Böhm and Taylor for All!

Aloÿs Dufour
Université Sorbonne Paris Nord, LIPN, CNRS, Villetaneuse, France

Damiano Mazza
CNRS, LIPN, Université Sorbonne Paris Nord, Villetaneuse, France

Abstract

Böhm approximations, used in the definition of Böhm trees, are a staple of the semantics of the lambda-calculus. Introduced more recently by Ehrhard and Regnier, Taylor approximations provide a quantitative account of the behavior of programs and are well-known to be connected to intersection types. The key relation between these two notions of approximations is a commutation theorem, roughly stating that Taylor approximations of Böhm trees are the same as Böhm trees of Taylor approximations. Böhm and Taylor approximations are available for several variants or extensions of the lambda-calculus and, in some cases, commutation theorems are known. In this paper, we define Böhm and Taylor approximations and prove the commutation theorem in a very general setting. We also introduce (non-idempotent) intersection types at this level of generality. From this, we show how the commutation theorem and intersection types may be applied to any calculus embedding in a sufficiently nice way into our general calculus. All known Böhm-Taylor commutation theorems, as well as new ones, follow by this uniform construction.

2012 ACM Subject Classification Theory of computation → Linear logic; Theory of computation → Denotational semantics; Theory of computation → Operational semantics; Theory of computation → Lambda calculus; Theory of computation → Process calculi

Keywords and phrases Linear logic, Differential linear logic, Taylor expansion of lambda-terms, Böhm trees, Process calculi

Digital Object Identifier 10.4230/LIPIcs.FSCD.2024.26

Acknowledgements The authors wish to thank Michele Pagani, who pointed out Lemma 23.

1 Introduction

Böhm trees [4, 5] and their variants are a staple of the semantics of λ-calculus. Intuitively, they are possibly infinite normal forms representing the essence of the behavior of a λ-term. As such, they may be seen as limits of a set of finite approximations, called *Böhm approximations*, describing larger and larger portions of the Böhm tree.

More recently, Ehrhard and Regnier introduced another notion of approximation for the λ-calculus, underlying their *Taylor expansion* [23, 25]. Based on the idea that programs may be seen as analytic functions on certain topological vector spaces [16, 17], this notion may be stripped of its quantitative aspects (the coefficients of the Taylor series) and, like a Böhm tree, be presented as a set of finite approximations. These *Taylor approximations* are syntactically very different from Böhm approximations: they are not necessarily normal forms, and they are linear, in the sense of linear logic [27] (they are related to Boudol's *terms with multiplicities* [7]), so their execution length is bounded by their size.

Given a λ-term t, since a Böhm approximation is, essentially, also a λ-term, one may take the set of all Taylor approximations of all Böhm approximations of t, yielding a set $\mathcal{T}(\mathrm{BT}(t))$. On the other hand, since Taylor approximations always normalize, one may take the set of all normal forms of all Taylor approximations of t, yielding a set $\mathrm{NF}(\mathcal{T}(t))$. A key result of Ehrhard and Regnier, known as the *commutation theorem*, states that $\mathcal{T}(\mathrm{BT}(t)) = \mathrm{NF}(\mathcal{T}(t))$. This connection between Böhm and Taylor approximations is a surprisingly powerful tool, implying a wealth of fundamental theorems in the pure λ-calculus [3].

The theory of Taylor approximations has been extended to call-by-value [18], to call-by-push-value [10] and to the $\lambda\mu$-calculus [2]. In the call-by-value setting, where a notion of Böhm tree is available, the commutation theorem is known to hold [29]. These results exploit the fact that, in reality, the notion of Taylor expansion exists in the much more general setting of *differential linear logic* [20]. Informally, whenever a system $\mathcal{S}$ may be encoded in differential linear logic, a notion of Taylor approximation for $\mathcal{S}$ may be "pulled back" along the encoding.

Interestingly, usual Böhm trees, as well as call-by-value Böhm trees, may also be seen as resulting from a general notion of "Böhm tree" for linear logic, pulled back along Girard's call-by-name and call-by-value encodings. It is natural to wonder whether the different forms of the commutation theorem are also "pullbacks" of a more general commutation theorem.

The main goal of this paper is to give an affirmative answer to the above question. We introduce a process calculus $\mathcal{P}roc$ corresponding to a form of differential linear logic, define Böhm and Taylor approximations for it and prove the commutation theorem at this level of generality. We also show that, as soon as a system $\mathcal{S}$ embeds in $\mathcal{P}roc$ in a sufficiently nice way, then Böhm trees, Taylor expansion and the commutation theorem automatically "pull back" to $\mathcal{S}$. The known commutation theorems of [23, 25, 29] are covered by our results. Furthermore, we show how Böhm trees for call-by-push-value and a fragment of the π-calculus may also be defined.

It is well-known that intersection types [11, 6], in particular in their non-idempotent version [26, 15], are strongly related to Taylor approximations [14, 18, 32]. We provide a system of non-idempotent intersection types for $\mathcal{P}roc$ and show how it too "pulls back" along embeddings, automatically characterizing the existence of normal forms via typability, as long as these are properly reflected in $\mathcal{P}roc$. Known examples, such as the above or those of [8], fit in this picture. From [32], we know that other forms of intersection types (affine, idempotent) may be treated in a similar way, but in this paper we restrict to the linear non-idempotent case for briefness.

For the reader acquainted with proof nets, our calculus $\mathcal{P}roc$ is a process calculus representation of untyped *proof structures* (*i.e*, not necessarily logically correct objects) of differential linear logic. More precisely, of a non-polarized version of Honda and Laurent proof structures [28]. In §5.1 we give a more precise statement about the connection with linear logic proofs. The reduction rules of $\mathcal{P}roc$ reflect the process calculi tradition and are less fine-grained than those of usual differential nets [24, 34, 33], while still being semantically correct. We do not claim any canonicity of $\mathcal{P}roc$: any other syntax for classical differential linear logic (for instance, an extension of Accattoli's syntax [1]) would probably work. The only real merit of $\mathcal{P}roc$ is to provide a manageable syntax.

Another difference is that we do not consider formal sums. In the literature on differential λ-calculi, formal sums as used to represent non-determinism: a non deterministic choice like $t \to u_1$ and $t \to u_2$ is expressed by the deterministic reduction $t \to u_1 + u_2$. By contrast, as customary in process calculi, there are no formal sums in $\mathcal{P}roc$ and we allow uncontrolled non-determinism (the calculus does not enjoy confluence). Also, our Taylor approximations are *rigid* in the sense of [35, 32]. Intuitively, this is implicit in the usual communication rule of polyadic process calculi, which is something like $\overline{a}\langle b_1, b_2\rangle \mid a(c_1, c_2).P \to P\{b_1/c_1\}\{b_2/c_2\}$ (cf. Definition 7, rule $\otimes/\parr$). Non-rigid syntaxes like the ones traditionally considered for defining Taylor approximations would correspond to b_1, b_2 being unordered in the output $\overline{a}\langle b_1, b_2\rangle$ (*i.e*, they form a multiset rather than a list) and the above reduction would become $\overline{a}\langle b_1, b_2\rangle \mid a(c_1, c_2).P \to P\{b_1/c_1\}\{b_2/c_2\} + P\{b_2/c_1\}\{b_1/c_2\}$, that is, we consider every possible ordering of the multiset and use formal sums to collect all results. Using rigidity and

dispensing with sums yields great syntactic simplifications with essentially no semantic loss. We do, however, neglect quantitative aspects (the coefficients of the Taylor series). Luckily, in a great number of interesting cases (for instance those of [3]), these are not needed.

About the applicability of our work, we must stress that the fact that an embedding $\mathcal{S} \to \mathcal{P}roc$ immediately establishes a nice theory of Böhm and Taylor approximations for $\mathcal{S}$ does not say anything about the actual interest of this theory. First of all, as is, the theory is formulated in the syntax of $\mathcal{P}roc$, and reformulating it in "$\mathcal{S}$-friendly" syntax is not automatic (the call-by-push-value case of §6.1 is an example). Second, its relevance and usefulness must be ascertained case by case. For example, we have not yet investigated the Böhm trees for the stack calculus or for the fragment of the π-calculus presented in §6.2 and §6.3.

Finally, we want to point out that our notion of embedding (Definition 1) is rather naive. While sufficient for the examples shown in this paper, many encodings, especially of process calculi, are not embeddings in the technical sense of this paper, typically because they are up to some equivalence. For instance, the encoding of the $\lambda\mu$-calculus in the stack calculus mentioned in §6.2 is up to β-equivalence, so it is not directly covered by this paper. Extending our results of §4.3 and §5.2 to a more general class of embeddings is an interesting subject for future work.

2 Preliminaries

In this paper we consider *reduction systems*, which are given by:

- a set of *objects*, to be thought of as terms, proofs, programs, states...
- a binary relation $\to^*$ which is reflexive and transitive.

So a reduction system is exactly the same thing as a preorder.

A *morphism* of reduction systems $f : \mathcal{S} \to \mathcal{T}$ is just a monotonic function: $s \to^* s'$ implies $f(s) \to^* f(s')$.

▶ **Definition 1** (embedding). *An* embedding $f : \mathcal{S} \to \mathcal{T}$ *is a morphism such that, for every object s of $\mathcal{S}$, $f(s) \to^* t'$ implies that there exists s' such that $t' \to^* f(s')$ and $s \to^* s'$.*

The following is straightforward:

▶ **Lemma 2.** *If $f : \mathcal{S} \to \mathcal{T}$ and $g : \mathcal{T} \to \mathcal{U}$ are embeddings, then $g \circ f$ is an embedding.*

3 Processes

We fix two disjoint, countably infinite sets of *linear names*, ranged over by $a, b, c \dots$ and *cartesian names*, ranged over by $x, y, z \dots$ As customary in process calculi, we denote by $\widetilde{a}$ sequences (possibly empty) of linear names, and we write $|\widetilde{a}|$ for the length of $\widetilde{a}$.

