An Axiomatic Notion of Approximation for Programming Languages and Machines

Damiano Mazza
CNRS, LIPN, Université Sorbonne Paris Nord

1 Introduction

1.1 Axiomatizing Approximations

In the last couple of decades, the idea that a programming language may be approximated by means of a (multi)linear calculus has seen an increasing number of applications. The original, and perhaps best known example is the so-called Taylor expansion of the untyped λ-calculus by means of the resource λ-calculus introduced by Ehrhard and Regnier [ER06, ER08], which has been linked to the theory of non-idempotent intersection types by de Carvalho, who further showed how this may be used to infer exact bounds on the running time of the Krivine abstract machine [dC07, dC18].

The purpose of this note is to propose an axiomatization of the notion of computational approximation, including the multilinear setup mentioned above. The general idea is that an approximation relation $t \sqsubset M$ between an approximating object t and a computational object M should induce a sort of "adjunction" between computations and approximated computations: for every computational object M and approximation object u,

$$\frac{M \text{ evaluates to } N \text{ such that } u \sqsubset N}{\text{there exists } t \sqsubset M \text{ such that } t \text{ evaluates to } u} \text{ iff}$$

or, diagrammatically,

$$\begin{tikzcd} & u & & t \arrow[r] & u \\ & \sqcap & \text{iff} & \sqcap & \\ M \arrow[r] & N & & M & \end{tikzcd}$$

The intuition is simple: if we consider approximations as pieces of information, and read $t \sqsubset M$ as saying that M contains information t, an approximation relation ensures that a computational object M evaluates to something containing a piece of information u iff an approximation of M evaluates to u itself. Notice the resemblance with topological continuity here.

In many cases, such as de Carvalho's results mentioned above, approximations have been used to study quantitative properties of computation. The basic setting above may be endowed with additional structure in order to make quantitative applications possible.

First, we assume that the computational system under consideration has a set of "observable values", which may be thought of as the results of computations. Such observable values must themselves be approximations, and are suitably behaved with respect to the approximation relation (in particular, every value approximates itself).

Second, we assume that there is a set C whose elements may be seen at the same time as "costs" of evaluations in the computational system and as "weights" of approximations: each computation $\rho : M \to N$ has a cost $c_1(\rho) \in C$ and each approximation u has a weight $c_0(u) \in C$. Then, we require that the fundamental "adjunction of approximations relates costs to weights: for every computational object M, observable value v and $c \in C$,

$$\frac{\rho : M \to v \text{ and the cost of } \rho \text{ is } c}{\text{there exists } t \sqsubset M \text{ such that } t \to v \text{ and the weight of } t \text{ is } c} \text{ iff}$$

i.e., the cost of the evaluation of M may be read directly from a suitable approximation of it. The main intuition here is given by Boolean circuits and Turing machines: the former approximate the latter, and the size of a circuit approximating a machine M applied to an input w basically corresponds to the running time of M on w.

1.2 Results and Perspectives

In the primordial example of approximation system (the Taylor expansion), and in many that were introduced afterwards in the literature (such as [Maz17, MPV18, CT20]), what is being approximated is a programming language. In this note, we will show the usefulness of the axiomatic presentation by applying it to the case where the main computational system is not a programming language but an abstract machine for executing a programming language, or even a plain Turing machine. In a way, this is a generalization of what de Carvalho did originally with the Krivine abstract machine.

In particular, we will show that, once suitably instantiated, our axiomatization gives the following results:

Theorem A. *Let M be a closed λ-term. The following are equivalent:*

1. *M evaluates to $*$ in l computational steps on the Krivine abstract machine;*
2. *there exists a linear approximation $t \sqsubset M$ such that $t \to^* *$ and the number l may be read directly from t (it is essentially its size).*

Theorem B. *Let M be a closed λ-term. The following are equivalent:*

1. *M evaluates to $*$ in l steps and using space m (for a suitable measure of space) on the interaction abstract machine;*
2. *there exists a simply-typed linear polyadic $t \sqsubset M$ such that $t \to^* *$ and the numbers l and m may be read directly from t (more precisely, from the types of its subterms).*

A precise formulation of Theorem A and Theorem B is given in Theorem 18 and Theorem 33 below, respectively. In the formal statements, we additionally use the bridge between approximations and intersection types drawn in [MPV18] to rephrase our results in terms of intersection type systems, which is a useful viewpoint.

The above results are only partially new: Theorem A is proved and reproved in [ER06, dC07, ADLV21]. The part of Theorem B concerning the execution length (*i.e.*, the number l) was proved in [ADLV21]. The part of Theorem B concerning space is new, although Accattoli, Dal Lago and Vanoni independently developed a proof of the same result,[1] discovering the idea of using non-associative intersection types (which we call "polyadic types" here). Nevertheless, we think that it is worth reproducing these results under the general perspective of approximation systems.

Additionally, we show how axiomatic approximations instantiate to the already mentioned case of Turing machines and Boolean circuits. From this, we reformulate the proofs of three classic results from complexity theory:

1. NP-completeness of SAT;
2. P-completeness of CIRCUIT VALUE;
3. P has uniform polysize Boolean circuits.

It is well known that these may all be seen as different "avatars" of a single result, which we could call the Cook-Levin theorem, albeit usually one refers only to (1) by that name.

On the negative side, we must say that the current axiomatization is far from being satisfactory: it is admittedly rather trivial and the general theorems (the various versions of Theorem 2) are not very deep at all. It does, however, have the virtue of encompassing many useful cases, as this note will hopefully show.

It is important to emphasize that, although all approximations discussed in this note are linear (in the computational sense of linear logic), our axiomatization of approximation system and the concept of approximation itself is much more general, as showcased by the framework developed in [Maz17, MPV18]. A more thorough understanding of this general notion and of its deep origins is definitely a topic for further investigations, which we feel to be the right path to finding a better, less superficial formulation of approximation system.

Finally, let us stress that, albeit useful, the categorical language we employ is by no means necessary to state and, even more so, to prove the results of this note. At the expense of less conceptual (and perhaps slightly more verbose) definitions, everything we do here may be presented without ever mentioning the words "category" or "functor".

2 Approximation Systems

In many cases, a computational system (machine / calculus / programming language) induces a (directed, multi)graph, whose nodes are states / terms /

[1]An ongoing work at the time of writing.

programs and whose edges are elementary transitions / rewrites / computations. It will be convenient to consider the free category generated by such a graph. So, more generally, let us identify a computational system with a small category.

The reason why a category is better than a graph is that functors are more general than graph homomorphisms, and we need this extra generality for the following notion. Let us define a *protofibration* as a functor between small categories

$$\begin{tikzcd}
\mathcal{E} \arrow[d, "p"] \\
\mathcal{B}
\end{tikzcd}$$

such that, for every object e' of $\mathcal{E}$ and for every arrow $f : b \to p(e')$ of $\mathcal{B}$, there exists an arrow $g : e \to e'$ such that $p(g) = f$ (one usually says that the arrow f is "lifted" to g). Dually, a *proto-opfibration* is a functor p as above such that, whenever $e \in \mathcal{E}$ and $f : p(e) \to b'$ is an arrow of $\mathcal{B}$, there exists an arrow $g : e \to e'$ in $\mathcal{E}$ such that $p(g) = f$. If, in the above definitions, existentials are unique, we obtain the standard notion of *discrete (op)fibration*.

Definition 1 (approximation system) *An* approximation system *is a span of functors*

$$\begin{tikzcd}
 & \mathcal{A}px \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
\mathcal{L} & & \mathcal{M}
\end{tikzcd}$$

where p_1 is a proto-opfibration and p_2 is a protofibration.

Given an approximation system as above and $t \in \mathcal{L}$, $M \in \mathcal{M}$, we write $t \sqsubset M$ (or, sometimes, $M \sqsupset t$) if there exists an element $a \in \mathcal{A}px$ such that $p_1(a) = t$ and $p_2(a) = M$. In fact, it will be convenient to write $t \sqsubset M$ for the object a of $\mathcal{A}px$ itself, even though such an a is not necessarily unique. With this notation in place, the following is a straightforward consequence of the definition:

Theorem 2 (approximation theorem) *Let $\mathcal{L} \overset{p_1}{\leftarrow} \mathcal{A}px \overset{p_2}{\to} \mathcal{M}$ be an approximation system. Then, for any $u \in \mathcal{L}$ and $M \in \mathcal{M}$, the following are equivalent:*

1. *there exists $N \sqsupset u$ and an arrow $M \to N$ in $\mathcal{M}$;*

2. *there exists $t \sqsubset M$ and an arrow $t \to u$ in $\mathcal{L}$.*

Proof. The proof is immediate, but let us give it anyway. Fix an object u of $\mathcal{L}$ and an object M of $\mathcal{M}$.

(1)⇒(2): let $N \in \mathcal{M}$ be such that $u \sqsubset N$ and such that there exists an arrow $f : M \to N$ in $\mathcal{M}$. Since p_2 is a protofibration, we may lift f to an arrow $g : (t \sqsubset M) \to (u \sqsubset N)$ for some $t \in \mathcal{L}$, from which, by functoriality of p_1, we infer the existence of $p_1(g) : t \to u$.

(2)⇒(1): let $t \in \mathcal{L}$ be such that $t \sqsubset M$ and suppose that there is an arrow $h : t \to u$ in $\mathcal{L}$. Since p_1 is a proto-opfibration, there exists $g : (t \sqsubset M) \to (u \sqsubset N)$

for some $N \in \mathcal{M}$ and by functoriality of p_2 we get an arrow $p_2(g) : M \to N$ in $\mathcal{M}$. $\square$

2.1 Böhm Approximation Systems

Let us say that an object N of a small category $\mathcal{M}$ is *normal* if, whenever $f : N \to N'$ is an arrow of $\mathcal{M}$, we have that $N' = N$ and $f = \mathrm{id}_N$. Given an object M of $\mathcal{M}$, we write

$$\mathsf{NF}(M) := \{N \in \mathcal{M} \mid N \text{ is normal and there is an arrow } M \to N\}.$$

We extend the notation to sets of objects by setting $\mathsf{NF}(A) := \bigcup_{M \in A} \mathsf{NF}(M)$.

Given an approximation system $\mathcal{L} \leftarrow \mathcal{A} \rightarrow \mathcal{M}$ and $M \in \mathcal{M}$, we may define the following two sets:

$$\mathcal{T}(M) := \{t \in \mathcal{L} \mid t \sqsubset M\},$$
$$\mathcal{B}(M) := \{u \in \mathcal{L} \mid u \text{ is normal, there is arrow } M \to N \text{ and } u \sqsubset N\}.$$

Theorem 3 *For every approximation system $\mathcal{L} \leftarrow \mathcal{A} \rightarrow \mathcal{M}$ and $M \in \mathcal{M}$,*

$$\mathsf{NF}(\mathcal{T}(M)) = \mathcal{B}(M).$$

Proof. By definition, $u \in \mathsf{NF}(\mathcal{T}(M))$ iff there exists $t \sqsubset M$ and an arrow $t \to u$; by Theorem 2, this is equivalent to the existence of $N \sqsupset u$ and an arrow $M \to N$; this, in turn is equivalent to $u \in \mathcal{B}(M)$ by definition. $\square$

In the following, we will say that $\mathcal{M}$ is *confluent* if, whenever we have a span of arrows $N_1 \leftarrow M \rightarrow N_2$ in $\mathcal{M}$ with $N_1 \neq N_2$, there is a cospan of arrows $N_1 \rightarrow M' \leftarrow N_2$ in $\mathcal{M}$.

