. However, the overall objective in [29] is different from ours: while we want to stay as close as possible to the λ-calculus so as to inspire the design of techniques guaranteeing termination of programs dealing with infinite data structures, Terui's aim is to better understand usual, finitary, computational complexity. Technically, the main difference is that we focus on the exponentials and let them be the core of our approach, while computational ludics is strongly based on focalisation: time passes whenever polarity changes.

Another closely related work is a recent one by Mazza [24], that shows how the ordinary, finitary, λ-calculus can be seen as the metric completion of a much weaker system, namely the affine λ-calculus. Again, the main commonalities with this paper are on the one hand the presence of infinite terms, and on the other a common technical background, namely that of linear logic. Again, the emphasis is different: we, following [19], somehow aim at going *beyond* finitary λ-calculus, while Mazza's focus is on the subrecursive, finite world: he is not even concerned with reduction of infinite length.

If one forgets about infinitary rewriting, linear logic has already been shown to be a formidable tool to support the process of isolating classes of λ-terms having good, quantitative normalisation properties. One can, for example, cite the work by Baillot and Terui [3] or the one by Gaboardi and Ronchi here [11]. This paper can be seen as a natural step towards transferring these techniques to the realm of infinitary rewriting.

Finally, among the many works on type-theoretical approaches to termination and productivity, the closest to ours is certainly the recent contribution by Cave et al. [6]: our treatment of the coinductive modality is very reminiscent to their way of handling LTL operators.

Acknowledgment

The author would like to thank Patrick Baillot, Marco Gaboardi and Olivier Laurent for useful discussions about the topics of this paper. The author is partially supported by the ANR project 12IS02001 PACE and the ANR project 14CE250005 ELICA.

References

[1] A. Abel. Mixed inductive/coinductive types and strong normalization. In *APLAS*, volume 4807 of *LNCS*, pages 286–301, 2007.

[2] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for pcf. *Inf. Comput.*, 163(2):409–470, 2000.

[3] P. Baillot and K. Terui. Light types for polynomial time computation in lambda calculus. *Inf. Comput.*, 207(1):41–62, 2009.

[4] H. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*. Elsevier, 1980.

[5] H. Barendregt and J. W. Klop. Applications of infinitary lambda calculus. *Inf. Comput.*, 207(5):559–582, 2009.

[6] A. Cave, F. Ferreira, P. Panangaden, and B. Pientka. Fair reactive programming. In *POPL*, pages 361–372, 2014.

[7] U. Dal Lago and M. Gaboardi. Linear dependent types and relative completeness. In *LICS*, pages 133–142. IEEE Computer Society, 2011.

[8] V. Danos and J.-B. Joinet. Linear logic and elementary time. *Inf. Comput.*, 183(1):123–137, 2003.

[9] E. W. Dijkstra. On the productivity of recursive definitions. Personal note EWD 749. Available at `http://www.cs.utexas.edu/users/EWD/ewd07xx/EWD749.PDF`, September 1980.

[10] J. Endrullis and A. Polonsky. Infinitary rewriting coinductively. In *TYPES*, pages 16–27, 2011.

[11] M. Gaboardi and S. Ronchi Della Rocca. A soft type assignment system for *lambda* -calculus. In *CSL*, volume 4646 of *LNCS*, pages 253–267, 2007.

[12] J. Girard, A. Scedrov, and P. J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.

[13] J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[14] J.-Y. Girard. Geometry of interaction I: interpretation of System F. In *Proceedings of the Logic Colloquium '88*, pages 221–260. North Holland, 1989.

[15] J.-Y. Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.

[16] J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *POPL*, pages 410–423. ACM Press, 1996.

[17] R. Kennaway and F.-J. de Vries. *Term Rewriting Systems*, chapter Infinitary Rewriting, pages 668–711. Cambridge University Press, 2003.

[18] R. Kennaway, J. W. Klop, M. R. Sleep, and F.-J. de Vries. Transfinite reductions in orthogonal term rewriting systems. In *RTA*, volume 488 of *LNCS*, pages 1–12, 1991.

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

[20] J. Ketema and J. G. Simonsen. Infinitary combinatory reduction systems. *Inf. Comput.*, 209(6):893–926, 2011.

[21] Y. Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.

[22] X. Leroy and H. Grall. Coinductive big-step operational semantics. *Inf. Comput.*, 207(2):284–304, 2009.

[23] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. *Electr. Notes Theor. Comput. Sci.*, 1:370–392, 1995.

[24] D. Mazza. An infinitary affine lambda-calculus isomorphic to the full lambda-calculus. In *LICS*, pages 471–480. IEEE, 2012.

[25] M. Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In *LPAR*, volume 624 of *LNCS*, pages 190–201, 1992.

[26] M. Parigot. Recursive programming with proofs. *Theor. Comput. Sci.*, 94(2):335–336, 1992.

[27] C. Raffalli. Data types, infinity and equality in system af_2. In *CSL*, volume 832 of *LNCS*, pages 280–294, 1993.

[28] A. K. Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In *RTA*, volume 3467 of *LNCS*, pages 219–234, 2005.

[29] K. Terui. Computational ludics. *Theor. Comput. Sci.*, 412(20):2048–2071, 2011.

[30] C. Wadsworth. Some unusual λ-calculus numeral systems. In J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[31] K. Weihrauch. *Computable analysis: an introduction*. Springer-Verlag New York, Inc., 2000.