▶ **Definition 3** (pre-process, context). Pre-processes *are defined as follows:*

$$P, Q ::= \mathbf{0} \mid P \mid Q \mid a \leftrightarrow b \mid \overline{a}\langle\widetilde{b}\rangle \mid a\langle\widetilde{b}\rangle \qquad \textit{(linear pre-processes)}$$
$$\mid \nu x P \mid \overline{a}(x)P \mid a(x)P \mid \overline{x}\langle a\rangle \mid x\langle a\rangle \mid !x(a).P$$

Linear pre-processes *are those generated only by the first line. We use $p, q \dots$ to range over linear pre-processes. Pre-processes of the form $!x(a).P$ are called* boxes. *In the literature on process calculi, $a \leftrightarrow b$ is usually called* linear forwarder*; we will call it* axiom *to stress the connection with proof nets.*

Contexts *are defined as pre-processes but with the addition of the* hole $\{-\}$*. As customary, we consider only contexts having exactly one occurrence of the hole, and denote them by* C*. We denote by* $\mathsf{C}\{P\}$ *the pre-process obtained by plugging pre-process* P *in the hole of* C*.*

▶ **Definition 4** (occurrences of names)**.** *We say that a linear name* a *occurs as* subject *in* $\overline{a}\langle\widetilde{b}\rangle$*,* $a\langle\widetilde{b}\rangle$*,* $\overline{a}(x)P$*,* $a(x)P$*,* $a \leftrightarrow b$ *and* $b \leftrightarrow a$*. All other occurrence of* a *are as* object*.*

The notations νx*,* $\overline{a}(x)$ *and* $a(x)$ *are called* cartesian binders*: in* νxP*,* $\overline{a}(x)P$ *and* $a(x)P$ *the cartesian name* x *is bound, and* α*-equivalence applies as usual. A cartesian name which is not bound is* free*.*

If a cartesian name x *appears elsewhere than in a binder, then we say that it appears as* subject*, and such an occurrence is* positive *if it is of the form* $\overline{x}\langle a\rangle$*, or* negative *if it is of the form* $x\langle a\rangle$ *or* $!x(a).P$*.*

▶ **Definition 5** (process)**.** *A (linear)* process *is a (linear) pre-process verifying the following:*

- *every linear name occurs at most twice. If it occurs once, we say that it is* free*, otherwise it is* bound *and* α*-equivalence applies to it. We write* $\mathsf{fn}(P)$ *for the set of free names of* P*, both linear and cartesian.*
- *In* $\overline{a}(x)P$ *(resp.* $a(x)P$*) every free occurrence of* x *in* P *(if any) is positive (resp. negative).*
- *In* $!x(a).P$*,* $\mathsf{fn}(P) = \{a\} \cup X$ *where* X *consists solely of cartesian variables with positive occurrences in* P *(the case* $X = \emptyset$ *is allowed).*

▶ **Definition 6** (structural congruence)**.** *Structural congruence is the relfexive, symmetric, transitive and contextual closure of the following rules:*

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad \nu x\mathbf{0} \equiv \mathbf{0}$$

$$\beta P \mid Q \equiv \beta(P \mid Q) \qquad \beta\gamma P \equiv \gamma\beta P \qquad a \leftrightarrow b \equiv b \leftrightarrow a \qquad a \leftrightarrow b \mid P \equiv P\{a/b\},$$

where: in the bottom row, β *and* γ *are cartesian binders and* β *binds* $x \notin \mathsf{fn}(Q)$*; in the last equation, we require* $b \in \mathsf{fn}(P)$*, and* $P\{a/b\}$ *means, as customary, that we rename the unique free occurrence of* b *in* P *with* a*.*

One may readily check that being a process is preserved under structural congruence.

For the reader acquainted with proof nets of differential linear logic, the following "dictionary" may be helpful (the unacquainted reader may safely ignore this paragraph): $\mathbf{0}$ is the empty net; $P \mid Q$ is juxtaposition of two nets; as mentioned above, $a \leftrightarrow b$ is an axiom; $\overline{a}\langle\widetilde{b}\rangle$ and $a\langle\widetilde{b}\rangle$ are n-ary tensor and par nodes, respectively; νxP is an "exponential cut" ("linear cuts" are represented as explained in Definition 8 below); $\overline{a}(x)P$ and $a(x)P$ are n-ary contraction and cocontraction, respectively (the arity is the number of occurrences of x in P); $\overline{x}\langle a\rangle$ and $x\langle a\rangle$ are derelection and coderelection nodes, respectively; as already mentioned, $!x(a).P$ is an exponential box, with principal port x and as many auxiliary ports as there are occurrences of free names in P. The idea is that the free names of a process correspond to the conclusions of an untyped net. Structural congruence corresponds to equality of nets (in the sense of graphical representations), plus elimination of cuts with axioms (this is consistent with interaction nets [24], in which axioms are just wires).

▶ **Definition 7** (reduction)**.** *The basic reduction rules are follows:*

$$\begin{array}{rlll}
\overline{a}\langle\widetilde{b}\rangle \mid a\langle\widetilde{c}\rangle \mid P & \to_{\otimes/\parr} & P\{\widetilde{b}/\widetilde{c}\} & |\widetilde{b}| = |\widetilde{c}|,\ \widetilde{c} \in \mathsf{fn}(P) \\
\overline{a}(x)P \mid a(x)Q & \to_{!/?} & \nu x(P \mid Q) & \\
\nu x(\overline{x}\langle a\rangle \mid x\langle b\rangle \mid P) & \to_{\mathsf{cod}_0} & \nu xP\{a/b\} & b \in \mathsf{fn}(P) \\
\nu x(\mathsf{C}\{\overline{x}\langle a\rangle\} \mid !x(b).P) & \to_{\mathsf{c}} & \nu x(\mathsf{C}\{P\{a/b\}\} \mid !x(b).P) & \\
\nu x(!x(a).P \mid Q) & \to_{\mathsf{w}} & \nu xQ & x \notin \mathsf{fn}_+(Q)
\end{array}$$

plus the rule

$$\nu x(!y(c).\mathsf{C}\{\overline{x}\langle a\rangle\} \mid x\langle b\rangle \mid P) \;\to_{\mathsf{cod}_!}\; \nu x(!y(c).\nu w\mathsf{C}\{\overline{w}\langle a\rangle\} \mid y\langle c\rangle \mid \nu z(\mathsf{C}\{\overline{z}\langle a\rangle\} \mid z\langle b\rangle) \mid P)$$

in which w and z are fresh names. In the w *rule, $x \notin \mathsf{fn}_+(Q)$ means that x has no positive free occurrence in Q.*

If $R \to R'$ by means of one of the above basic reductions, we write $P \to Q$ to say there exists a context C such that $P \equiv \mathsf{C}\{R\}$ and $Q \equiv \mathsf{C}\{R'\}$. We write $\to^*$ for the reflexive-transitive closure of $\to$. This induces a reduction system $\mathcal{P}roc$ (resp. $\mathcal{L}in\mathcal{P}roc$) whose objects are processes (resp. linear processes) modulo structural equivalence.

▶ **Definition 8** (cut-free process). *A* linear cut *in a process is a linear name occurring twice as subject. A* cartesian cut *is any subprocess of the form $\nu x P$. A process is* cut-free *if it contains neither linear nor cartesian cuts. A process P* has a cut-free form *if $P \to^* N$ with N cut-free.*

The next two paragraphs provide additional explanations for linear logic experts, which unacquainted readers may ignore. The basic rules of Definition 7 are a reformulation of cut-elimination steps of differential linear logic proof nets. The rule $\otimes/\parr$ is the multiplicative step. The rule !/? reduces a cut between a contraction and a cocontraction. However, instead of the usual rule making the two commute with each other, this rule creates a cartesian cut (which we could also call "exponential cut"). Cartesian cuts should be seen as *communication zones* in the sense of Ehrhard and Laurent [22]: any number of derelictions and coderelections/boxes (as many as the premises of the contraction and the cocontraction originating the cut) may be paired up non-deterministically in a cartesian cut, using the rules cod_0 (derelection/codereliction), $\mathsf{cod}_!$ (a codereliction interacting with the border of a box) or c (a dereliction extracting one copy of a box). The rule w erases a box when there is no dereliction left. Intuitively, we are taking communication zones as primitive, rather than implementing them as Ehrhard and Laurent do in [22].

In fact, the usual presentation of cut-elimination of differential linear logic [24, 34, 33] is based on finer-grained rules than ours. In their encoding of process calculi, Ehrhard and Laurent use this finer granularity to implement "communication zones" ensuring that inputs may interact with outputs [22]. This means, in particular, that our formulation is semantically correct with respect to the usual one: the finer-granularity rules may simulate our rules. However, our formulation has the advantage of matching the tradition of process calculi, as well as solving the problems pointed out in [31], which are precisely due to the fact that the usual cut-elimination rules are too fine-grained for expressing concurrency in the same way as process calculi.

Nevertheless, $\mathcal{P}roc$ does exhibit some unusual features with respect to standard process calculi. The most striking one is perhaps the presence of linear names and the convention that a linear name is bound as soon as it appears twice. For instance, the process $\overline{a}\langle\rangle \mid a\langle\rangle \mid \overline{b}\langle\rangle$ would be traditionally written $\nu a(\overline{a}\langle\rangle \mid a\langle\rangle \mid \overline{b}\langle\rangle)$. This would lead to a proliferation of ν's, so we chose to leave these name restrictions implicit.

Another remarkable difference is that we allow reduction to occur inside boxes, *i.e*, if $P \to Q$, then $!x(a).P \to !x(a).Q$. This is useful for encoding λ-calculi whose reductions may occur in arbitrary positions. It is essential for recovering usual Böhm trees, for example. In §5.2 we will consider *shallow reduction*, which only happens outside of boxes and corresponds to standard reduction of process calculi. This is the notion of reduction to which intersection types are most immediately applicable. It is also useful for encoding weak reduction strategies of λ-calculi, which do not reduce under λ.