Definition 4 (Böhm approximation system) *A* Böhm approximation system *is an approximation system $\mathcal{L} \overset{p_1}{\leftarrow} \mathcal{A} \overset{p_2}{\rightarrow} \mathcal{M}$ in which:*

1. *$\mathcal{M}$ is confluent;*
2. *p_2 is conservative (for every morphism g of $\mathcal{A}$, $p_2(g)$ is an identity iff g is an identity);*
3. *p_2 admits oplifts of normal approximations: for every $a \in \mathcal{A}$ such that $p_1(a)$ is normal, every arrow $f : p_2(a) \to M$ admits an oplifting $g : a \to b$ such that $p_2(g) = f$.*

Proposition 5 *Let $\mathcal{L} \overset{p_1}{\leftarrow} \mathcal{A} \overset{p_2}{\rightarrow} \mathcal{M}$ be a Böhm approximation system.*

1. *Whenever there is an arrow $M \to N$ in $\mathcal{M}$, we have $\mathcal{B}(M) = \mathcal{B}(N)$.*
2. *Whenever N is normal, $\mathcal{B}(N) = \mathcal{T}(N)$.*

Proof. (1) That $\mathcal{B}(N) \subseteq \mathcal{B}(M)$ is immediate. For the converse, let $u \in \mathcal{B}(M)$. By definition, u is normal and $u \sqsubset N'$ such that there is an arrow $M \to N'$. If $N' = N$, we are done. Otherwise, by confluence, we have a cospan $N' \overset{f}{\rightarrow}$

$M' \leftarrow N$. Now, since u is normal, the arrow f may be oplifted to an arrow g of $\mathcal{A}$ such that $p_1(g) : u \to u'$ with $u' \sqsubset M$. But, again, u is normal, so by definition $p_1(g) = \mathrm{id}_u$ and $u' = u$, which implies that $u \in \mathcal{B}(N)$, as desired.

(2) Since N is normal, we immediately have $\mathcal{B}(N) \subseteq \mathcal{T}(N)$. For the converse, by definition $u \in \mathcal{T}(N)$ means $u \sqsubset N$; if we show that u is normal, we may conclude, because in that case we certainly have $u \in \mathcal{B}(N)$ by definition. Let $h : u \to u'$ be an arrow of $\mathcal{L}$. Since p_1 is a proto-opfibration, we may oplift it to an arrow g of $\mathcal{A}$, which induces an arrow $p_2(g) : N \to N'$. Now, since N is normal, by definition $p_2(g) = \mathrm{id}_N$, which by conservativity of p_2 implies that g is an identity; but g is an oplift of h, so $p_1(g) = h$, so h too is an identity and u is indeed normal. □

If we take the original multilinear approximation system based on the Taylor expansion [ER06, ER08] as a guideline, we may make the following informal identifications:

$\mathcal{T}(M) =$ "the support of the Taylor expansion of M",

$\mathcal{B}(M) =$ "the support of the Taylor expansion of the Böhm tree of M".

In §3.2 we will see how the axiomatization may be instantiated so that this is literally true, but in general this is only an intuition, because our setting is too abstract to offer a meaningful definition of Taylor expansion or Böhm tree. Nevertheless, a Böhm approximation system guarantees that these notions behave as expected: $\mathcal{B}(M)$ is an invariant of computation (Proposition 5.1), and, on normal forms, "fictitious Böhm trees" behave like real Böhm trees, in the sense that they are trivial (the Böhm tree of a normal λ-term N is N itself, which is the claim of Proposition 5.2).

Moreover, Theorem 3, which is valid even in non-Böhm approximation systems, tells us that "the support of the Taylor expansion of the Böhm tree of M" may be computed by taking the normal forms of "the support of the Taylor expansion of M". The acquainted reader will recognize here a generalized form of an important result of [ER06, ER08] (we will expand on this in §3.2).

2.2 Values

Computational systems usually come with special states / terms / programs representing results of computations. We may incorporate this into approximation systems by slightly modifying the definition:

Definition 6 (approximation system with observable values) *Let us say that a* computational context *is a small category together with a distinguished set of objects, called* observable values*.*

A (Böhm) approximation system with observable values *is a (Böhm) approximation system $\mathcal{L} \overset{p_1}{\leftarrow} \mathcal{A}px \overset{p_2}{\to} \mathcal{M}$ in which $\mathcal{L}$ and $\mathcal{M}$ are computational contexts whose observable values are in bijection (which we will abusively but without loss of generality assume to be the identity), such that, for every observable value v, we have $v \sqsubset v$ and, for any $M \in \mathcal{M}$, $v \sqsubset M$ implies $M = v$.*

The intuition behind the two additional conditions of Definition 6 with respect to Definition 1 is that observable values are "finite", in that they approximate themselves, and, moreover, they represent "complete" information to which nothing may be added: if a state/term/program contains an observable value, then it must coincide with it.

In presence of observable values, Theorem 2 may be restated in a slightly more compact form (although of course the original form still holds):

Theorem 7 (approximation theorem, with values) *Let $\mathcal{L} \overset{p_1}{\leftarrow} \mathcal{A}px \overset{p_2}{\rightarrow} \mathcal{M}$ be an approximation system with observable values. Then, for any observable value v and $M \in \mathcal{M}$, the following are equivalent:*

1. *there is an arrow $M \to v$ in $\mathcal{M}$;*

2. *there exists $t \sqsubset M$ and an arrow $t \to v$ in $\mathcal{L}$.*

So an approximation system with observable values ensures that a state / term / program has an observable value iff an approximation of it has that value.

2.3 Quantitative Information

In a sense, the above axiomatizations of approximation system are purely qualitative. However, the results that are truly of interest to us (and that will be derived in this note) are quantitative. To this end, we equip approximation systems with a little more structure.

Below, given a small category $\mathcal{C}$, we denote by $\mathcal{C}_0$ and $\mathcal{C}_1$ its set of objects and arrows, respectively.

Definition 8 (quantitative approximation system) *A* quantitative (Böhm) approximation system *is a (Böhm) approximation system with observable values $\mathcal{L} \xleftarrow{p_1} \mathcal{A} \xrightarrow{p_2} \mathcal{M}$ together with two cost functions*

$$\begin{tikzcd}
\mathcal{L}_0 \arrow[dr, "c_0"'] & & \mathcal{M}_1 \arrow[dl, "c_1"] \\
& C &
\end{tikzcd}$$

where C is an arbitrary set of costs, *such that, for every arrow $g : a \to b$ of $\mathcal{A}$ such that $p_2(b)$ is an observable value, we have $c_0(p_1(a)) = c_1(p_2(g))$.*

In the quantitative case, approximation systems ensure that a state / term / program M evaluates to an observable value v with cost c iff there exists an approximation t of M evaluating to v and, moreover, the cost c may be read directly from t. This is formalized by the following variant of Theorem 2, the proof of which is just as immediate:

Theorem 9 (quantitative approximation theorem) *Let $\mathcal{L} \longleftarrow \mathcal{A} \longrightarrow \mathcal{M}$ and $\mathcal{L}_0 \xrightarrow{c_0} C \xleftarrow{c_1} \mathcal{M}_1$ form a quantitative approximation system. Then, for every observable value v, $M \in \mathcal{M}$ and cost $c \in C$, the following are equivalent:*

1. *there is an arrow $f : M \to v$ such that $c_1(f) = c$;*

2. *there exists $t \sqsubset M$ and an arrow $h : t \to v$ such that $c_0(t) = c$.*

3 The Original Example

3.1 The Lambda-Calculus

We consider the untyped λ-calculus with a constant $*$, representing the generic result of computations:

$$M, N, P, Q ::= x \mid \lambda x.M \mid MN \mid *.$$

We write $\mathsf{fv}(M)$ for the set of free variables of M and we denote by $M\{N/x\}$ the capture-free substitution of N to every free occurrence of x in M. We write C for one-hole contexts, *i.e.*, terms containing exactly one occurrence of a special variable $\{\cdot\}$, called *hole*, and we denote by $\mathsf{C}\{M\}$ the term resulting from plugging M into the hole of C (as usual, this may capture free variables of M).

The *applicative depth* of a context C is the number of nested arguments of applications under which the hole is found. Inductively:

- the applicative depth of $\{\cdot\}$ is 0;
- if the applicative depth of C is n, then the applicative depth of any immediately larger context is still n except for the case $M\mathsf{C}$, in which it is $n+1$.

Given a context $\lambda x.\mathsf{C}$ respecting Barendregt's convention (*i.e.*, no variable appears both free and bound and every binder binds a different variable), its *index* is defined to be the positive integer $i+1$ such that i is the number of occurrences of the variable x appearing to the left of the hole in C.

Terms are evaluated by means of the standard β-rule:

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

We define *closed reduction* to be reduction restricted to β-rules as the above in which N is closed.

We denote by Λ (resp. $\Lambda_\bullet$) the free category on the graph whose nodes are (closed) λ-terms and whose edges are (closed) one-step reductions. We turn $\Lambda_\bullet$ into a computational context by letting $*$ be the only observable value.

3.2 The Taylor Expansion

The purpose of this section is to show how the original notion of (multilinear) approximation given by resource λ-terms and the Taylor expansion [ER06, ER06] fits within our axiomatization. It is intended for readers already familiar with the topic. The uninterested reader may safely skip this section and resume with §4, where the definitions are spelled out in detail.

Let $\mathcal{R}$ be the category whose objects are resource λ-terms (including $*$) and whose arrows from t to u are reductions $t \to^* u + S$ for some sum of terms S. Composition is as follows: if $\rho : t \to t'$ and $\rho' : t' \to t''$, by definition there are reductions $t \to^* t' + S'$ and $t' \to^* t'' + S''$; these may be composed into a reduction $t \to^* t'' + S' + S''$, which we take to be the witness of the arrow $\rho' \circ \rho$.

Let now $\mathcal{T}ay$ be the category whose objects are pairs $t \sqsubset M$ such that M is a λ-term and t is in the support of the Taylor expansion of M as defined in [ER08], and such that an arrow $(t \sqsubset M) \to (t' \sqsubset M')$ is a pair of reductions $\xi : t \to^* t' + S$ and $\rho : M \to^* M'$ such that ξ "follows" ρ (in the intuitive sense).

Lemma 10 *The two obvious projection functors*

$$\begin{tikzcd} & \mathcal{T}ay \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\ \mathcal{R} & & \Lambda \end{tikzcd}$$

form a Böhm approximation system.

This is arguably the "original" approximation system. In this setting, Theorem 3 takes the form of a particularly important statement, namely that the support of the Taylor expansion commutes with taking normal forms, a result first established by Ehrhard and Regnier [ER06, ER08]. In the λ-calculus, the "normal form" of a term M should be understood in the generalized sense of taking the Böhm tree of M, which we denote by $\mathsf{BT}(M)$. Below, we denote by $\mathcal{T}(\mathsf{BT}(M))$ the support of the Taylor expansion of the Böhm tree of M, as defined in [ER08].

Lemma 11 *For every λ-term M, $\mathcal{T}(\mathsf{BT}(M)) = \mathcal{B}(M)$.*

Proof. The equality is basically by definition of Taylor expansion of a Böhm tree. First, let's consider the case in which M has no head normal form, which means that $\mathsf{BT}(M) = \bot$ and $\mathcal{T}(\mathsf{BT}(M)) = \emptyset$. It is easy to see that, if P is not a head normal form and $w \sqsubset P$, then w is not normal, therefore $\mathcal{B}(M) = \emptyset$ as well. So, in what follows, we suppose that the Böhm tree of M is not $\bot$.