Observe that all basic reductions involve a cut. Therefore, a cut-free process is normal with respect to reduction. The converse is false: for example, $\overline{a}\langle\rangle \mid a\langle b\rangle$ is a normal process containing a cut with an "arity mismatch". Cuts to which no reduction applies are called *irreducible*. Other examples of irreducible linear cuts are "clashes" such as $\overline{a}\langle b\rangle \mid a(x)P$ or $\overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$, or "vicious circles" such as $a \leftrightarrow a$ (for the acquainted reader: the first corresponds to a cut beween a tensor and a contraction; the second to a cut between two tensors; the third to a cut between the two conclusions of an axiom). Irreducible cartesian cuts are of the form $\nu x(\overline{x}\langle a\rangle \mid P)$ (resp. $\nu x(x\langle a\rangle \mid P)$) with P containing no negative (resp. positive) occurrence of x (for the acquainted reader: the first corresponds to a cut between a dereliction and a coweakening, the second to a cut between a codereliction and a weakening). In §5.1 we will introduce types and correctness. The former eliminate clashes and arity mismatches, the latter vicious circles. Irreducible cartesian cuts, instead, may be present even in well-typed correct processes. They correspond to situations which, in the usual syntax of differential linear logic using formal sums, reduce to the empty sum.

The following lemma will be useful later:

▶ **Lemma 9.** *$P \to^* Q$ implies $P\{a/b\} \to^* Q\{a/b\}$.*

Proof. By induction on the length of the reduction, we reduce to the case of a single step $\mathsf{C}\{R\} \to \mathsf{C}\{R'\}$, which is proved by induction on C and by inspection of Definition 7. ◀

4 Approximations

4.1 Taylor Approximations

▶ **Definition 10** (Taylor process)**.** Taylor processes *are linear processes in which certain inputs and outputs are marked as "special" and denoted by $\overline{a}\langle\!\langle\widetilde{b}\rangle\!\rangle$ and $a\langle\!\langle\widetilde{b}\rangle\!\rangle$. Reduction is defined just as in linear processes, i.e, with rule $\otimes/\parr$ of Definition 7, but restricted to special/special and non-special/non-special pairs, that is, $\overline{a}\langle\widetilde{b}\rangle \mid a\langle\!\langle\widetilde{c}\rangle\!\rangle$ is irreducible even in case $|\widetilde{b}| = |\widetilde{c}|$. We denote by $\mathcal{T}ay$ the reduction system with Taylor processes as objects.*

There is an obvious morphism $\mathcal{T}ay \to \mathcal{L}in\mathcal{P}roc$ which forgets the "special" annotations. It is *not* an embedding: $\overline{a}\langle\!\langle\rangle\!\rangle \mid a\langle\rangle$ is mapped to $\overline{a}\langle\rangle \mid a\langle\rangle$, which reduces to $\mathbf{0}$, but the original process cannot reduce.

▶ **Lemma 11.** *In Taylor processes, reduction is strongly confluent and terminating.*

Proof. The only reduction steps are of kind $\otimes/\parr$, which cannot overlap, hence strong confluence. For what concerns termination, define the size of a Taylor process to be the number of axioms and subprocesses of the form $\overline{a}\langle\widetilde{b}\rangle$, $a\langle\widetilde{b}\rangle$, $\overline{a}\langle\!\langle\widetilde{b}\rangle\!\rangle$, $a\langle\!\langle\widetilde{b}\rangle\!\rangle$ which are present in the process. By inspecting Definition 6 and the $\otimes/\parr$ rule, we immediately see that size is preserved by structural congruence and that it strictly decreases under reduction, which implies termination. ◀

The *Taylor approximation relation* is defined in Fig. 1. It uses *approximation judgments* of the form $p \sqsubset P \vdash \Xi; \Xi'$ where:

- p is a Taylor process and P an arbitrary process;
- Ξ and Ξ' are disjoint finite sets of pairs of the form $a \sqsubset x$, where a is a linear name not appearing free in P and x a cartesian name, such that every linear name appears at most once in $\Xi \cup \Xi'$.

$$\frac{}{\mathbf{0} \sqsubset \mathbf{0} \vdash ;} \qquad \frac{p \sqsubset P \vdash \Xi ; \Xi' \quad q \sqsubset Q \vdash \Upsilon ; \Upsilon'}{p \mid q \sqsubset P \mid Q \vdash \Xi, \Upsilon ; \Xi', \Upsilon'} \qquad \frac{}{a \leftrightarrow b \sqsubset a \leftrightarrow b \vdash ;}$$

$$\frac{}{\overline{a}\langle \widetilde{b} \rangle \sqsubset \overline{a}\langle \widetilde{b} \rangle \vdash ;} \qquad \frac{}{a\langle \widetilde{b} \rangle \sqsubset a\langle \widetilde{b} \rangle \vdash ;} \qquad \frac{p \sqsubset P \vdash \Xi, \widetilde{a} \sqsubset x ; \Xi', \widetilde{b} \sqsubset x}{p\{\widetilde{a}/\widetilde{b}\} \sqsubset \nu x P \vdash \Xi ; \Xi'} \; |\widetilde{a}| = |\widetilde{b}|$$

$$\frac{p \sqsubset P \vdash \Xi, \widetilde{b} \sqsubset x ; \Xi'}{\overline{a}\langle\!\langle \widetilde{b} \rangle\!\rangle \mid p \sqsubset \overline{a}(x) P \vdash \Xi ; \Xi'} \; x \notin \Xi \qquad \frac{p \sqsubset P \vdash \Xi ; \Xi', \widetilde{b} \sqsubset x}{a\langle\!\langle \widetilde{b} \rangle\!\rangle \mid p \sqsubset a(x) P \vdash \Xi ; \Xi'} \; x \notin \Xi'$$

$$\frac{}{a \leftrightarrow b \sqsubset \overline{x}\langle b \rangle \vdash a \sqsubset x ;} \; a \neq b \qquad \frac{}{a \leftrightarrow b \sqsubset x\langle b \rangle \vdash ; a \sqsubset x} \; a \neq b$$

$$\frac{p_1 \sqsubset P\{a_1/a\} \vdash \Xi_1 ; \quad \dots \quad p_n \sqsubset P\{a_n/a\} \vdash \Xi_n ;}{p_1 \mid \dots \mid p_n \sqsubset !x(a).P \vdash \Xi_1, \dots, \Xi_n ; a_1 \sqsubset x, \dots, a_n \sqsubset x} \; \forall i \, a_i \notin \mathsf{fn}(P)$$

Figure 1 Taylor approximations.

The intuition is that $a \sqsubset x$ in Ξ (resp. Ξ') means that the linear name a approximates one positive (resp. negative) occurrence of the cartesian name x.

In the sequel, we will write $p \sqsubset P$ when a judgment of the form $p \sqsubset P \vdash \Xi ; \Xi'$ is derivable for some Ξ, Ξ'.

▶ **Lemma 12.** *Let $p \sqsubset P$. Then:*

1. *$P = Q\{a/b\}$ iff $p = q\{a/b\}$ for some $q \sqsubset Q$;*
2. *$P = \mathsf{C}\{Q\}$ with $\mathsf{fn}(Q) = \{a\} \cup X$ with X consisting of cartesian names with only positive occurrences implies $p \equiv t \mid q_1 \mid \dots \mid q_n$ for some q_i and t such that $t \sqsubset \mathsf{C}\{\overline{x}\langle a \rangle\}$ whenever $x \in \mathsf{fn}(\mathsf{C}\{\overline{x}\langle a \rangle\})$, and $q_i \sqsubset Q\{a_i/a\}$ for all $1 \leq i \leq n$ and some pairwise distinct a_i;*
3. *$P \equiv P'$ (resp. $p \equiv p'$) implies $p \equiv p'$ for some p' (resp. $P \equiv P'$ for some P') such that $p' \sqsubset P'$.*

Proof. Point (1) is proved by induction on P. Point (2) is proved by induction on C. Point (3) is proved by checking every rule of Definition 6 and then by induction on contexts. ◀

Point (3) of Lemma 12 assures us that we may transparently use structural congruence with Taylor approximations, which is what we will do from now on.

We now prove the two fundamental properties of Taylor approximations, namely that they may be pulled back along arbitrary reductions and pushed forward along reductions to cut-free forms.

▶ **Lemma 13** (pull-back). *Let $P \to^* Q$ and $q \sqsubset Q$. Then, there exists $p \sqsubset P$ such that $p \to^* q$. Diagrammatically:*

$$\begin{array}{ccc} P & \longrightarrow^* & Q \\ & & \sqcup \\ & & q \end{array} \qquad \Longrightarrow \qquad \begin{array}{ccc} P & \longrightarrow^* & Q \\ \sqcup & & \sqcup \\ p & \longrightarrow^* & q \end{array}$$

Proof. We start by proving the lemma when $P = R$, $Q = R'$ and $R \to R'$ by means of one of the basic rules of Definition 7. The cases $\otimes/\parr$ and $!/?$ are immediate applications of point

(1) of Lemma 12. For the cases cod_0, $\mathsf{cod}_!$, c and w, one shows that $q \sqsubset R'$ implies $q \sqsubset R$, so the pull-back is the empty reduction. This claim is straightforward for the rules cod_0, $\mathsf{cod}_!$ and w; for the rule c, point (2) of Lemma 12 is used.

Next, we prove the lemma for one-step reduction, that is, when $P \equiv \mathsf{C}\{R\}$, $Q \equiv \mathsf{C}\{R'\}$ and $R \to R'$ by means of a basic rule. Define the *depth* of the context C to be number of nested boxes within which the hole is located. The proof is by induction on the depth of C. If the depth is zero, then $\mathsf{C} \equiv \nu\widetilde{z}(S \mid \{-\})$ for some S, from which we infer $q \equiv (s \mid r')\sigma$ where $s \sqsubset S$, $r' \sqsubset R'$ and σ is a substitution. We apply the result we proved above, and obtain $r \sqsubset R$ such that $r \to^* r'$. Then, if we let $p := (s \mid r)\sigma$, we have $p \sqsubset P$ such that, using Lemma 9, $p \to^* q$ as desired. If C has depth $d+1$, then $\mathsf{C} \equiv \nu\widetilde{z}(S \mid !x(a).\mathsf{C}')$ for some process S and context C' of depth d. We therefore have $q \equiv (s \mid q_1 \mid \cdots \mid q_n)\sigma$ with $s \sqsubset S$ and $q_i \sqsubset \mathsf{C}'\{R'\}\{a_i/a\}$ for all $1 \le i \le n$. By hypothesis and Lemma 9, we have $\mathsf{C}'\{R\}\{a_i/a\} \to \mathsf{C}'\{R'\}\{a_i/a\}$, so we apply the induction hypothesis to each q_i and obtain $p_i \sqsubset \mathsf{C}'\{R\}\{a_i/a\}$ such that $p_i \to^* q_i$. Then, if we let $p := (s \mid p_1 \mid \cdots \mid p_n)\sigma$, we have $p \sqsubset P$ such that $p \to^* q$, and we conclude.

Finally, we prove the general version of the lemma by induction on the length of the reduction $P \to^* Q$. The statement is trivial for length zero because $P \equiv Q$. For length $k+1$, we have $P \to P_1 \to^* Q$ with $P_1 \to^* Q$ of length k. Given $q \sqsubset Q$, the induction hypothesis gives us $p_1 \sqsubset P_1$ such that $p_1 \to^* q$. Then, we apply the one-step case proved above to p_1 and obtain the desired $p \sqsubset P$ such that $p \to^* p_1 \to^* q$. ◀

▶ **Lemma 14.** *Let $p \sqsubset P$ such that p has a cut-free form and $p \to p'$ (one-step reduction). Then, there exist reductions $p' \to^* q$ and $P \to Q$ such that $q \sqsubset Q$. Diagrammatically:*

$$\begin{array}{ccc} P & & P \longrightarrow Q \\ \sqcup & \Longrightarrow & \sqcup \quad\quad \sqcup \\ p \longrightarrow p' & & p \longrightarrow p' \longrightarrow^* q \end{array}$$

Proof. The proof is by induction on the structure of P. We only include the complicated cases, which are when P is a parallel composition, a box or a restriction. If $P = P_1 \mid P_2$, then we know that $p = p_1 \mid p_2$ with $p_i \sqsubset P_i$. The reduction may be performed within one of the p_i, in which case we use the induction hypothesis to conclude, or it may be a reduction between an output in, say, p_1 and an input in p_2. By Fig. 1, this means that P_1 is of the form $\overline{a}\langle\widetilde{b}\rangle \mid P_1'$ or $\overline{a}(x)P_1'$, and P_2 of the dual form, so P may perform a matching reduction.

If $P = !x(a).P'$, then $p = p_1 \mid \cdots \mid p_n$ with $p_i \sqsubset P'\{a_i/a\}$. We claim that the step $p \to p'$ cannot happen because of a communication between two distinct p_i. If this were the case, then we would have a free name b occurring as input subject in some p_i. Now, we cannot have $b = a_i$, because a_i does not appear in any other p_j with $j \neq i$. Then, since boxes may have no free linear name, b must approximate a free cartesian name of P', but this is impossible, because free cartesian names of boxes may only appear as outputs, and approximations of outputs are still outputs. So $p_j \to p_j'$ for some j, and $p' = p_1 \mid \cdots \mid p_j' \mid \cdots \mid p_n$. Since approximations do not introduce cuts, the cut reduced in $p_j \to p_j'$ comes from a cut of P'. Such a cut induces a cut in *every* p_i, because these all approximate P'. But p has a cut-free form, so all these cuts are reducible, therefore $p_i \to p_i'$ for *all* i. The induction hypothesis gives us $p_i' \to^* q_i$ and $P' \to Q_i'$ such that $q_i \sqsubset Q_i'$. However, since it is the same cut of P' that is being reduced in each $P' \to Q_i'$, all Q_i' are actually equal to some Q', so $!x(a).P \to^* !x(a).Q'$ and $q_1 \mid \cdots \mid q_n \sqsubset !x(a).Q'$, as desired.

If $P = \nu x P_1$, then $p = p_1\{\widetilde{a}/\widetilde{b}\}$ with $p_1 \sqsubset P_1$ and $\widetilde{a}, \widetilde{b}$ approximate x as output and input, respectively. In case the reduction $p \to p'$ is already present in p_1, *i.e*, in case $p' = p_1'\{\widetilde{a}/\widetilde{b}\}$

with $p_1 \to p_1'$, the induction hypothesis allows us to conclude immediately. Otherwise, the reduction is made possible by the substitution of a_i to b_i for some i, which means that there is an occurrence of x as input matching an occurrence of x as output in P_1. Such occurrences are uniquely determined by the index i and induce a one-step reduction in P. More precisely, we have $P = \nu x \mathsf{C}\{R\}$ and $R \to_{\mathsf{x}} R'$ where x is one of cod_0, $\mathsf{cod}_!$ or c. At this point, the proof splits in two cases, according to the shape of C. If C is shallow (*i.e*, the hole is not under a box), then we conclude straightforwardly (we omit the details). In case the hole is under a box, we use the same argument given above (for the case $P = !x(a).P'$) to conclude that the step $p \to p'$ may be "completed" by reducing the other cuts in p approximating the cut corresponding to $R \to_{\mathsf{x}} R'$, yielding the desired push-forward. ◀

▶ **Lemma 15** (push-forward). *Let $p \sqsubset P$ and $p \to^* n$ such that n is cut-free. Then, there exists a reduction $P \to^* Q$ such that $n \sqsubset Q$. Diagrammatically:*

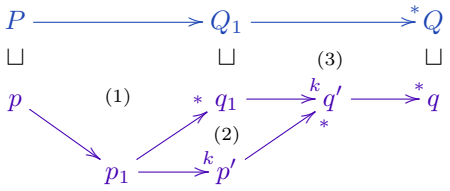
$$\begin{array}{ccc} P & & P \longrightarrow^* Q \\ \sqcup & \Longrightarrow & \sqcup \quad\ \sqcup \\ p \longrightarrow^* n & & p \longrightarrow^* n \end{array}$$

Proof. We prove the following stronger result, which is a generalization of Lemma 14 to reductions of arbitrary length: for all $p \sqsubset P$ with p having a cut-free form and for all $p \to^* p'$, there exist $p' \to^* q$ and $P \to^* Q$ such that $q \sqsubset Q$. Diagrammatically:

$$\begin{array}{ccc} P & & P \longrightarrow^* Q \\ \sqcup & \Longrightarrow & \sqcup \quad\ \sqcup \\ p \longrightarrow^* p' & & p \longrightarrow^* p' \longrightarrow^* q \end{array}$$

The lemma is the special case in which p' is cut-free, which implies $q = p'$.

The proof is by induction on the length of the reduction $p \to^* p'$. If the length is zero, then p is itself cut-free and the claim is immediate. Now suppose that the length is $k+1$, which means $p \to p_1 \to^k p'$, with $\to^k$ denoting reduction in k steps. Diagrammatically, the proof may be depicted as follows:



where (1) holds by Lemma 14, (2) by Lemma 11 (strong confluence) and (3) by the induction hypothesis. Let us spell out the details. By Lemma 14, there exist reductions $p_1 \to^* q_1$ and $P \to^* Q_1$ such that $q_1 \sqsubset Q_1$. Now, it is well known that, in a strongly confluent system, the two sides of confluence diamonds have the same length, that is, if $t \to^m t'$ and $t \to^n t''$, there there exists u such that $t' \to^n u$ and $t'' \to^m u$. Therefore, by Lemma 11, there exists q' such that $p' \to^* q'$ and $q_1 \to^k q'$. But q_1 still has a cut-free form, because it is a reduct of p, so the induction hypothesis allows us to conclude. ◀

Lemma 15 is why "special" inputs and outputs are needed in Taylor approximations. If, for instance, $\overline{a}(x)P$ where approximated by $\overline{a}\langle\widetilde{b}\rangle \mid p$ rather than $\overline{a}\langle\!\langle\widetilde{b}\rangle\!\rangle \mid p$, then the push-forward property would fail: for example, $\overline{a}(x) \mid a\langle\rangle$ cannot reduce, but would be approximated by $\overline{a}\langle\rangle \mid a\langle\rangle$, which reduces to $\mathbf{0}$. This is related to the failure of the forgetful map $\mathcal{T}ay \to \mathcal{L}in\mathcal{P}roc$ to be an embedding, as pointed out above.

4.2 Böhm Approximations and the Commutation Theorem

▶ **Definition 16** (Böhm approximations). *We define $P \leq_0 Q$ just if P is obtained from Q by replacing any number of its boxes (i.e, subprocesses of the form $!x(a).R$) with $\mathbf{0}$. The* Böhm approximation relation *is defined as follows: $N < P$ iff N is cut-free and there exists P' such that $P \to^* P'$ and $N \leq_0 P'$.*

Observe that Definition 16 applies to Taylor processes as well. However, in that case, $\leq_0$ degenerates to equality, because there are no boxes in Taylor processes, so $n < p$ simply means that n is the cut-free form of p (not all Taylor processes have one).

▶ **Lemma 17.** *For every cut-free Taylor process n, $n \sqsubset P$ iff there exists a cut-free process N such that $n \sqsubset N \leq_0 P$.*

Proof. Formally, both directions are by induction on P. Rather than detailing all cases, let us give the intuition. For the forward direction, the key point is that the only way Taylor approximations may "forget" cuts is by approximating a box with $\mathbf{0}$. So, if $n \sqsubset P$ with n cut-free, either P is already cut-free, and we are done, or all of its cuts are within boxes approximated by $\mathbf{0}$ in n. Therefore, we define N as P in which those same boxes are replaced with $\mathbf{0}$, and obtain N cut-free such that $n \sqsubset N \leq_0 P$.

The converse does not even rely on cut-freeness: we have $n \sqsubset N$ and N is obtained from P by replacing some boxes with $\mathbf{0}$, but $\mathbf{0}$ is a Taylor approximation of any box, so $n \sqsubset P$. ◀

▶ **Theorem 18** (Böhm-Taylor commutation). *The relations $\sqsubset<$ and $<\sqsubset$ coincide.*

Proof. The proof is in the following diagram:

$$\begin{array}{ccccc} P & \xrightarrow{*} & Q & \geq_0 & N \\ \sqcup & (1) & \sqcup & (2) & \sqcup \\ p & \xrightarrow{*} & n & = & n \end{array}$$

Indeed, observe that, by definition, $n \sqsubset< P$ is equivalent to the situation depicted in the red part of the diagram, for some Q and cut-free N. Similarly, $n <\sqsubset P$ is equivalent to the situation depicted in the green part of the diagram, for some p.