If $u \in \mathcal{T}(\mathsf{BT}(M))$, then $u \sqsubset N$ for some "Böhm approximation" N of M, *i.e.*, N is a finite portion of $\mathsf{BT}(M)$ where some subtrees (always arguments of applications) are replaced by $\bot$, and occurrences of $\bot$ are approximated by empty multisets in u. But this means, on the one hand, that u is normal, and, on the other hand, that $M \to^* N'$ such that N may be obtained from N' by substituting some subterms (always arguments of applications) with $\bot$, which implies that $u \sqsubset N'$, so $u \in \mathcal{B}(M)$ by definition.

For the converse, $u \in \mathcal{B}(M)$ means that u is normal and $u \sqsubset N$ such that $M \to^* N$. Since u is normal, we certainly have $u \in \mathcal{T}(\mathsf{BT}(N))$, but $\mathsf{BT}(N) = \mathsf{BT}(M)$, and we conclude. $\square$

Theorem 12 ([ER06, ER08]) *For every λ-term M, $\mathsf{NF}(\mathcal{T}(M)) = \mathcal{T}(\mathsf{BT}(M))$.*

Proof. An instance of Theorem 3, modulo Lemma 11. $\square$

To be fair, Theorem 12 is a slightly weakened form of Ehrhard and Regnier's Theorem 2 from [ER06]: indeed, Theorem 12 may be restated as "$u \in \mathcal{T}(\mathsf{BT}(M))$ iff there exists $t \in \mathcal{T}(M)$ such that $t \to^* u + S$", and Ehrhard and Regnier additionally prove that, if it exists, such a t is unique.

This uniqueness is related to the fact that the functor p_2 above is a discrete fibration, but that property alone is not enough, in our current axiomatization, to fully obtain Theorem 2 of [ER06].

More importantly, we must mention that a large part of [ER08] is in fact devoted to accounting for the coefficients appearing in the Taylor expansion, whereas here we merely account for its support. We ignore whether it is possible to define an approximation system including coefficients in $\mathcal{R}$ and $\mathcal{T}ay$ and reflecting more faithfully Ehrhard and Regnier's result. If it is, proving that it is indeed an approximation system would certainly be more delicate.

While we are at it, let us mention the connection with non-idempotent intersection types discovered by de Carvalho. Borrowing Accattoli's terminology, resource λ-terms may be typed with *multitypes*

$$A, B ::= \alpha \mid [A_1, \dots, A_n] \multimap B,$$

where α ranges over a set of base types and $[A_1, \dots, A_n]$ is the multiset containing $A_1, \dots, A_n$. The typing judgments are of the form $\Gamma \vdash t : A$ where t is a resource λ-term, A a multitype and Γ a list, permutable at will, of declaration of the form $x : \mu$ where μ is a non-empty multiset of multitypes and, as usual, Γ contains at most one declaration per variable. The typing rules are as follows:

$$\frac{}{x : [A] \vdash x : A} \qquad \frac{\Gamma, x : [A_1, \dots, A_n] \vdash t : B}{\Gamma \vdash \lambda x.t : [A_1, \dots, A_n] \multimap B} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash \lambda x.t : [\,] \multimap B}\ x \notin \Gamma$$

$$\frac{\Gamma \vdash t : [A_1, \dots, A_n] \multimap B \quad \Delta_1 \vdash u_1 : A_1 \quad \dots \quad \Delta_n \vdash u_n : A_n}{\Gamma + \sum_{i=1}^n \Delta_i \vdash t[u_1, \dots, u_n] : B}$$

where the context $\Gamma + \sum_{i=1}^n \Delta_i$ is defined as follows. First, let X be the set of all variables declared in at least one of $\Gamma, \Delta_1, \dots, \Delta_n$. We define μ_x^Γ (resp. $\mu_x^{\Delta_i}$ with $1 \leq i \leq n$) to be the multiset μ if $x : \mu$ appears in Γ (resp. Δ_i), or the empty multiset otherwise. Then, for each $x \in X$, we set $\mu_x = \mu_x^\Gamma + \sum_{i=1}^n \mu_x^{\Delta_i}$, where $+$ denotes multiset sum, and define $\Gamma + \sum_{i=1}^n \Delta_i$ to contain the declarations $x : \mu_x$ for all $x \in X$.

The above type system is Curry-style, but of course we may consider Church-style typed resource λ-terms by writing the type annotations directly inside the terms. We may then define the categories

- $\mathcal{R}_{\text{typed}}$, whose objects are Church-style typed resource λ-terms and arrows are just like in $\mathcal{R}$ (reduction preserves types);
- $\mathcal{T}ay_{\text{typed}}$, which is defined just like $\mathcal{T}ay$ but on objects $t \sqsubset M$ such that t is a Church-style typed resource λ-term.

We then have the following:

Lemma 13 *The two obvious projection functors*

$$\begin{tikzcd} & \mathcal{T}ay_{\text{typed}} \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\ \mathcal{R}_{\text{typed}} & & \Lambda \end{tikzcd}$$

form a Böhm approximation system in which, moreover, p_2 is both a proto-opfibration and a discrete fibration.

Now, a proper formulation of the link between Lemma 13 and non-idempotent intersection types requires a 2-categorical framework like the one employed in [MPV18], but let us try and give the idea: in the spirit of Melliès and Zeilberger's "functors are type systems" viewpoint [MZ15], the functor p_2 may be seen as a type system for the λ-calculus. In this system, the type derivations of a judgment $\Gamma \vdash M : A$ are objects δ of $\mathcal{T}ay_{\text{typed}}$ such that $p_2(\delta) = M$ and $p_1(\delta)$ is a Church-style resource λ-term term $\Gamma \vdash t : A$. In particular, the types of the system are just multitypes. Moreover, by definition we have $t \sqsubset M$, and the inductive definition of the approximation relation (which we did not give here but which may be found *e.g.* in [ER06, ER08]) matches the rules for assigning multitypes to resource λ-terms, so the type system results from superposing the two definitions, and is exactly de Carvalho's "system R" [dC07].

A variant of such a non-idempotent intersection type system had in fact been introduced earlier by Gardner [Gar94] (and also discussed by Kfoury [Kfo00]), the difference being that it was not based on multitypes as defined above, but on types in which, instead of multisets, one uses sequences. Gardner's system allows one to permute the elements of the sequences at will, so it is in fact equivalent to System R (which de Carvalho introduced independently). We will define this sequence-based system in detail in §4.4.

Some properties of this type system may be deduced directly from the axiomatic setting. For example, the fact that p_2 is a proto-bifibration hints to the type system enjoying subject reduction and subject expansion, albeit, strictly speaking, such properties are not provable in our present axiomatization because types do not appear explicitly (we would need a 2-categorical formulation for that). Subject reduction and subject expansion tell us that the set of types assigned to a λ-term is an invariant of β-reduction. Indeed, the type system coming from Lemma 13 is just a way of defining the so-called relational semantics of the λ-calculus, and was in fact de Carvalho's original motivation for his system R ("R" stands for "relational").

4 Approximating the Milner Abstract Machine

4.1 The Milner Abstract Machine

An *assignment* is an object of the form $[e := M]$ where x is a variable and M a λ-term not containing x free. We say that x is *declared* by the assignment.

A state of the *Milner abstract machine* (MAM) is a triple

$$M \mid S \mid E$$

where M is a λ-term, S is a stack of λ-terms and E, which is referred to as the *environment*, is a stack of assignments. We will assume that all terms appearing in a state respect Barendregt's convention, and that the state as whole respects the convention, with variables declared in environments behaving like bound variables. We may also assume the following closure constraints:

$$\begin{array}{c|c|c c c|c|c}
x & S & E'[x := M]E'' & \to_{\mathsf{v}} & M^{\alpha} & S & E'[x := M]E'' \\
MN & S & E & \to_{\mathsf{a}} & M & N \cdot S & E \\
\lambda x.M & N \cdot S & E & \to_{\mathsf{l}} & M & S & [x := N]E \\
* & \cdot & E'[x := N]E'' & \to_{\mathsf{gc}} & * & \cdot & E'E''
\end{array}$$

Figure 1: Transitions of the MAM. The notation M^{α} means an α-renaming of M using fresh variables (*i.e.*, variables not appearing in the previous state).

- the free variables of M and of all terms appearing in S are declared in E;
- if $E = E'[x := N]E''$, then the free variables of N are declared in E''.

The transitions of the MAM are given in Fig. 1. It is easy to check that the transitions preserve the above closure constraints. A state is *initial* if it is of the form $M \mid \cdot \mid \cdot$ with M closed (and respecting Barendregt's convention). A state is *reachable* if it results from a finite number of transitions from an initial state.

We define $\mathcal{MAM}$ to be the free category on the graph such that:

- its nodes are reachable MAM states;
- its edges are single transitions.

We turn $\mathcal{MAM}$ into a computational context by declaring the state $* \mid \cdot \mid \cdot$ to be the only observable value.

4.2 The Delayed Substitution Calculus

Terms of the *delayed substitution calculus* are defined as follows:

$$t, u, w ::= x \mid \lambda\langle x_1, \dots, x_n\rangle.t \mid t\langle u_1, \dots, u_n\rangle \mid t[\langle x_1, \dots, x_n\rangle := \langle u_1, \dots, u_m\rangle] \mid *,$$

where we require variables to appear linearly. The notation $t[\langle x_1, \dots, x_n\rangle := \langle u_1, \dots, u_m\rangle]$ is an explicit substitution and binds $x_1, \dots, x_n$ in t. We use bold metavariables for sequences of metavariables, *i.e.*, $\mathbf{x}$ and $\mathbf{t}$ stand for $x_1, \dots, x_n$ and $t_1, \dots, t_n$, with the number n left unspecified. We will further abbreviate $\lambda\langle\mathbf{x}\rangle.t$, $t\langle\mathbf{u}\rangle$ and $t[\langle\mathbf{x}\rangle := \langle\mathbf{t}\rangle]$ into $\lambda\mathbf{x}.t$, $t\mathbf{u}$ and $t[\mathbf{x} := \mathbf{u}]$, respectively.

The definition of reduction follows the "at a distance" approach: first, we introduce *substitution contexts*

$$[-] ::= \{\cdot\} \mid [-][\mathbf{x} := \mathbf{u}],$$

and we write $t[-]$ for $[-]\{t\}$; then, the basic reduction rules are:

$$\begin{aligned}
(\lambda\mathbf{x}.t)[-]\mathbf{u} &\to_{\mathsf{db}} t[\mathbf{x} := \mathbf{u}][-], \\
t[\langle x, \mathbf{x}\rangle := \langle u, \mathbf{u}\rangle[-]] &\to_{\mathsf{ls}} t\{u/x\}[\mathbf{x} := \mathbf{u}][-], \\
t[\langle\rangle := \langle\rangle[-]] &\to_{\mathsf{gc}} t[-].
\end{aligned}$$

$$\frac{}{y \sqsubset x \vdash y \sqsubset x} \qquad \frac{\Xi, x_1 \sqsubset x, \dots, x_n \sqsubset x \vdash t \sqsubset M}{\Xi \vdash \lambda\langle x_1, \dots, x_n\rangle.t \sqsubset \lambda x.M}$$

$$\frac{\Xi \vdash t \sqsubset M \quad \Upsilon_1 \vdash u_1 \sqsubset N \quad \dots \quad \Upsilon_n \vdash u_n \sqsubset N}{\Xi, \Upsilon_1, \dots, \Upsilon_n \vdash t\langle u_1, \dots, u_n\rangle \sqsubset MN} \qquad \frac{}{\vdash * \sqsubset *}$$

(a) Approximating terms.

$$\frac{}{\vdash \{\cdot\} \sqsubset \cdot}$$

$$\frac{u_1 \sqsubset N \quad \dots \quad u_n \sqsubset N \qquad\qquad \Xi \vdash [-] \sqsubset E}{\Xi, x_1 \sqsubset x, \dots, x_n \sqsubset x \vdash \{\cdot\}[\langle x_1, \dots, x_n\rangle := \langle u_1, \dots, u_n\rangle][-] \sqsubset [x := N]E}$$

(b) Approximating environments. The second rule uses the approximation relation introduced in Fig. 2a, without specifying the contexts, which are irrelevant (under the closure assumption, the variables declared therein will be also declared in Ξ).

$$\frac{\Xi \vdash w \sqsubset MN_1 \cdots N_n \quad \Xi \vdash [-] \sqsubset E}{w[-] \sqsubset M \mid N_1 \cdots N_n \mid E}$$

(c) Approximating states. The left premise uses the relation introduced in Fig. 2a, whereas the right premise uses the relation introduced in Fig. 2b.

Figure 2: Linear polyadic approximations for the MAM.

We also define *structural congruence* by means of the following rule:

$$t[\mathbf{x} := \mathbf{u}]\mathbf{w} \;\equiv\; t\mathbf{w}[\mathbf{x} := \mathbf{u}],$$

where we ask that none of the variables in $\mathbf{x}$ is free in $\mathbf{w}$. It is well known (see *e.g.* [ABM14]) that $\equiv$ is a bisimulation, in the sense that $t \to_{\times} u$ and $t' \equiv t$ implies $t' \to_{\times} u'$ such that $u' \equiv u$.

We let $\mathcal{D}el\mathcal{S}ub_{\bullet}$ be the free category on the graph such that:

- its nodes are closed terms of the delayed substitution calculus;
- its edges are either one-step reductions or structural equivalences.

We turn $\mathcal{D}el\mathcal{S}ub_{\bullet}$ into a computational context by letting $*$ be the only observable value.

4.3 Linear Approximations for the MAM

We define the approximation relation $\sqsubset$ between terms of the delayed substitution calculus and states of the MAM in Fig. 2. From this, we approximate transitions with reductions, as follows. The general definition will have the shape

$$\begin{tikzcd} t \arrow[r, "\xi"] \arrow[d, phantom, "\sqcap"] & u \arrow[d, phantom, "\sqcap"] \\ U \arrow[r, "\sigma"'] & V \end{tikzcd}$$

which means that we are declaring reduction ξ to approximate transition σ, as long as $t \sqsubset U$ holds:

- if σ is $U \to_{\mathsf{v}} V$, then ξ is a reduction of the form
$$x\mathbf{u}_1 \cdots \mathbf{u}_n[-][\langle x, \mathbf{y}\rangle := \langle v, \mathbf{w}\rangle][-]' \;\to\; v\mathbf{u}_1 \cdots \mathbf{u}_n[-][\mathbf{y} := \mathbf{w}][-]'.$$
- If σ is $U \to_{\mathsf{a}} V$, then ξ is the empty reduction.
- If σ is $U \to_{\mathsf{l}} V$, then ξ is a reduction of the form
$$(\lambda \mathbf{x}.t)\mathbf{u}\mathbf{u}_1 \cdots \mathbf{u}_n[-] \;\to t[\mathbf{x} := \mathbf{u}]\mathbf{u}_1 \cdots \mathbf{u}_n[-].$$
- If σ is $U \to_{\mathsf{gc}} V$, then ξ is the reduction
$$*[-][\langle\rangle := \langle\rangle][-]' \;\to\; *[-][-]'.$$

Notice that if $\xi : t \to^* u$, $\sigma : U \to_{\mathsf{x}} V$ and $\xi \sqsubset \sigma$, then $u \sqsubset V$ except when $\mathsf{x} = \mathsf{l}$, in which case we have $u \equiv\sqsubset V$. This observation allows to compositionally approximate sequences of MAM transitions with alternations of reductions and structural equivalence relations in the delayed substitution calculus: if $\rho = \sigma_1 \cdots \sigma_n$ is a sequence of MAM transitions, we write $\xi \sqsubset \rho$ just if $\xi = (\xi_1 \equiv) \cdots (\xi_n \equiv)$ such that $\xi_i \sqsubset \sigma_i$ for all $1 \leq i \leq n$. Such alternations of reductions and structural equivalence are arrows of $\mathcal{D}el\mathcal{S}ub_\bullet$, so we may define a category $\mathcal{A}px_{\mathrm{MAM}}$ as follows:

- its objects are pairs $t \sqsubset U$ where t is a closed term of the delayed substitution calculus and U a reachable state of the MAM;
- its arrows $(t \sqsubset U) \to (t' \sqsubset U')$ are pairs consisting of an arrow $\xi : t \to t'$ of $\mathcal{D}el\mathcal{S}ub_\bullet$ and a sequence of transitions $\rho : U \to^* U'$ such that $\xi \sqsubset \rho$;
- composition is concatenation.

It is now straightforward to check the following:

Lemma 14 *The span*

$$\begin{tikzcd} & \mathcal{A}px_{\mathrm{MAM}} \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\ \mathcal{D}el\mathcal{S}ub_\bullet & & \mathcal{MAM} \end{tikzcd}$$

induced by the two obvious projection functors is a Böhm approximation system with observable values (which, we recall, are $*$ for $\mathcal{D}el\mathcal{S}ub_\bullet$ and $* \mid \cdot \mid \cdot$ for $\mathcal{MAM}$).

4.4 Non-Idempotent Intersection Types

As shown in [MPV18], Fig. 2a is the basis of the standard idempotent intersection type system first introduced by Gardner [Gar94] and de Carvalho [dC07, dC18], whose derivations are given in Fig. 3.

$$\frac{}{\vdash_{*} * : \mathsf{unit}} \qquad \frac{}{x_i : A \vdash_{x_i} x : A} \qquad \frac{\Gamma, x_{i_1} : A_1, \ldots, x_{i_n} : A_n \vdash_t M : B}{\Gamma \vdash_{\lambda\langle x_{i_1},\ldots,x_{i_n}\rangle.t} \lambda x.M : A_1 \wedge \cdots \wedge A_n \to B}$$

$$\frac{\Gamma \vdash_t M : A_1 \wedge \cdots \wedge A_n \to B \quad \Delta_1 \vdash_{u_1} N : A_1 \quad \ldots \quad \Delta_n \vdash_{u_n} N : A_n}{\Gamma, \Delta_1, \ldots, \Delta_n \vdash_{t\langle u_1,\ldots,u_n\rangle} MN : B}$$

Figure 3: Non-idempotent intersection types, decorated with the underlying linear approximation. In the abstraction rule, we ask that x_i does not appear in Γ for any i.

The types of the system are given by

$$A, B ::= \mathsf{unit} \mid A_1 \wedge \cdots \wedge A_n \to B.$$

In the arrow type, the case $n = 0$ is allowed and written $\top \to B$. Type judgments are of the form $\Gamma \vdash M : A$ where Γ is a finite set (finite repetition-free list permutable at will) of declarations of the form $x_i : B$, where x is a free variable of M and $i \in \mathbb{N}$. Observe that it is *not* required that all free variables of M be declared in Γ. Indeed, in the abstraction rule we may have $n = 0$, in which case $\lambda x.M$ gets type $\top \to B$.

If we change $\wedge$ to $\otimes$ and $\to$ to $\multimap$, we see that intersection types are isomorphic to a subset of multiplicative linear logic formulas, with which delayed substitution terms may be typed in the standard way. Modulo this isomorphism, we have:

Proposition 15 ([MPV18]) *A type derivation $\Gamma \vdash M : A$ in non-idempotent intersection types is the same thing as an approximation $t \sqsubset M$ such that $\Gamma \vdash t : A$.*

The approximation corresponding to the derivation is given as the gray annotation in Fig. 3.

Notice that explicit substitutions are not used in (terms corresponding to) intersection type derivations. However, they are needed to properly approximate the MAM.

4.5 Time Complexity of the MAM

Lemma 16 *Let $\rho : U_0 \to_{\mathsf{x}_1} U_1 \to_{\mathsf{x}_2} \cdots \to_{\mathsf{x}_n} U_n$ be a sequence of MAM transitions. Then, $\xi \sqsubset \rho$ implies that $\xi = \xi_1\xi_2\cdots\xi_n$ and, for all $1 \leq i \leq n$:*

- *if $\mathsf{x}_i = \mathsf{v}$, then ξ_i is $\to_{\mathsf{ls}}$;*
- *if $\mathsf{x}_i = \mathsf{a}$, then ξ_i is $=$;*
- *if $\mathsf{x}_i = \mathsf{l}$, then ξ_i is $\to_{\mathsf{db}}\equiv$;*
- *if $\mathsf{x}_i = \mathsf{gc}$, then ξ_i is $\to_{\mathsf{gc}}$.*

Proof. Immediate from the definitions. $\square$

We define three measures on terms of the delayed substitution calculus, with $\mathsf{x} \in \{\lambda, @, \mathsf{s}\}$:

$$\begin{aligned}
\|x\|_{\mathsf{x}} &:= 0 \\
\|\lambda \mathbf{x}.t\|_{\mathsf{x}} &:= \|t\|_{\mathsf{x}} + l_{\mathsf{x}} \\
\|t\langle u_1, \ldots, u_n\rangle\|_{\mathsf{x}} &:= a_{\mathsf{x}} + \|t\|_{\mathsf{x}} + \sum_{i=1}^{n} \|u_i\|_{\mathsf{x}} \\
\|t[\mathbf{x} := \langle u_1, \ldots, u_n\rangle]\|_{\mathsf{x}} &:= a_{\mathsf{x}} + \|t\|_{\mathsf{x}} + \sum_{i=1}^{n} \|u_i\|_{\mathsf{x}}
\end{aligned}$$

where l_{x} and a_{x} are defined as follows:

- $l_\lambda := 1$ and $a_\lambda := 0$;
- $l_@ := 0$ and $a_@ := 1$;
- $l_{\mathsf{s}} := 0$ and $a_{\mathsf{s}} := n$.

In other words, $\|t\|_\lambda$, $\|t\|_@$ and $\|t\|_{\mathsf{s}}$ are the number of λ's, the number of sequences and the total length of sequences in t, respectively.