For red implies green, notice that Taylor approximations do not introduce cuts, so N cut-free implies n cut-free. Then, Lemma 17 gives us square (2), and Lemma 13 gives us square (1).

For green implies red, square (1) is given by Lemma 15 and square (2) by Lemma 17. ◀

4.3 Pulling Back the Commutation Theorem

In what follows, we fix an arbitrary reduction system $\mathcal{S}$ equipped with an embedding $f : \mathcal{S} \to \mathcal{P}roc$ (in the technical sense of Definition 1).

▶ **Definition 19** (Böhm tree). *Let s be an objects of $\mathcal{S}$. A* Böhm approximation *of s is a cut-free process N such that $s \to^* s'$ and $N \leq_0 f(s')$. The* Böhm tree *of s, denoted by* $\mathrm{BT}(s)$*, is the set of all Böhm approximations of s.*

Observe that we could have defined Böhm approximations by asking that $f(s) \to^* f(s')$ and $N \leq_0 f(s')$. We now prove that this is equivalent, because f is an embedding.

▶ **Lemma 20.** *For every process P and cut-free process N, $N \leq_0 P$ and $P \to^* P'$ implies $N \leq_0 P'$.*

Proof. Since N is cut-free, and since N differs from P only by the fact that some boxes of P are replaced by $\mathbf{0}$ in N, then any cut of P is inside a box. Therefore, P' differs from P only inside some boxes, and replacing them with $\mathbf{0}$ yields again N. ◀

▶ **Lemma 21.** *For every object s of $\mathcal{S}$, N is a Böhm approximation of s iff $f(s) \to^* f(s')$ such that $N \leq_0 f(s')$.*

Proof. The forward reduction is immediate: by definition, we have a reduction $s \to^* s'$ in $\mathcal{S}$ such that $N \leq_0 f(s')$, so we take the reduction $f(s) \to^* f(s')$ which we have by monotonicity. For the converse, suppose that $f(s) \to^* f(s')$. By definition of embedding, there exists s'' such that $f(s') \to^* f(s'')$ and $s \to^* s''$. But by Lemma 20, $N \leq_0 f(s')$ implies $N \leq_0 f(s'')$, so we conclude. ◀

The terminology "Böhm tree" is somewhat abusive because, strictly speaking, $\mathrm{BT}(s)$ has nothing of a tree. In good cases, however, it does have the properties of Böhm trees. First of all, it follows immediately from the definition that $s \to^* s'$ implies $\mathrm{BT}(s') \subseteq \mathrm{BT}(s)$. In case $\mathcal{S}$ is confluent, *i.e*, if any span of the form $s \to^* s_1$, $s \to^* s_2$ may be closed by reductions $s_1 \to^* s'$ and $s_2 \to^* s'$, then the converse implication also holds:

▶ **Proposition 22.** *Let $\mathcal{S}$ be confluent. Then, $s \to^* s'$ implies $\mathrm{BT}(s) = \mathrm{BT}(s')$.*

Proof. We only need to show that $\mathrm{BT}(s) \subseteq \mathrm{BT}(s')$. Let $N \in \mathrm{BT}(s)$. By definition, $s \to^* s_1$ such that $N \leq_0 f(s_1)$. By confluence, there exists s'' such that $s' \to^* s''$ and $s_1 \to^* s''$, so we conclude by Lemma 20. ◀

Additionally, when $\mathcal{S}$ is confluent, $\mathrm{BT}(s)$ may be seen as a possibly infinite cut-free process, in the sense of taking the construct $!x(a).P$ coinductively:

▶ **Lemma 23.** *Let $\mathcal{S}$ be confluent. Then, $\mathrm{BT}(s)$ is either empty or an ideal w.r.t. $\leq_0$.*

Proof. Downward closure is immediate from the definition and transitivity of $\leq_0$. It remains to prove that $N_1, N_2 \in \mathrm{BT}(s)$ implies that there is $N \in \mathrm{BT}(s)$ such that $N_1, N_2 \leq_0 N$. We start by proving that, given an arbitrary process P, the poset $\{Q \mid Q \leq_0 P\}$ ordered by $\leq_0$ has binary suprema. Let τ_P be the rooted forest whose nodes are the boxes of P and such that there is an edge from R to S if S is a subprocess of R. By definition, $Q \leq_0 P$ iff Q is obtained by replacing some boxes of P with $\mathbf{0}$, so the poset $\{Q \mid Q \leq_0 P\}$ is isomorphic to the poset $\{\tau \mid \tau \text{ is a rooted subforest of } \tau_P\}$ ordered by rooted inclusion of forests, and the latter obviously has binary suprema.

Now, by definition, $N_1, N_2 \in \mathrm{BT}(s)$ means that $s \to^* s_1$ and $s \to^* s_2$ such that $N_1 \leq_0 f(s_1)$ and $N_2 \leq_0 f(s_2)$. By confluence, we have s' such that $s_1 \to^* s'$ and $s_2 \to^* s'$. By Lemma 20 and the fact that f is a morphism, $N_1, N_2 \leq_0 f(s')$, so we take N to be the supremum of N_1 and N_2. ◀

▶ **Definition 24** (Taylor expansion)**.** *Let s be an object of $\mathcal{S}$. A* Taylor approximation *of s is a Taylor process p such that $p \sqsubset f(s)$. The* Taylor expansion *of s, denoted by $\mathcal{T}(s)$, is the set of all Taylor approximations of s.*

The Taylor expansion of the Böhm tree *of s is the following set of Taylor processes:*

$$\mathcal{T}(\mathrm{BT}(s)) := \{n \sqsubset N \mid N \in \mathrm{BT}(s)\}.$$

Notice that, since Böhm approximations are cut-free, $\mathcal{T}(\mathrm{BT}(s))$ is actually a set of cut-free linear processes.

$$\frac{}{\vdash ; ; A^\perp, A} \qquad \frac{\vdash \Theta_1; \Theta_1'; \Gamma, A^\perp \quad \vdash \Theta_2; \Theta_2'; \Delta, A}{\vdash \Theta_1, \Theta_2; \Theta_1', \Theta_2'; \Gamma, \Delta} \qquad \frac{\vdash \Theta_1, A^\perp, \dots, A^\perp; \Theta_1'; \Gamma \quad \vdash \Theta_2; \Theta_2', A; \Delta}{\vdash \Theta_1, \Theta_2; \Theta_1', \Theta_2'; \Gamma, \Delta}$$

$$\frac{\vdash \Theta_1; \Theta_1'; \Gamma_1, A_1 \quad \dots \quad \vdash \Theta_1; \Theta_n'; \Gamma_n, A_n}{\vdash \Theta_1, \dots, \Theta_n; \Theta_1', \dots, \Theta_n'; \Gamma_1, \dots, \Gamma_n, A_1 \otimes \dots \otimes A_n} \qquad \frac{\vdash \Theta; \Theta'; \Gamma, A_1, \dots, A_n}{\vdash \Theta; \Theta'; \Gamma, A_1 \parr \dots \parr A_n}$$

$$\frac{\vdash \Theta; \Theta'; \Gamma, A}{\vdash \Theta, A; \Theta'; \Gamma} \qquad \frac{\vdash \Theta, A, \dots, A; \Theta'; \Gamma}{\vdash \Theta; \Theta'; \Gamma, ?A} \qquad \frac{\vdash \Theta; ; A}{\vdash \Theta; A;} \qquad \frac{\vdash \Theta; \Theta', A; \Gamma}{\vdash \Theta; \Theta'; \Gamma, !A}$$

Figure 2 Linear logic. Exchange rules (applicable within every segment of sequents) are implicit.

Given an arbitrary set of Taylor processes X, we define $\mathrm{NF}(X)$ as the set of cut-free forms of processes in X: $\mathrm{NF}(X) := \{n \text{ cut-free} \mid \exists p \in X \text{ such that } p \to^* n\}$.

▶ **Theorem 25** (Böhm-Taylor commutation, pulled back)**.** *For every object s of $\mathcal{S}$,*

$$\mathrm{NF}(\mathcal{T}(s)) = \mathcal{T}(\mathrm{BT}(s)).$$

Proof. Unfolding the definitions, we have $n \in \mathrm{NF}(\mathcal{T}(s))$ iff $n <\sqsubset f(s)$. Similarly, $n \in \mathcal{T}(\mathrm{BT}(s))$ iff $n \sqsubset< f(s)$: the forward implication is immediate from the definitions and the fact that f is a morphism; the converse follows from Lemma 21. We then conclude by Theorem 18. ◀

5 Linear Logic

5.1 Proofs as Processes

The sequent calculus of classical linear logic is presented in Fig. 2. We leave out additive connectives as they are unessential for our purposes. Sequents are divided in three segments, conveniently matching our process calculus. It is an exercise to show that this is equivalent to a more standard presentation, such as Girard's original one [27], in the following sense:

▶ **Proposition 26.** *A sequent $\vdash \Theta; \Theta'; \Gamma$ is provable in the sequent calculus of Fig. 2 iff the sequent $\vdash ?\Theta, !\Theta', \Gamma$ is provable in the sequent calculus of classical linear logic given in [27].*

Proof. Observe that the translation of each rule of Fig. 2 is derivable in Girard's calculus. Conversely, every rule of Girard's calculus is derivable in ours. Deriving contraction and promotion introduces cuts, using the derivability of $\vdash A; ; !A^\perp$. ◀

The sequent calculus of Fig. 2 may be decorated with processes and converted to a type system. This would be a "Curry-style" presentation of the correspondence between our processes and classical linear logic. We opt instead for a "Church-style" presentation:

▶ **Definition 27** (typed process)**.** *A* typed process *is a process in which every occurrence of name (except in binders) is decorated by a formula of linear logic, in such a way that:*

- *in $a^A \leftrightarrow b^B$, $B = A^\perp$;*
- *two occurrences of the same linear name are decorated by the same formula if one is subject and the other object, or by dual formulas if they are both subject or both object;*
- *in $\overline{a}^A\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle$ (resp. $a^A\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle$), $A = B_1 \otimes \dots \otimes B_n$ (resp. $A = B_1 \parr \dots \parr B_n$);*

$$\frac{a \neq b}{a \leftrightarrow b \rhd \Xi; ; a, b} \qquad \frac{P \rhd \Xi; \Xi'_1; \Gamma, a \quad Q \rhd \Xi; \Xi'_2; \Delta, a}{P \mid Q \rhd \Xi; \Xi'_1, \Xi'_2; \Gamma, \Delta}$$

$$\frac{}{\overline{a}\langle \widetilde{b} \rangle \rhd \Xi; ; a, \widetilde{b}} \qquad \frac{P \rhd \Xi; \Xi'; \Gamma, \widetilde{b}}{a\langle \widetilde{b} \rangle \mid P \rhd \Xi; \Xi'; \Gamma, a} \qquad \frac{P \rhd \Xi, x; \Xi'_1; \Gamma \quad Q \rhd \Xi; \Xi'_2, x; \Delta}{\nu x(P \mid Q) \rhd \Xi; \Xi'_1, \Xi'_2; \Gamma, \Delta}$$

$$\frac{}{\overline{x}\langle a \rangle \rhd \Xi, x; ; a} \qquad \frac{P \rhd \Xi, x; \Xi'; \Gamma}{\overline{a}(x)P \rhd \Xi; \Xi'; \Gamma, a} \qquad \frac{P \rhd \Xi; ; a}{!x(a).P \rhd \Xi; x;} \qquad \frac{P \rhd \Xi; \Xi', x; \Gamma}{a(x)P \rhd \Xi; \Xi'; \Gamma, a}$$

Figure 3 Correct processes.