Lemma 17 (quantitative approximation system for the MAM) *Let c_0 be the function from closed terms of the delayed substitution calculus to $\mathbb{N}^3$ defined by*

$$c_0(t) := (\|t\|_\lambda, \|t\|_{\mathsf{s}}, \|t\|_@).$$

Let c_1 be the function from sequences of transitions of the MAM to $\mathbb{N}^3$ defined by

$$c_1(\rho) := (l, v, g)$$

where l, v and g are the number of transitions of ρ of type l, v *and* gc, *respectively. With these cost functions, the span of Lemma 14 is a quantitative Böhm approximation system.*

Proof. Given $\varphi : (t \sqsubset U) \to (* \sqsubset * \mid \cdot \mid \cdot)$, we need to show that $c_0(t) = c_1(p_2(\varphi))$. By definition, the latter is a triple (l, v, g) such that l, v and g are the number of l-, v- and gc-transitions in the execution $p_2(\varphi) : U \to^* * \mid \cdot \mid \cdot$ of the MAM. Since, by definition $p_1(\varphi) \sqsubset p_2(\varphi)$, by Lemma 16 we have that the arrow $p_1(\varphi) : t \to *$ is a composition of exactly l instances of $\to_{\mathsf{db}}\equiv$, v instances of $\to_{\mathsf{ls}}$ and g instances of $\to_{\mathsf{gc}}$. By inspection, we may now check that, for any delayed substitution term u such that $u \to_{\mathsf{x}} u'$, we have:

- if $\mathsf{x} = \mathsf{db}$, then $\|u\|_\lambda = \|u'\|_\lambda + 1$, $\|u\|_{\mathsf{s}} = \|u'\|_{\mathsf{s}}$ and $\|u\|_@ = \|u'\|_@$;
- if $\mathsf{x} = \mathsf{ls}$, then $\|u\|_\lambda = \|u'\|_\lambda$, $\|u\|_{\mathsf{s}} = \|u'\|_{\mathsf{s}} + 1$ and $\|u\|_@ = \|u'\|_@$;
- if $\mathsf{x} = \mathsf{db}$, then $\|u\|_\lambda = \|u'\|_\lambda$, $\|u\|_{\mathsf{s}} = \|u'\|_{\mathsf{s}}$ and $\|u\|_@ = \|u'\|_@ + 1$.

Moreover, still by inspection, we have that $u \equiv u'$ implies $\|u\|_\lambda = \|u'\|_\lambda$, $\|u\|_{\mathsf{s}} = \|u'\|_{\mathsf{s}}$ and $\|u\|_{@} = \|u'\|_{@}$. Therefore, since $c_0(*) = (0,0,0)$, we must have $c_0(t) = (l, v, g)$, as desired. $\square$

We may now prove a slightly refined version (from the quantitative viewpoint) of the results of [ER06] and [dC07, dC18]:[2]

Theorem 18 *Let M be a closed λ-term. The following are equivalent:*

1. *$M \mid \cdot \mid \cdot \to^* * \mid \cdot \mid \cdot$ on the MAM and the number of* l*-transitions,* v*-transitions and* gc*-transitions is l, v and g, respectively;*

2. *there exists $t \sqsubset M$ such that $t \to^* *$ and $\|t\|_\lambda = l$, $\|t\|_{\mathsf{s}} = v$ and $\|t\|_{@} = g$;*

3. *there is a non-idempotent intersection type derivation of $\vdash M : \mathsf{unit}$ such that the number of abstraction rules, the sum of the arities of application rules, and the number of application rules are l, v and g, respectively.*

Proof. The equivalence between (1) and (2) is an instance of Theorem 9, via Lemma 17. The equivalence between (2) and (3) is Proposition 15, modulo the observation that terms underlying intersection types derivations do not contain explicit substitutions, hence sequences only appear as arguments of applications and the measures $\|\cdot\|_{\mathsf{s}}$ and $\|\cdot\|_{@}$ therefore count the total arities of applications and the number of applications, respectively. $\square$

Transitions of type a of the MAM are not accounted for by approximations/intersection types. From the complexity viewpoint, this is not an issue, because of the following quantitative result from [AB17], refining a result in [ABM14] and stating that the number of a transitions is linearly related to the number of other transitions:

Lemma 19 *For every execution of the MAM starting from a state of the form $M \mid \varepsilon \mid \varepsilon$ with M closed and consisting of a, l and v transitions of type* a, l *and* v*, respectively, we have $a = O((|M| + 1)(l + v))$.*

5 Approximating the Interaction Abstract Machine

5.1 The Interaction Abstract Machine

An *exponential signature*, which we generically denote by σ, is a finite non-empty ordered tree (in the sense that the children of a node are linearly ordered) whose nodes are labelled by positive integers. We write $\sigma = i\langle\sigma_1, \ldots, \sigma_k\rangle$ to denote an exponential signature whose root is labelled by i and whose immediate subtrees are $\sigma_1, \ldots, \sigma_k$, in that order. We denote by Σ the set of exponential signatures.

A *multiplicative* (resp. *exponential*) *stack* is a finite sequence over $\{\mathsf{q}, *\} \cup \Sigma$ (resp. Σ). We denote by $|S|$ the length of a stack S and by $\alpha \cdot S$ the stack obtained by adjoining the symbol α to S.

[2]Those papers actually talk about the KAM (Krivine abstract machine), but it is well known that the executions of the KAM and MAM on any closed term are isomorphic [ABM14].

$$\begin{array}{r|c|c|c c c|c|c|c}
\blacktriangle\, x & \mathsf{C}\{\lambda x.\mathsf{C}'\} & S & \sigma_1 \cdots \sigma_k \cdot B & \leftrightarrow & \blacktriangledown\, \lambda x.\mathsf{C}'\{x\} & \mathsf{C} & i\langle \sigma_1, \ldots, \sigma_k\rangle \cdot S & B \\
\blacktriangle\, \lambda x.M & \mathsf{C} & \mathsf{q} \cdot S & B & \leftrightarrow & \blacktriangle\, M & \mathsf{C}\{\lambda x.\{\cdot\}\} & S & B \\
\blacktriangle\, MN & \mathsf{C} & S & B & \leftrightarrow & \blacktriangle\, M & \mathsf{C}\{\{\cdot\}N\} & \mathsf{q} \cdot S & B \\
\blacktriangledown\, N & \mathsf{C}\{M\{\cdot\}\} & S & \sigma \cdot B & \leftrightarrow & \blacktriangle\, M & \mathsf{C}\{\{\cdot\}N\} & \sigma \cdot S & B \\
\blacktriangle\, * & \mathsf{C} & \mathsf{q} \cdot S & B & \leftrightarrow & \blacktriangledown\, * & \mathsf{C} & * \cdot S & B
\end{array}$$

Figure 4: The transitions of the IAM. In the first transition, k and i are the applicative depth and the index of $\lambda x.\mathsf{C}'$, respectively (as defined in §3.1).

A state of the *interaction abstract machine* (IAM) is a quintuple

$$\blacklozenge\, M \mid \mathsf{C} \mid S \mid B$$

where $\blacklozenge \in \{\blacktriangle, \blacktriangledown\}$, M is a λ-term, C is a context such that $\mathsf{C}\{M\}$ is a closed term respecting Barendregt's convention (*i.e.*, every binder binds a distinct variable), S is a multiplicative stack and B is an exponential stack.

The transitions of the IAM are given in Fig. 4. The notation $\leftrightarrow$ means that transitions are *reversible*, in the sense that, for each such transition of the form

$$\blacklozenge_1\, M_1 \mid \mathsf{C}_1 \mid S_1 \mid B_1 \to \blacklozenge_2\, M_2 \mid \mathsf{C}_2 \mid S_2 \mid B_2,$$

the transition

$$\blacklozenge_2^*\, M_2 \mid \mathsf{C}_2 \mid S_2 \mid B_2 \to \blacklozenge_1^*\, M_1 \mid \mathsf{C}_1 \mid S_1 \mid B_1$$

is also a transition of the IAM, where $\blacktriangle^* := \blacktriangledown$ and $\blacktriangledown^* := \blacktriangle$.

Definition 20 (GoI path) *Let M be a λ-term. A* GoI path *of M is a sequence of transitions of the IAM*

$$\blacklozenge\, N \mid \mathsf{C} \mid S \mid B \;\to^*\; \blacklozenge'\, N' \mid \mathsf{C}' \mid S' \mid B'$$

such that $\mathsf{C}\{N\} = M$ (and, as a consequence, $\mathsf{C}'\{N'\} = M$ as well). Notice that, given a GoI path φ, we may unambiguously retrieve from it the term of which it is a path, which we denote by $\mathrm{tm}(\varphi)$. *A path of M is* closed *if $\blacklozenge = \blacktriangle$, $\blacklozenge' = \blacktriangledown$, $N = N'$, $\mathsf{C} = \mathsf{C}'$ and $B = B'$, and no intermediate state is of the form $\blacklozenge''\, N \mid \mathsf{C} \mid S'' \mid B''$. A closed path is* principal *if, furthermore, $N = M$, $\mathsf{C} = \{\cdot\}$ and B is empty. In that case, S and S' are called the* endpoints *of the path.*

Lemma 21 (invariance) *There is a functor*

$$\mathrm{Paths} : \Lambda_\bullet^{\mathrm{op}} \longrightarrow \mathbf{Set}$$

sending a closed λ-term M to its set of principal paths, such that, for any closed reduction $\rho : M \to^* M'$, the function $\mathrm{Paths}(\rho)$ *is an endpoint-preserving bijection.*

Proof. This is "the" GoI theorem, proved in several different contexts in the literature, starting with Girard's original paper [Gir89]. The classical argument goes through proof nets, where the result is particularly intuitive. A proof for the formulation we use is in [ADLV20]. The only thing we add here is functoriality, which is implicit in all these proofs. $\square$

We denote by $\int \mathrm{Paths}$ the category of elements of Paths. Explicitly, this is (isomorphic to) the category such that

- its objects are principal paths, varying over all closed λ-terms;
- its arrows $\varphi \to \varphi'$ are closed reductions $\rho : \mathrm{tm}(\varphi) \to^* \mathrm{tm}(\varphi')$ such that $\mathrm{Paths}(\rho)(\varphi') = \varphi$, *i.e.*, φ is the principal path corresponding to φ' via the reduction ρ;
- composition is composition of reductions.

By standard categorical lore, the functor

$$\begin{tikzcd} \int \mathrm{Paths} \arrow[d, "\mathrm{tm}"] \\ \Lambda_\bullet \end{tikzcd}$$

sending each principal path φ to its underlying term $\mathrm{tm}(\varphi)$ is a discrete fibration (in fact, Lemma 21 tells us that it is a discrete bifibration, *i.e.*, it is a discrete opfibration as well, but we will not use this).

For completeness, let us state the (restricted) soundness of the IAM, although we will not need it in these notes:

Proposition 22 (soundness of the IAM) *Let M be a closed λ-term such that $M \to^* *$. Then,*

$$\blacktriangle\, M \mid \{\cdot\} \mid \mathsf{q} \mid \cdot \quad \to^* \quad \blacktriangledown\, M \mid \{\cdot\} \mid * \mid \cdot.$$

Proof. It is not hard to prove that $M \to^* *$ by means of a closed reduction ρ. The result is then immediate from Lemma 21, by taking the image of the principal path $\blacktriangle * \mid \{\cdot\} \mid \mathsf{q} \mid \cdot \to \blacktriangledown * \mid \{\cdot\} \mid * \mid \cdot$ under $\mathrm{Paths}(\rho)$. □

5.2 The Simply-Typed Linear Polyadic Calculus

An *address* is a finite sequence of positive integers. We denote by $\mathbb{A}$ the set of addresses and use α, β to range over them. Given $\alpha, \beta \in \mathbb{A}$, we write:

- $\alpha \cdot \beta$ for the concatenation of α and β;
- $\alpha < \beta$ if α strictly precedes β in the lexicographic order;
- $\alpha \# \beta$ if α and β are not prefixes of one another.

A *pattern* is a finite, possibly empty sequence $\tau = \alpha_1 < \cdots < \alpha_n$ of addresses such that, for all $i \neq j$, $\alpha_i \# \alpha_j$. If $\tau = \beta_1 < \cdots < \beta_n$ is a pattern and α an address, we define

$$\alpha(\tau) := \alpha \cdot \beta_1 < \cdots < \alpha \cdot \beta_n,$$

which is still a pattern. If $\tau = \alpha_1 < \cdots < \alpha_n$ and, for all $1 \leq i \leq n$, τ'_i are patterns, we define

$$\tau(\tau'_1, \ldots, \tau'_n) := \alpha_1(\tau'_1) < \cdots < \alpha_n(\tau'_n),$$

$$A, B ::= \mathsf{unit} \mid \tau(A_1, \ldots, A_n) \multimap B$$

(a) Types, where $\tau = \alpha_1 < \cdots < \alpha_n$ ranges of patterns.

$$\frac{}{x : \alpha(A) \vdash x : A} \qquad \frac{\Gamma, \mathbf{x} : \tau(\mathbf{A}) \vdash t : B}{\Gamma \vdash \lambda\tau(\mathbf{x}).t : \tau(\mathbf{A}) \multimap B} \qquad \frac{}{\vdash * : \mathsf{unit}}$$

$$\frac{\Gamma \vdash t : \tau(A_1, \ldots, A_n) \multimap B \quad \Delta_1 \vdash u_1 : A_1 \quad \ldots \quad \Delta_n \vdash u_n : A_n}{\Gamma, \tau(\Delta_1, \ldots, \Delta_n) \vdash t\tau(u_1, \ldots, u_n) : B}$$

(b) Typing rules.

Figure 5: The simply-typed linear polyadic calculus.

which is still a pattern. Intuitively, a pattern is a finite tree whose leaves are its addresses, and $\tau(\tau'_1, \ldots, \tau'_n)$ is the tree obtained by "grafting" $\tau'_1, \ldots, \tau'_n$ on top of the leaves of τ.

Linear polyadic terms are defined as follows:

$$t, u, v, w ::= x \mid \lambda\tau(x_1, \ldots, x_n).t \mid t\tau(u_1, \ldots, u_n) \mid *$$

where τ ranges over patterns and each variable is required to occur at most once in a term. In $\lambda\tau(x_1, \ldots, x_n).t$ and $t\tau(u_1, \ldots, u_n)$, we require τ to be of the form $\alpha_1 < \cdots < \alpha_n$ and we say that α_i is the address of x_i and u_i, respectively. Using the abbreviation of §4.2, we may write $\lambda\tau(\mathbf{x}).t$ and $t\tau(\mathbf{u})$ for the above terms. In case τ is empty, which is possible, we simply write $\lambda().t$ and $t()$. With these notations, reduction is defined by

$$(\lambda\tau(\mathbf{x}).t)\,\tau(\mathbf{u}) \;\to\; t\{\mathbf{u}/\mathbf{x}\}.$$

Simply-typed linear polyadic terms are defined in Fig. 5. In Fig. 5a, τ is a pattern and, just like for terms, we require it to be of the form $\alpha_1 < \cdots < \alpha_n$ and say that α_i is the address of the type A_i. The empty pattern is allowed, yielding the type $() \multimap B$. Type judgments are of the form $\Gamma \vdash t : A$ where t is a term, A is a type, and Γ is a list, permutable at will, of declarations of the form $x : \alpha(B)$ where x is a variable, B a type and α an address, and in which no variable is declared twice. Whenever a list of declarations $x_1 : \alpha_1(A_1), \ldots, x_n : \alpha_n(A_n)$ is such that $\tau := \alpha_1 < \cdots < \alpha_n$ is a pattern, we write more succintly $\mathbf{x} : \tau(\mathbf{A})$, where $\mathbf{x} = x_1, \ldots, x_n$ and $\mathbf{A} = A_1, \ldots A_n$. The case $n = 0$ is possible: we may derive $\Gamma \vdash \lambda().t : () \multimap B$ from $\Gamma \vdash t : B$. If α is an address and $\Delta = x_1 : \beta_1(B_1), \ldots x_n : \beta_n(B_n)$, we define

$$\alpha(\Delta) := x_1 : \alpha \cdot \beta_1(B_1), \ldots x_n : \alpha \cdot \beta_n(B_n).$$

In the application rule, by definition, τ must be of the form $\alpha_1 < \cdots < \alpha_n$, so we set

$$\tau(\Delta_1, \ldots, \Delta_n) := \alpha_1(\Delta_1), \ldots, \alpha_n(\Delta_n).$$

▲ x_i	$\mathsf{c}\{\lambda\tau(\mathbf{x}).\mathsf{c}'\}$	s	$\leftrightarrow$	▼ $\lambda\tau(\mathbf{x}).\mathsf{c}'\{x_i\}$	c	$\alpha \cdot s$
▲ $\lambda\tau(\mathbf{x}).t$	c	$\mathsf{q} \cdot s$	$\leftrightarrow$	▲ t	$\mathsf{c}\{\lambda\tau(\mathbf{x}).\{\cdot\}\}$	s
▲ $t\tau(\mathbf{u})$	c	s	$\leftrightarrow$	▲ t	$\mathsf{c}\{\{\cdot\}\tau(\mathbf{u})\}$	$\mathsf{q} \cdot s$
▼ u_i	$\mathsf{c}\{t\tau(u_1,\ldots,\{\cdot\},\ldots,u_n)\}$	s	$\leftrightarrow$	▲ t	$\mathsf{c}\{\{\cdot\}\tau(\mathbf{u})\}$	$\alpha \cdot s$
▲ $*$	c	$\mathsf{q} \cdot s$	$\leftrightarrow$	▼ $*$	c	$* \cdot s$

Figure 6: The transitions of the pIAM. In the first transition, $\mathbf{x} = x_1, \ldots, x_n$ and α is the address of x_i. In the fourth transition, $\mathbf{u} = u_1, \ldots, u_n$ and α is the address of u_i.

We let $\mathcal{LinPoly}_\bullet$ be the free category on the graph whose nodes are closed, simply-typed linear polyadic terms and whose edges are one-step reductions. (By "simply-typed" we mean in the Church sense, *i.e.*, each object of $\mathcal{LinPoly}_\bullet$ is a term equipped with type decorations). We turn it into a computational context by taking $*$ as the only observable value.

5.3 The IAM for Polyadic Terms

Linear polyadic terms may be executed on a machine similar to the IAM, which we call pIAM ("p" is for "polyadic"), whose states are of the form

$$\blacklozenge\, t \mid \mathsf{c} \mid s$$

where $\blacklozenge \in \{▲, ▼\}$, t and c are a polyadic term and a polyadic context, respectively, such that $\mathsf{c}\{t\}$ is closed and respects Barendregt's convention, and s is a stack of elements of the set

$$\{\mathsf{q}, *\} \cup \mathbb{A},$$

where we remind that $\mathbb{A}$ is the set of addresses.

The transitions of the pIAM are given in Fig. 6. The concept of GoI path, closed path and principal path (with its endpoints) are just as in Definition 20.

In the linear case, invariance holds without any restriction on reduction:

Lemma 23 (invariance, linear case) *There is a functor*

$$\mathrm{LinPaths} : \mathcal{LinPoly}_\bullet \longrightarrow \mathbf{Set}$$

sending a linear polyadic term t to the set of its principal paths such that, for every reduction $\xi : t \to t'$, the function $\mathrm{LinPaths}(\xi)$ *is an endpoint-preserving bijection.*

Proof. Another classic of the GoI. □

We denote by $\int \mathrm{LinPaths}$ the category of elements of LinPaths. Explicitly, this is (isomorphic to) the category such that

- its objects are principal paths, varying over all closed linear polyadic terms;
- its arrows $\psi \to \psi'$ are reductions $\xi : \mathrm{tm}(t) \to^* \mathrm{tm}(t')$ such that $\psi' = \mathrm{LinPaths}(\xi)(\psi)$;

$$\frac{}{\mathsf{q} : \mathsf{unit}} \qquad\qquad \frac{}{* : \mathsf{unit}}$$

$$\frac{s : B}{\mathsf{q} \cdot s : \tau(\mathbf{A}) \multimap B} \qquad\qquad \frac{s : A_i \qquad \alpha \text{ is the address of } A_i}{\alpha \cdot s : \tau(A_1, \ldots, A_n) \multimap B}$$

Figure 7: Types of pIAM stacks.

- composition is composition of reductions.

By standard categorical lore, the functor

$$\begin{tikzcd}\int \mathrm{LinPaths} \arrow[d, "\mathrm{tm}"] \\ \mathcal{LinPoly}_\bullet\end{tikzcd}$$

sending each principal path ψ to its underlying term $\mathrm{tm}(\psi)$ is a discrete opfibration (in fact, Lemma 23 tells us that it is a discrete bifibration, *i.e.*, it is a discrete fibration as well, but we will not use this).

We will actually need finer invariants of the pIAM than Lemma 23. In order to state them, let us first define typability of stacks. We say that a pIAM stack s has type A, and we write $s : A$, if the typing may be derived from the rules of Fig. 7. Notice that A is just a linear polyadic type, and that the empty stack is not typable. We say that a state $\lozenge\, t \mid \mathsf{c} \mid s$ of the pIAM is *well-typed* if t has type A and $s : A$ and, in case $A = \mathsf{unit}$, then $s = \mathsf{q}$ if $\lozenge = \blacktriangle$, otherwise $s = *$ if $\lozenge = \blacktriangledown$.

Lemma 24 *pIAM transitions preserve well-typedness of states.*

Proof. Case inspection. □

We define the *size* of a linear polyadic type A as follows:

$$\|\mathsf{unit}\| := 1$$
$$\|\tau(A_1, \ldots, A_n) \multimap B\| := \|B\| + \sum_{i=1}^{n} \|A_i\|$$

So, the size of a type is the number of occurrences of unit appearing in it.

Lemma 25 *Let t be a closed term of type* unit *such that $\xi : t \to^* *$. Let ψ_0 be the unique principal path of $*$ and let*

$$\psi := \mathrm{LinPaths}(\xi)^{-1}(\psi_0).$$

Then, for every occurrence of subterm u of t of type A with $t = \mathsf{c}\{u\}$, ψ contains $\|A\|$ closed subpaths on u, i.e., of the form $\blacktriangle\, u \mid \mathsf{c} \mid s \to^* \blacktriangledown\, u \mid \mathsf{c} \mid s'$. Moreover, each state of ψ going through u is the extremity of one such closed path.

Proof. By induction on the length of ξ. The lemma is obviously verified for $t = *$, so the proof amounts to checking a generic reduction step $t = \mathsf{c}\{(\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q})\} \to \mathsf{c}\{p\{\mathbf{q}/\mathbf{x}\}\}$ such that $\mathsf{c}\{p\{\mathbf{q}/\mathbf{x}\}\} \to^* *$ and where we denote by ψ, ψ' the inverse image of ψ_0 in $\mathsf{c}\{(\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q})\}, \mathsf{c}\{p\{\mathbf{q}/\mathbf{x}\}\}$, respectively.