- *all occurrences of same polarity of a cartesian name are decorated by the same formula;*
- *in νxP, positive and negative occurrences of x (if any) are decorated by dual formulas;*
- *in $\overline{a}^A(x)P$ (resp. $a^A(x)P$), $A = ?B$ (resp. $A = !B$), where B is the formula decorating x in P, or is arbitrary if $x \notin \mathsf{fn}(P)$;*
- *in $\overline{x}^A\langle a^B \rangle$ and $x^A\langle a^B \rangle$, we have $A = B$; similarly, in $!x^A(a).P$, the decoration of a in P (which must appear) is A.*

The type *of a free occurrence of name as subject (resp. as object) is its decoration (resp. the negation of its decoration). We say that $\vdash \Theta; \Theta'; \Gamma$ is a* sequent associated with *P if Θ (resp. Θ', Γ) contains all types of all positive cartesian (resp. negative cartesian, linear) free occurrences of variables. This is unique up to a permutation of the occurrences, so in the sequel we will simply speak of "the" sequent associated with P.*

It is important to observe that typed does not imply logically correct: for example, $a^A \leftrightarrow a^{A^\perp}$ is typed, but its associated sequent is empty. In the literature on linear logic, there are *correctness criteria* [27, 13] for isolating "proof-like" objects. Here, we rely on an inductive presentation, given in Fig. 3, where a judgment $P \rhd \Xi; \Xi'; \Gamma$ means that P is correct and its free positive cartesian (resp. negative cartesian, linear) names are in Ξ (resp. Ξ', Γ). Observe that, in accordance with the yoga of linear logic, typability and correctness are independent notions: a correct process need not be typable.

▶ **Proposition 28.** *A sequent is provable in the sequent calculus of Fig. 2 iff it is associated with a correct typed process. Moreover, cut-free proofs correspond to cut-free processes.*

Proof. Both directions are by induction, on the last rule of the sequent calculus proof or on the last rule of the correctness derivation. ◀

So correct typed processes may be regarded as linear logic proofs, and reduction as a cut-elimination procedure. If one wished, confluence and termination could be proved for correct typed processes, but this is beyond the scope of this paper. We do observe, however, that Fig. 3 may easily be extended to include correctness for non-deterministic processes. It suffices to replace the last rule of Fig. 3 with the rule below on the left, and to add the rule below on the right:

$$\frac{P_1 \rhd \Xi; \Xi'_1, x; \Gamma_1 \quad \dots \quad P_n \rhd \Xi; \Xi'_n, x; \Gamma_n}{a(x)(P_1 \mid \cdots \mid P_n) \rhd \Xi; \Xi'_1, \dots, \Xi'_n; \Gamma_1, \dots, \Gamma_n, a} \qquad \frac{}{x\langle a \rangle \rhd \Xi; x; a}$$

Correct (for this extended notion of correctness) typed processes are in correspondence with proofs of differential linear logic (with promotion but without zero and sums).

$$\frac{}{\mathbf{0} \vdash ; ;} \qquad \frac{P \vdash \Xi_1; \Xi_1'; \Gamma, [a : A] \quad Q \vdash \Xi_2; \Xi_2'; \Delta, [a : A^\perp]}{P \mid Q \vdash \Xi_1, \Xi_2; \Xi_1', \Xi_2'; \Gamma, \Delta} \qquad \frac{}{a \leftrightarrow b \vdash ; ; a : A, b : A^\perp}$$

$$\frac{}{\overline{a}\langle b_1, \ldots, b_n \rangle \vdash ; ; a : \bigotimes_i A_i, b_1 : A_1^\perp \ldots, b_n : A_n^\perp}$$

$$\frac{}{a\langle b_1, \ldots, b_n \rangle \vdash ; ; a : \parr_i A_i, b_1 : A_1^\perp \ldots, b_n : A_n^\perp}$$

$$\frac{P \vdash \Xi_1, x : A_1, \ldots, x : A_n; \Xi_1'; \Gamma \quad Q \vdash \Xi_2; \Xi_2', x : A_1^\perp, \ldots, x : A_n^\perp; \Delta}{\nu x(P \mid Q) \vdash \Xi_1, \Xi_2; \Xi_1', \Xi_2'; \Gamma, \Delta}$$

$$\frac{P \vdash \Xi, x : A_1, \ldots, x : A_n; \Xi'; \Gamma}{\overline{a}(x)P \vdash \Xi; \Xi'; \Gamma, a : \bigvee_i A_i}$$

$$\frac{P_1 \vdash \Xi_1; \Xi_1', x : A_1; \Gamma_1 \quad \ldots \quad P_n \vdash \Xi_n; \Xi_n', x : A_n; \Gamma_n}{a(x)(P_1 \mid \cdots \mid P_n) \vdash \Xi_1, \ldots, \Xi_n; \Xi_1', \ldots, \Xi_n'; \Gamma_1, \ldots, \Gamma_n, a : \bigwedge_i A_i}$$

$$\frac{}{\overline{x}\langle a \rangle \vdash x : A; ; a : A^\perp} \qquad \frac{}{x\langle a \rangle \vdash ; x : A; a : A^\perp}$$

$$\frac{P \vdash \Xi_1; ; a : A_1 \quad \ldots \quad P \vdash \Xi_n; ; a : A_n}{!x(a).P \vdash \Xi_1, \ldots, \Xi_n; x : A_1, \ldots, x : A_n;}$$

Figure 4 Intersection types. In the second rule at the top, the declarations $a : A$ and $a : A^\perp$ are either both present or both absent (the rule is a cut in the first case, a mix in the second).

One may also add correctness of the so-called *mix rules*:

$$\frac{}{\mathbf{0} \rhd ; ;} \qquad \frac{P \rhd \Xi; \Xi_1'; \Gamma \quad Q \rhd \Xi; \Xi_2'; \Delta}{P \mid Q \rhd \Xi; \Xi_1', \Xi_2'; \Gamma, \Delta}$$

This more general notion of correctness, for non-deterministic processes and with mix rules, is the one we will consider in the next section.

5.2 Intersection Types

Intersection types are defined as follows:

$$A, B ::= X \mid X^\perp \mid A_1 \otimes \cdots \otimes A_n \mid A_1 \parr \cdots \parr A_n \mid A_1 \wedge \cdots \wedge A_n \mid A_1 \vee \cdots \vee A_n.$$

Duality (*i.e*, linear negation) is defined as usual, with $\wedge$ dual to $\vee$.

An *intersection type judgment* is of the form $P \vdash \Xi; \Xi'; \Gamma$ where Ξ and Ξ' (resp. Γ) contain type declarations of the form $x : A$ (resp. $a : A$) with x a cartesian name (resp. a a linear name) and A an intersection type. The same cartesian name may appear in multiple type declarations in Ξ and Ξ', even multiple times with the same type (this means that we are considering non-idempotent intersection types). By contrast, a linear name may only be declared once in Γ. The intersection type system for processes is given in Fig. 4.

A *typed Taylor process* is defined as in Definition 27, with the addition of the constraints that in $\overline{a}^A\langle\!\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle\!\rangle$ (resp. $a^A\langle\!\langle b_1^{B_1}, \dots, b_n^{B_n}\rangle\!\rangle$) we have $A = B_1 \vee \cdots \vee B_n$ (resp. $A = B_1 \wedge \cdots \wedge B_n$). Taylor processes do not have cartesian names, so the associated sequent of a typed Taylor process is of the form $\vdash;;\Gamma$, which we just write $\vdash \Gamma$. If p is a typed Taylor process, we write p^- for the underlying process, without the decorations. In the following, correctness is meant in the generalized sense of the end of §5.1 (with mix).

▶ **Lemma 29.** *If p is a correct typed Taylor process, then p^- has a cut-free form.*

Proof. Taylor processes are linear, so reduction always terminates (Lemma 11). Therefore, it is enough to show that there are no irreducible cuts. This is straightforward: irreducible cuts are incorrect or not typable and reduction preserves correctness and type decorations. ◀

The following is a reformulation of the results of [32], where a general connection between Taylor approximations and intersection types is described in detail. We state it without proof, but one may intuitively justify the result by observing that Fig. 4 is a superposition of Fig. 1 and Fig. 3 (with the additional rules given at the end of §5.1), annotated with types.