Let $\mathbf{q} = q_1, \ldots, q_n$, $p : B$ and $q_i : A_i$ for $1 \leq i \leq n$. By the induction hypothesis, we have

$$\psi' = \theta_0\chi'_1 \cdots \theta_{\|B\|-1}\chi'_{\|B\|}\theta_{\|B\|}$$

where each χ'_j is a closed path of the form

$$\blacktriangle\, p\{\mathbf{q}/\mathbf{x}\} \mid \mathsf{c} \mid s_j \to^* \blacktriangledown\, p\{\mathbf{q}/\mathbf{x}\} \mid \mathsf{c} \mid s'_j.$$

Observe that the χ'_i are disjoint, *i.e.*, they cannot be nested, because they are on the same subterm. By construction,

$$\psi = \theta_0\chi_1 \cdots \theta_{\|B\|-1}\chi_{\|B\|}\theta_{\|B\|}$$

where each χ_j is a closed path of the form

$$\blacktriangle\, (\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q}) \mid \mathsf{c} \mid s_j \to^* \blacktriangledown\, (\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q}) \mid \mathsf{c} \mid s'_j$$

arising from χ'_j. Now, if u is a subterm of t which is *not* a subterm of the redex $(\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q})$, then u appears in the θ_k's, which are (essentially) identical in ψ and ψ', so the lemma holds because it holds on ψ'. Otherwise, u appears in the χ_j's, and we will verify the lemma by inspecting such paths more closely.

First, observe that χ_j is obtained from χ'_j by "lengthening" the paths in χ'_j passing through the q_i's; for the rest, χ_j and χ'_j are (essentially) identical. That is, if χ'_j is of the form

$$\blacktriangle\, p\{\mathbf{q}/\mathbf{x}\} \mid \mathsf{c} \mid s_j \xrightarrow{1}{}^* \blacktriangle\, q_i \mid \mathsf{c}_i \mid s_{j,i} \xrightarrow{2}{}^* \blacktriangledown\, q_i \mid \mathsf{c}_i \mid s'_{j,i} \xrightarrow{3}{}^* \blacktriangledown\, p\{\mathbf{q}/\mathbf{x}\} \mid \mathsf{c} \mid s'_j$$

then χ_j is of the form

$$\begin{aligned}
\blacktriangle\, (\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q}) \mid \mathsf{c} \mid s_j \quad &\to \blacktriangle\, \lambda\tau(\mathbf{x}).p \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{q})\} \mid \mathsf{q}\cdot s_j \to \blacktriangle\, p \mid \mathsf{c}\{(\lambda\tau(\mathbf{x}).\{\cdot\})\tau(\mathbf{q})\} \mid s_j \\
&\xrightarrow{1}{}^* \blacktriangle\, x_i \mid \mathsf{c}_i \mid s_{j,i} \to \blacktriangledown\, \lambda\tau(\mathbf{x}).p \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{q})\} \mid \alpha\cdot s_{j,i} \to \blacktriangle\, q_i \mid \mathsf{c}'_i \mid s_{j,i} \\
&\xrightarrow{2}{}^* \blacktriangledown\, q_i \mid \mathsf{c}'_i \mid s'_{j,i} \to \blacktriangle\, \lambda\tau(\mathbf{x}).p \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{q})\} \mid \alpha\cdot s'_{j,i} \to \blacktriangledown\, x_i \mid \mathsf{c}_i \mid s'_{j,i} \\
&\xrightarrow{3}{}^* \blacktriangledown\, p \mid \mathsf{c}\{(\lambda\tau(\mathbf{x}).\{\cdot\})\tau(\mathbf{q})\} \mid s'_j \to \blacktriangledown\, \lambda\tau(\mathbf{x}).p \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{q})\} \mid \mathsf{q}\cdot s'_j \\
&\to \blacktriangledown\, (\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q}) \mid \mathsf{c} \mid s'_j
\end{aligned}$$

where $\mathsf{c}'_i = \mathsf{c}\{(\lambda\tau(\mathbf{x}).p)\tau(q_1, \ldots, \{\cdot\}, \ldots, q_n)\}$, α is the address of q_i and the three paths marked by 1, 2, 3 are (essentially) the same as in χ'_j. Notice that the closed path on q_i depicted in χ'_j (marked by 2) may be one of many, for several different i's, or there may even be none of such paths. The important thing is that each of them, if present, is transformed as depicted in χ_j.

So, if u is a subterm of t occurring inside a path marked by 1, 2 or 3, then it will be crossed in ψ as many times as in ψ', thus verifying the lemma. The only subterms of t that are left to verify are therefore:

$$\frac{}{\alpha(y) \sqsubset x \vdash y \sqsubset x} \qquad \frac{\Xi, \tau(\mathbf{x}) \sqsubset x \vdash t \sqsubset M}{\Xi \vdash \lambda\tau(\mathbf{x}).t \sqsubset \lambda x.M}\ x \text{ not in } \Xi \qquad \frac{}{\vdash * \sqsubset *}$$

$$\frac{\Xi \vdash t \sqsubset M \quad \Upsilon_1 \vdash u_1 \sqsubset N \quad \dots \quad \Upsilon_n \vdash u_n \sqsubset N}{\Xi, \tau(\Upsilon_1, \dots, \Upsilon_n) \vdash t\tau(u_1, \dots, u_n) \sqsubset MN}\ \tau \text{ pattern}$$

Figure 8: Linear polyadic approximations of λ-terms.

- $(\lambda\tau(\mathbf{x}).p)\tau(\mathbf{q})$, which is of type B. There are $\|B\|$ closed paths on it in ψ, namely the χ_j's.
- $\lambda\tau(\mathbf{x}).p$, which is of type $\tau(A_1, \dots, A_n) \multimap B$. Notice that each χ_j contains $1 + k_j$ closed paths on this subterm, where k_j is the number of paths marked by 2 in χ_j. These are closed paths on the q_i's, and the induction hypothesis tells us that in ψ' there are, for each $1 \leq i \leq n$, exactly $\|A_i\|$ of these, so $\sum_{j=1}^{\|B\|} k_j = \sum_{i=1}^{n} \|A_i\|$, so the total number of closed paths on $\lambda\mathbf{x}.p$ in ψ is $\sum_{j=1}^{\|B\|} 1 + k_j = \|B\| + \sum_{i=1}^{n} \|A_i\|$, which is exactly $\|\tau(A_1, \dots, A_n) \multimap B\|$.
- p, which is of type B. Each χ_j contains exactly one closed path on p, so by induction there are $\|B\|$ of them in ψ.
- the q_i's, which are of type A_i. For a given i, each χ_j contains $k_{j,i}$ closed paths on q_i, among those marked by 2. These are found also in χ'_j, and the induction hypothesis tells us that $\sum_{j=1}^{\|B\|} k_{j,i} = \|A_i\|$.

In each case, the lemma is verified, so we may conclude. □

5.4 Linear Approximations for the IAM

We start by defining polyadic approximations for λ-terms in Fig. 8, which is a generalization of Fig. 2a. Approximation judgments are still of the form $\Xi \vdash t \sqsubset M$ where t is a simply-typed linear polyadic term (of which we omit the type information) and M a λ-term, but now Ξ is a list, permutable at will, of declarations of the form $\alpha(y) \sqsubset x$ where x is a λ-calculus variable, y is a variable of the linear polyadic calculus and α an address. We succintly write $\tau(\mathbf{x}) \sqsubset x$ for a sequence of declarations $\alpha_1(x_1) \sqsubset x, \dots, \alpha_n(x_n) \sqsubset x$ such that $\tau = \alpha_1 < \dots < \alpha_n$ is a pattern, where $\mathbf{x} = x_1, \dots, x_n$. The case $n = 0$ gives us $\Xi \vdash \lambda().t \sqsubset \lambda x.M$ as soon as $\Xi \vdash t \sqsubset M$ and x does not appear in Ξ. Also, given $\Xi = \beta_1(y_1) \sqsubset x_1, \dots, \beta_n(y_n) \sqsubset x_n$ and an address α, we let

$$\alpha(\Xi) \quad := \quad \alpha \cdot \beta_1(y_1) \sqsubset x_1, \dots, \alpha \cdot \beta_n(y_n) \sqsubset x_n,$$

and, if $\tau = \alpha_1 < \dots < \alpha_n$ and $\Upsilon_1, \dots, \Upsilon_n$ are lists of declarations, we set

$$\tau(\Upsilon_1, \dots, \Upsilon_n) \quad := \quad \alpha_1(\Upsilon_1), \dots, \alpha_n(\Upsilon_n),$$

which is empty in case τ is empty.

Lemma 26 *We have that $\Xi \vdash w \sqsubset M\{N/x\}$ iff there exist terms t, $\mathbf{u} = u_1, \dots u_n$, a pattern τ and approximation contexts $\Upsilon_0, \Upsilon_1, \dots, \Upsilon_n$ such that:*

- $w = t\{\mathbf{u}/\mathbf{x}\}$ *and* $\Xi = \Upsilon_0, \tau(\Upsilon_1, \dots, \Upsilon_n)$,
- $\Upsilon_0, \tau(\mathbf{x}) \sqsubset x \vdash t \sqsubset M$
- *and* $\Upsilon_i \vdash u_i \sqsubset N$ *for all* $1 \leq i \leq n$.

Proof. By induction on M. □

In what follows, if $\varphi : M \to^* M'$ is a reduction, we write $\mathsf{C}\{\varphi\}$ for the reduction $\mathsf{C}\{M\} \to^* \mathsf{C}\{M'\}$ performing φ under the context C. We use the same notation for reductions in the polyadic calculus. In fact, we extend it as follows: if $\xi_1, \dots, \xi_n$ are reductions such that $\xi_i : u_i \to^* u'_i$ for all $1 \leq i \leq n$, and if t and $\tau(x_1, \dots, x_n)$ are a term and a pattern, then we have one reduction of the form

$$t\tau(u_1, \dots, u_n) \to^* t\tau(u'_1, \dots, u'_n)$$

for each possible way of interleaving the steps of $\xi_1, \dots, \xi_n$. We abusively denote by $t\tau(\xi_1, \dots, \xi_n)$ *any* such reduction.

Let $\sigma_0 : (\lambda x.M)N \to M\{N/x\}$ be a β-rule, let $\sigma := \mathsf{C}\{\sigma_0\}$ be a reduction step and let ξ be a reduction in the linear polyadic calculus. We define the relation $\Xi \vdash \xi \sqsubset \sigma$ by induction on C:

- $\Xi \vdash \xi \sqsubset \sigma_0$ holds just if ξ is a reduction step such that

  $$\begin{tikzcd} (\lambda\tau(\mathbf{x}).t)\tau(\mathbf{u}) \arrow[r, "\xi"] \arrow[d, phantom, "\sqcap"] & t\{\mathbf{u}/\mathbf{x}\} \arrow[d, phantom, "\sqcap"] \\ (\lambda x.M)N \arrow[r, "\sigma_0"'] & M\{N/x\} \end{tikzcd}$$

  where the approximation relations between terms hold under Ξ.