▶ **Proposition 30.** *The judgment $P \vdash \Xi; \Xi'; \Gamma$ is derivable in the system of Fig. 4 iff there exists a correct typed Taylor process p whose associated sequent is $\vdash \Xi, \Xi', \Gamma$ such that $p^- \sqsubset P$.*

▶ **Lemma 31** (subject expansion)**.** *If $Q \vdash \Xi; \Xi'; \Gamma$ is derivable and P is correct such that $P \to Q$, then $P \vdash \Xi; \Xi'; \Gamma$ is derivable.*

Proof. Modulo Proposition 30, this is Lemma 13, except that we need to add type decorations. Correctness of P ensures that the pullback approximation is also correct. ◀

We say that a context is *shallow* if the hole does not appear inside a box. *Shallow reduction* is defined by modifying Definition 7 as follows: the reduction rule $\mathsf{cod}_!$ is discarded, c is restricted to the case $\mathsf{C} = Q \mid \{-\}$, with Q arbitrary, and the reduction rules are only closed under shallow contexts. We write $\to_0$ for shallow reduction. A process is said to be *shallow cut-free* if all of its cuts, if any, are inside boxes.

▶ **Lemma 32** (progress)**.** *Let $P \vdash \Xi; \Xi'; \Gamma$ be derivable and let p be the associated typed Taylor process according to Proposition 30. Then, either P is shallow cut-free or $P \to_0^* Q$ and there is a derivation $Q \vdash \Xi; \Xi'; \Gamma$ whose associated Taylor process q is such that $p^- \to q^-$.*

Proof. By inspection of Fig. 1, we see that a shallow cut in P yields a cut in p. As observed in Lemma 29, such a cut cannot be irreducible, so we have $p \to q$ by reducing it. Removing type annotations, we have $p^- \sqsubset P$ and $p^- \to q^-$, so we apply Lemma 15 and Proposition 30. ◀

Observe that Lemma 32 does not hold for general cuts: since boxes may be approximated by $\mathbf{0}$, P may contain a cut inside a box which is invisible to the intersection type derivation.

▶ **Theorem 33.** *A process P is typable as in Fig. 4 iff it is correct (in the generalized sense of the end of §5.1) and $P \to_0^* P_0$ with P_0 shallow cut-free.*

Proof. Let P be typable. Correctness is immediate from the typing rules: they are essentially a decoration of Fig. 3 plus the extra correctness rules at the end of §5.1. We need to show that P reduces to a shallow cut-free process. Let p be the typed Taylor approximation given by Proposition 30. We reason by induction on the size of p^-, as defined in the proof of Lemma 11. We apply Lemma 32 and either conclude immediately because P is shallow cut-free or obtain $P \to^* Q$ with Q typable with an associated approximation q such that

$p^- \to q^-$. This implies that the size of q^- is strictly smaller than the size of p^-, so we conclude by the induction hypothesis.

Suppose now that $P \to_0^* P_0$ with P_0 shallow cut-free. It is straighforward to prove, by induction on P_0, that P_0 is typable (intuitively, we approximate all boxes by 0 and then typability is guaranteed by correctness and absence of cuts), so we conclude by Lemma 31. ◀

If $f : \mathcal{S} \to \mathcal{P}roc$ is a morphism of reduction systems, then we may say that an object of s is *typable in intersection types* if $f(s)$ is typable according to Fig. 4.

Let $\mathcal{P}roc_0$ be the reduction system with processes as objects but with $\to_0^*$ as preorder. In the following, we consider a reduction system $\mathcal{S}$ with a distinguished set of objects suggestively called "normal". We say that an embedding $f : \mathcal{S} \to \mathcal{P}roc_0$ is *sound* if

- for every object s, $f(s)$ is correct (in the generalized sense);
- for every object s_0, s_0 is normal iff $f(s_0)$ is shallow cut-free.

▶ **Theorem 34.** *Let $f : \mathcal{S} \to \mathcal{P}roc_0$ be a sound embedding. Then, an object s of $\mathcal{S}$ is typable in intersection types iff $s \to^* s_0$ with s_0 normal.*

Proof. Suppose that s is typable, which means that $f(s)$ is. By Theorem 33, $f(s) \to_0^* P_0$ with P_0 shallow cut-free. Since f is an embedding, we have $P_0 \to_0^* f(s_0)$ such that $s \to^* s_0$. But P_0 is shallow cut-free, so $f(s_0) = P_0$ and we conclude s_0 normal by soundness.

Suppose now that $s \to^* s_0$ with s_0 normal. This implies $f(s) \to_0^* f(s_0)$. By soundness, $f(s_0)$ is correct and shallow cut-free, so it is typable. By soundness and Lemma 31, $f(s)$ is typable, so s is typable by definition. ◀

6 Applications

6.1 Call by Push Value

It is known that Paul Levy's call-by-push-value [30] may be expressed in intuitionistic linear logic [19], yielding the *bang-calculus* [21]. We use a recent reformulation due to Bucciarelli et al. [8], which allows us to show at the same time how Accattoli and Kesner's explicit substitutions "at a distance" may be handled painlessly. Terms are defined by

$$t, u ::= x \mid \lambda x.t \mid tu \mid !t \mid \operatorname{der} t \mid t[x := u],$$

where x ranges over a countable set of variables, which, for convenience, we take to be the set of cartesian names. Contexts are defined as expected (add a hole $\{-\}$ to the above definition). We use the notation $t[-]$ for a term of the form $t[x_1 := u_1] \cdots [x_n := u_n]$ (n may be zero). The constructor $!(-)$ binds more strongly than binary constructors, *i.e*, $!tu$ and $!t[x := u]$ are to be understood as $(!t)u$ and $(!t)[x := u]$, respectively. Reduction is the contextual closure of the following rules:

$$(\lambda x.t)[-]u \;\to\; t[x := u][-] \qquad t[x := !u[-]] \;\to\; t\{u/x\}[-] \qquad \operatorname{der}(!t[-]) \;\to\; t[-]$$

where $t\{u/x\}$ denotes, as usual, the capture-free substitution of u to all free occurrences of x in t. This induces a reduction system $\Lambda_!$. We inductively define a family of maps $(\!|-|\!)a : \Lambda_! \to \mathcal{P}roc$ parametric in a linear name a:

$$\begin{aligned}
(\!|x|\!)a &:= \overline{x}\langle a\rangle & (\!|\lambda x.t|\!)a &:= a\langle c, d\rangle \mid \overline{c}(x)(\!|t|\!)d \\
(\!|tu|\!)a &:= (\!|t|\!)b \mid \overline{b}\langle c, a\rangle \mid (\!|u|\!)c & (\!|!t|\!)a &:= a(z)!z(b).(\!|t|\!)b \\
(\!|\operatorname{der} t|\!)a &:= \overline{c}(z)\overline{z}\langle a\rangle \mid (\!|t|\!)c & (\!|t[x := u]|\!)a &:= \overline{b}(x)(\!|t|\!)a \mid (\!|u|\!)b.
\end{aligned}$$

The paper [8] also introduces *weak reduction* for the bang-calculus, which does not reduce under $!(-)$. We denote by $\Lambda_!^w$ the corresponding reduction system. In this system, we take normal forms to be terms whose redexes and clashes (undesirable configurations defined in [8]) only appear under a $!(-)$.

▶ **Proposition 35.** *For any linear name a, $(\!|-|\!)a$ is an embedding. Moreover, considered as a map $\Lambda_!^w \to \mathcal{P}roc_0$, it is a sound embedding.*

With a bit of work (which we spare the reader), the results of §4.3 may be formulated directly in the syntax of the bang-calculus. Böhm trees are as expected: given a bang-calculus term t, if weak reduction for t does not terminate, then $\mathrm{BT}(t) = \bot$. Otherwise, it terminates on a term of the form $\mathsf{C}\{!u_1, \dots, !u_n\}$ where C is a multi-hole context containing no $!(-)$. Then, $\mathrm{BT}(t) = \mathsf{C}\{!\mathrm{BT}(u_1), \dots, !\mathrm{BT}(u_n)\}$. Notice that $\bot$ and $!\bot$ are different Böhm trees.

The Taylor expansion of call by push value has already been defined and studied in [10]. Here, we find a reformulation in the context of the bang-calculus with explicit substitutions. Taylor approximation terms are defined as follows:

$$r, s ::= a \mid \lambda\langle\widetilde{a}\rangle.r \mid rs \mid \langle r_1, \dots, r_n\rangle \mid \mathrm{der}\, r \mid r[\langle\widetilde{a}\rangle := s],$$

where a ranges over linear variables, *i.e*, no variable occurs twice and every variable of $\widetilde{a}$ in binders $\lambda\langle\widetilde{a}\rangle.t$ or $t[\langle\widetilde{a}\rangle := u]$ must occur free in t. The reduction rules are as follows:

$$(\lambda\langle\widetilde{a}\rangle.r)s \to r[\langle\widetilde{a}\rangle := s] \qquad r[\langle\widetilde{a}\rangle := \langle\widetilde{s}\rangle] \to r\{\widetilde{s}/\widetilde{a}\} \qquad \mathrm{der}\langle r\rangle \to r,$$

with the condition that, in the second rule, $|\widetilde{a}| = |\widetilde{s}|$. The approximation relation is defined using judgments $\Xi \vdash r \sqsubset t$ with Ξ consisting of declarations of the form $a \sqsubset x$, with no linear variable appearing twice in Ξ. The relation is defined as follows:

$$\frac{}{a \sqsubset x \vdash a \sqsubset x} \qquad \frac{\Xi, \widetilde{a} \sqsubset x \vdash r \sqsubset t}{\Xi \vdash \lambda\langle\widetilde{a}\rangle.r \sqsubset \lambda x.t} \qquad \frac{\Xi \vdash r \sqsubset t \quad \Upsilon \vdash s \sqsubset u}{\Xi, \Upsilon \vdash rs \sqsubset tu}$$

$$\frac{\Xi_1 \vdash r_1 \sqsubset t \quad \dots \quad \Xi_n \vdash r_n \sqsubset t}{\Xi_1, \dots, \Xi_n \vdash \langle\widetilde{r}\rangle \sqsubset\, !t} \qquad \frac{\Xi \vdash r \sqsubset t}{\Xi \vdash \mathrm{der}\, r \sqsubset \mathrm{der}\, t} \qquad \frac{\Xi, \widetilde{a} \sqsubset x \vdash r \sqsubset t \quad \Upsilon \vdash s \sqsubset u}{\Xi, \Upsilon \vdash r[\langle\widetilde{a}\rangle := s] \sqsubset t[x := u]}$$

By Proposition 35 and Theorem 25, we know that the above Böhm trees and Taylor approximations interact well.

Proposition 35 also entails the results of §5.2. We will not detail them here, but these allow us to obtain for free the intersection type system of [8], along with the property that it characterizes terms with weakly clash-free normal forms.

Both [10] and [8] consider the well-known embeddings of call-by-name and call-by-value λ-calculus in call by push value, and extrapolate from these embeddings suitable notions of Taylor approximations and intersection type systems for call-by-name and call-by-value, recovering the results of [23, 25, 18, 29] and [26, 14, 15, 18]. In our setting, these are embeddings in the technical sense of Definition 1, and since embeddings compose, we also recover these results in a uniform manner. Additionally, we recover usual Böhm trees [4] and the call-by-value Böhm trees of [29], as well as the related commutation theorems [23, 25, 29].

6.2 Classical Logic

The *stack calculus* [9] is a simple calculus for classical computation, embedding the $\lambda\mu$-calculus. Its syntax has *stacks* π, *terms* t and *processes* P, defined as follows:

$$\pi ::= \alpha \mid t \cdot \pi \mid \mathsf{tl}(\pi) \qquad t ::= \mu\alpha.P \mid \mathsf{hd}(\pi) \qquad P ::= \langle t, \pi\rangle,$$

where α ranges over a countably infinite set of *stack variables*, which, for technical convenience, we take to be a subset of cartesian names of our processes. The construct $\mu\alpha$ is a binder. Reduction is defined as follows:

$$\langle \mu\alpha.P, \pi\rangle \to P\{\pi/\alpha\} \qquad \mathsf{hd}(t\cdot\pi) \to t \qquad \mathsf{tl}(t\cdot\pi) \to \pi.$$

Let $\mathcal{S}tk$, $\mathcal{T}erm$ amd $\mathcal{S}tkProc$ be the reduction systems induced by the above definitions, with stacks, terms and processes as objects, respectively. We define two families of maps $(\!|-|\!)a : \mathcal{S}tk \to \mathcal{P}roc$ and $(\!|-|\!)a : \mathcal{T}erm \to \mathcal{P}roc$ parametric in a linear name a, as well as a map $(\!|-|\!) : \mathcal{S}tkProc \to \mathcal{P}roc$, as follows:

$$\begin{array}{ll}
(\!|\alpha|\!)a := \overline{\alpha}\langle a\rangle & (\!|t\cdot\pi|\!)a := \overline{a}\langle b,c\rangle \mid b(x)!x(d).(\!|t|\!)d \mid c(y)!y(e).(\!|\pi|\!)e \\
(\!|\mu\alpha.P|\!)a := \overline{a}(\alpha)(\!|P|\!) & (\!|\mathsf{hd}(\pi)|\!)a := (\!|\pi|\!)b \mid b\langle c,d\rangle \mid \overline{c}(x)\overline{d}(y)(\overline{x}\langle a\rangle) \\
(\!|\langle t,\pi\rangle|\!) := (\!|t|\!)a \mid a(x)!x(b).(\!|\pi|\!)b & (\!|\mathsf{tl}(\pi)|\!)a := (\!|\pi|\!)b \mid b\langle c,d\rangle \mid \overline{c}(x)\overline{d}(y)(\overline{y}\langle a\rangle).
\end{array}$$

▶ **Proposition 36.** *The above maps are embeddings.*

As mentioned in the introduction, the encoding of the $\lambda\mu$-calculus in the stack calculus is not an embedding in our technical sense, so we cannot directly apply our results to the $\lambda\mu$-calculus. Nevertheless, now we do have a working theory of Böhm trees and Taylor expansion for a calculus Curry-Howard-isomorphic to classical logic, which is a novelty as far as we know (in the context of the $\lambda\mu$-calculus, it is mentioned as an open question in [2]). We leave the investigation of this theory, in particular the significance of Böhm trees, to future work.

6.3 Concurrent Computation

The asynchronous polyadic π-calculus is defined as follows:

$$P, Q ::= \mathbf{0} \;\Big|\; P \mid Q \;\Big|\; \nu x P \;\Big|\; \overline{x}\langle\widetilde{y}\rangle \;\Big|\; !x(\widetilde{y}).P,$$

where we suppose names to be cartesian names of our processes. Structural congruence and reduction are standard, with the rules

$$\overline{x}\langle\widetilde{y}\rangle \mid !x(\widetilde{z}).P \to P\{\widetilde{y}/\widetilde{z}\} \mid !x(\widetilde{z}).P \qquad\qquad \nu x(!x(\widetilde{y}).P \mid Q) \to \nu x Q$$

with the proviso, in the second rule, that x does not occur as subject of an output in Q.

We consider here the *hyperlocalized* variant of the calculus [12], which is defined by restricting to processes such that, in $!x(\widetilde{y}).P$, no free name of P occurs as subject of an input. Also, reduction is allowed only under a restriction. As shown in [12], this is a reasonably expressive calculus, with full non-determinism, locks, etc.

Let Π be the reduction system corresponding to the above calculus. Using the notation $(z \Rightarrow y) := !z(c).\overline{y}\langle c\rangle$ (as a process of $\mathcal{P}roc$), we define a map $(\!|-|\!) : \Pi \to \mathcal{P}roc_0$ by letting

$$\begin{array}{l}
(\!|\overline{x}\langle y_1,\dots,y_n\rangle|\!) := \overline{x}\langle a\rangle \mid \overline{a}\langle b_1,\dots,b_n\rangle \mid b_1(z_1)(z_1 \Rightarrow y_1) \mid \cdots \mid b_n(z_n)(z_n \Rightarrow y_n) \\
(\!|!x(y_1,\dots,y_n).P|\!) := !x(a).(a\langle b_1,\dots,b_n\rangle \mid \overline{b}_1(y_1)\cdots\overline{b}_n(y_n)(\!|P|\!))
\end{array}$$

and by making $(\!|-|\!)$ act homomorphically on $\mathbf{0}$, parallel composition and restriction.

▶ **Proposition 37.** *The map $(\!|-|\!)$ is an embedding.*

Unfortunately, the above embedding is not sound with respect to any reasonable notion of normal form for the π-calculus, because $(\!|P|\!)$ is not necessarily correct (processes may have all sorts of vicious cycles). However, this does not prevent from taking the intersection type system of §5.2 and use it as a starting point for coming up with a working intersection type system for the hyperlocalized π-calculus. This is exactly the genesis of the paper [12].

References

1 Beniamino Accattoli. Exponentials as substitutions and the cost of cut elimination in linear logic. *Log. Methods Comput. Sci.*, 19(4), 2023.

2 Davide Barbarossa. Resource approximation for the $\lambda\mu$-calculus. In *Proceedings of LICS*, pages 27:1–27:12, 2022.

3 Davide Barbarossa and Giulio Manzonetto. Taylor subsumes Scott, Berry, Kahn and Plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020.

4 Henk Barendregt. The type free lambda calculus. In Jon Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 1091–1132. Elsevier, 1977.

5 Henk Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985.

6 Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. Symb. Log.*, 48(4):931–940, 1983.

7 Gérard Boudol. The lambda-calculus with multiplicities (abstract). In *Proceedings of CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 1993.

8 Antonio Bucciarelli, Delia Kesner, Alejandro Ríos, and Andrés Viso. The bang calculus revisited. *Inf. Comput.*, 293:105047, 2023.

9 Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. The stack calculus. In *Proceedings of LSFA*, volume 113 of *EPTCS*, pages 93–108, 2012.

10 Jules Chouquet and Christine Tasson. Taylor expansion for call-by-push-value. In *Proceedings of CSL*, volume 152 of *LIPIcs*, pages 16:1–16:16, 2020.

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

12 Ugo Dal Lago, Marc de Visme, Damiano Mazza, and Akira Yoshimizu. Intersection types and runtime errors in the pi-calculus. *Proceedings of the ACM on Programming Languages*, 3(POPL:7), 2019.

13 Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Log.*, 28(3):181–203, 1989.

14 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Ph.D. thesis, Université de la Méditerranée–Aix-Marseille 2, 2007.

15 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Math. Struct. Comput. Sci.*, 28(7):1169–1203, 2018.

16 Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Math. Struct. Comput. Sci.*, 12(5):579–623, 2002.

17 Thomas Ehrhard. Finiteness spaces. *Math. Struct. Comput. Sci.*, 15(4):615–646, 2005.

18 Thomas Ehrhard. Collapsing non-idempotent intersection types. In *Proceedings of CSL*, volume 16 of *LIPIcs*, pages 259–273, 2012.

19 Thomas Ehrhard. Call-by-push-value from a linear logic point of view. In *Proceedings of ESOP*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228, 2016.

20 Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Math. Struct. Comput. Sci.*, 28(7):995–1060, 2018.

21 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In *Proceedings of PPDP*, pages 174–187, 2016.

22 Thomas Ehrhard and Olivier Laurent. Acyclic solos and differential interaction nets. *Log. Methods Comput. Sci.*, 6(3), 2010.

23 Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine's machine and the Taylor expansion of lambda-terms. In *Proceedings of CiE 2006*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197, 2006.

24 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.

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

26 Philippa Gardner. Discovering needed reductions using type theory. In *Proceedings of TACS*, pages 555–574, 1994.

27 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

28 Kohei Honda and Olivier Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. *Theor. Comput. Sci.*, 411(22-24):2223–2238, 2010.

29 Axel Kerinec, Giulio Manzonetto, and Michele Pagani. Revisiting call-by-value Böhm trees in light of their taylor expansion. *Log. Methods Comput. Sci.*, 16(3), 2020.

30 Paul Blain Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis*, volume 2 of *Semantics Structures in Computation*. Springer, 2004.

31 Damiano Mazza. The true concurrency of differential interaction nets. *Math. Struct. Comput. Sci.*, 28(7):1097–1125, 2018.

32 Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. *Proceedings of the ACM on Programming Languages*, 2(POPL:6), 2018.

33 Michele Pagani and Paolo Tranquilli. The conservation theorem for differential nets. *Math. Struct. Comput. Sci.*, 27(6):939–992, 2017.

34 Paolo Tranquilli. Confluence of pure differential nets with promotion. In *Proceedings of CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 500–514, 2009.

35 Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Generalised species of rigid resource terms. In *Proceedings of LICS*, pages 1–12, 2017.