- $\Xi \vdash \xi \sqsubset \lambda x.\mathsf{C}'\{\sigma_0\}$ holds whenever $\xi = \lambda\tau.\xi_0$ such that $\Xi, \tau \sqsubset x \vdash \xi_0 \sqsubset \mathsf{C}'\{\sigma_0\}$ holds.
- $\Xi \vdash \xi \sqsubset \mathsf{C}'\{\sigma_0\}Q$ holds whenever $\xi = \xi_0\tau(\mathbf{q})$ such that $\Xi_0 \vdash \xi_0 \sqsubset \mathsf{C}'\{\sigma_0\}$, $\Upsilon_i \vdash q_i \sqsubset Q$ for all $1 \leq i \leq n$ and $\Xi = \langle \Xi_0, \tau(\Upsilon_1, \dots, \Upsilon_n)\rangle$.
- $\Xi \vdash \xi \sqsubset P\mathsf{C}'\{\sigma_0\}$ whenever ξ is of the form $p\tau(\xi_1, \dots, \xi_n)$ such that $\Xi_0 \vdash p \sqsubset P$, $\Upsilon_i \vdash \xi_i \sqsubset \mathsf{C}'\{\sigma_0\}$ for all $1 \leq i \leq n$ and $\Xi = \langle \Xi_0, \tau(\Upsilon_1, \dots, \Upsilon_n)\rangle$

Now, if $\varphi = \sigma_1 \cdots \sigma_n$ is an arbitrary reduction in the λ-calculus composed of β-steps σ_i, and if ξ is a reduction in the polyadic calculus, then we define $\Xi \vdash \xi \sqsubset \varphi$ just if $\xi = \xi_1 \cdots \xi_n$ such that $\Xi \vdash \xi_i \sqsubset \sigma_i$ for each $1 \leq i \leq n$. In particular, the identity reduction on M is approximated under Ξ by the identity reduction on any t such that $\Xi \vdash t \sqsubset M$.

We define an approximation relation $\Xi \vdash \mathsf{c} \sqsubset \mathsf{C}$ between contexts of the linear polyadic calculus and contexts of the λ-calculus adapting Fig. 8. We only give the two non-trivial cases, the others are immediate:

$$\frac{}{\vdash \{\cdot\} \sqsubset \{\cdot\}} \qquad \frac{\Xi \vdash t \sqsubset M \quad \Upsilon \vdash \mathsf{c} \sqsubset \mathsf{C}}{\Xi, \Upsilon \vdash t\tau(u_1, \dots, \mathsf{c}, \dots, u_n) \sqsubset M\mathsf{C}}$$

$$\frac{}{\mathsf{q} \sqsubset \mathsf{q}} \qquad \frac{}{* \sqsubset *} \qquad \frac{s \sqsubset S}{\mathsf{q} \cdot s \sqsubset \mathsf{q} \cdot S} \qquad \frac{s \sqsubset S \quad \alpha \text{ linearization of } \sigma}{\alpha \cdot s \sqsubset \sigma \cdot S}$$

Figure 9: Approximating multiplicative stacks.

where τ is an arbitrary pattern and $u_1, \ldots, u_n$ arbitrary terms. We write $\mathsf{c} \sqsubset \mathsf{C}$ when c and C are closed.

The *linearization* of an exponential signature $i\langle\sigma_1, \ldots, \sigma_k\rangle$ is the address defined inductively as

$$i \cdot \alpha_1 \cdots \alpha_k$$

where α_i is the linearization of σ_i, for all $1 \leq i \leq k$. Let s and S be a stack of the pIAM and a multiplicative stack of the IAM, respectively. We define the approximation relation $s \sqsubset S$ in Fig. 9.

We define an approximation relation between states of the pIAM and states of the IAM as follows. We write

$$\Diamond\, t \mid \mathsf{c} \mid s \;\sqsubset\; \blacklozenge\, M \mid \mathsf{C} \mid S \mid B$$

just if:

1. $\Diamond = \blacklozenge$;
2. $\mathsf{c} \sqsubset \mathsf{C}$ and $\mathsf{c}\{t\} \sqsubset \mathsf{C}\{M\}$;
3. $s \sqsubset S$.

One may check that (2) implies $\Xi \vdash t \sqsubset M$ for some Ξ.

Finally, we define an approximation relation $\psi \sqsubset \varphi$ between GoI paths of the pIAM and GoI paths of the IAM. The definition is by induction on the length of φ, so it boils down to going through each transition of Fig. 4 and defining the approximating transition of the pIAM:

$$\begin{tikzcd}
\blacktriangle\, x_i \mid \mathsf{c}\{\lambda\mathbf{x}.\mathsf{c}'\} \mid s \arrow[rr, leftrightarrow] & & \blacktriangledown\, \lambda\mathbf{x}.\mathsf{c}'\{x_i\} \mid \mathsf{c} \mid \alpha \cdot s \\
\sqcap & \sqcap & \sqcap \\
\blacktriangle\, x \mid \mathsf{C}\{\lambda x.\mathsf{C}'\} \mid S \mid \sigma_1 \cdots \sigma_k \cdot B \arrow[rr, leftrightarrow] & & \blacktriangledown\, \lambda x.\mathsf{C}'\{x\} \mid \mathsf{C} \mid j\langle\sigma_1, \ldots, \sigma_k\rangle \cdot S \mid B
\end{tikzcd}$$

$$\begin{tikzcd}
\blacktriangle\, \lambda\tau(\mathbf{x}).t \mid \mathsf{c} \mid \mathsf{q} \cdot s \arrow[rr, leftrightarrow] & & \blacktriangle\, t \mid \mathsf{c}\{\lambda\tau(\mathbf{x}).\{\cdot\}\} \mid s \\
\sqcap & \sqcap & \sqcap \\
\blacktriangle\, \lambda x.M \mid \mathsf{C} \mid \mathsf{q} \cdot S \mid B \arrow[rr, leftrightarrow] & & \blacktriangle\, M \mid \mathsf{C}\{\lambda x.\{\cdot\}\} \mid S \mid B
\end{tikzcd}$$

$$\begin{tikzcd}
\blacktriangle\, t\tau(\mathbf{u}) \mid \mathsf{c} \mid s \arrow[rr, leftrightarrow] & & \blacktriangle\, t \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{u})\} \mid \mathsf{q} \cdot s \\
\sqcap & \sqcap & \sqcap \\
\blacktriangle\, MN \mid \mathsf{C} \mid S \mid B \arrow[rr, leftrightarrow] & & \blacktriangle\, M \mid \mathsf{C}\{\{\cdot\}N\} \mid \mathsf{q} \cdot S \mid B
\end{tikzcd}$$

$$\begin{tikzcd}
\blacktriangledown\, u_i \mid \mathsf{c}\{t\tau(u_1,\ldots,\{\cdot\},\ldots,u_n)\} \mid s \arrow[rr, leftrightarrow] & & \blacktriangle\, t \mid \mathsf{c}\{\{\cdot\}\tau(\mathbf{u})\} \mid \alpha\cdot s \\
\sqcap & \sqcap & \sqcap \\
\blacktriangledown\, N \mid \mathsf{C}\{M\{\cdot\}\} \mid S \mid \sigma\cdot B \arrow[rr, leftrightarrow] & & \blacktriangle\, M \mid \mathsf{C}\{\{\cdot\}N\} \mid \sigma\cdot S \mid B
\end{tikzcd}$$

$$\begin{tikzcd}
\blacktriangle\, * \mid \mathsf{c} \mid \mathsf{q}\cdot s \arrow[rr, leftrightarrow] & & \blacktriangledown\, * \mid \mathsf{c} \mid *\cdot s \\
\sqcap & \sqcap & \sqcap \\
\blacktriangle\, * \mid \mathsf{C} \mid \mathsf{q}\cdot S \mid B \arrow[rr, leftrightarrow] & & \blacktriangledown\, * \mid \mathsf{C} \mid *\cdot S \mid B
\end{tikzcd}$$

Notice that $\psi \sqsubset \varphi$ implies $\mathsf{tm}(\psi) \sqsubset \mathsf{tm}(\varphi)$.

We may now define a category $\mathcal{A}px_{\text{IAM}}$ as follows:

- its objects are pairs $\psi \sqsubset \varphi$ as defined above;
- its arrows $(\psi \sqsubset \varphi) \to (\psi' \sqsubset \varphi')$ are pairs (ξ, ρ) s.t. $\xi : \mathsf{tm}(\psi) \to^* \mathsf{tm}(\psi')$ is a reduction and $\rho : \mathsf{tm}(\varphi) \to^* \mathsf{tm}(\varphi')$ is a closed reduction such that $\xi \sqsubset \rho$ and ψ, ψ' correspond to each other via ξ and φ, φ' correspond to each other via ρ;
- composition is concatenation of reductions.

The following is now a tedious but unsurprising verification:

Lemma 27 *The span*

$$\begin{tikzcd}
& \mathcal{A}px_{\text{IAM}} \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
\int \mathrm{LinPaths} & & \int \mathrm{Paths}
\end{tikzcd}$$

composed of the obvious projection functors is an approximation system.

Recall the observations made in §5.1 and §5.2 about the functors $\mathrm{tm} : \mathrm{LinPaths} \to \mathcal{L}in\mathcal{P}oly_\bullet$ and $\mathrm{tm} : \mathrm{Paths} \to \Lambda_\bullet$ being a discrete opfibration and a discrete fibration, respectively. This means, in particular, that these are a proto-opfibration and a protofibration. Since proto(op)fibrations compose, by Lemma 27 we obtain

Lemma 28 *The span*

$$\begin{tikzcd}
& \mathcal{A}px_{\text{IAM}} \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
\int \mathrm{LinPaths} \arrow[d, "\mathrm{tm}"'] & & \int \mathrm{Paths} \arrow[d, "\mathrm{tm}"] \\
\mathcal{L}in\mathcal{P}oly_\bullet & & \Lambda_\bullet
\end{tikzcd}$$

is a Böhm approximation system with observable values (which, we recall, is $*$ both in $\mathcal{L}in\mathcal{P}oly_\bullet$ and $\Lambda_\bullet$).

$$\frac{}{x : \alpha(A) \vdash x : A} \qquad \frac{\Gamma, x : \tau(\mathbf{A}) \vdash M : B}{\Gamma \vdash \lambda x.M : \tau(\mathbf{A}) \multimap B}\ x \text{ not in } \Gamma \qquad \frac{}{\vdash * : \mathsf{unit}}$$

$$\frac{\Gamma \vdash M : \tau(A_1, \dots, A_n) \multimap B \quad \Delta_1 \vdash N : A_1 \quad \dots \quad \Delta_n \vdash N : A_n}{\Gamma, \tau(\Delta_1, \dots, \Delta_n) \vdash MN : B}$$

Figure 10: Polyadic types for the λ-calculus.

5.5 Polyadic Types for the λ-Calculus

Following [MPV18], we know that as soon as we have a well-behaved notion of simply-typed approximation for λ-terms, we have an intersection type system corresponding to it. In particular, simply-typed linear polyadic terms yield the type derivations whose rules are defined in Fig. 10. These are a non-associative variant of intersection types, which we will simply call *polyadic types*.

Typing judgments are of the form $\Gamma \vdash M : A$ where M is a λ-term, A a polyadic type (Fig. 5a) and Γ is a list, permutable at will, of typing declarations of the form $x : \alpha(B)$, where x is a λ-calculus variable, B a polyadic type and α an address. Contrarily to Fig. 5, a variable may be declared more than once, even with the same type (polyadic types are "non-idempotent"), but never with the same address. We write $x : \tau(\mathbf{A})$ for the typing context $x : \alpha_1(A_1), \dots, x : \alpha_n(A_n)$ in which $\tau = \alpha_1 < \cdots < \alpha_n$ is a pattern, and $\mathbf{A} = A_1, \dots, A_n$. This notation is used in the abstraction rule. The case $n = 0$ gives us $\Gamma \vdash \lambda x.M : () \multimap B$ from $\Gamma \vdash M : B$, as long as x does not appear in Γ. In the application rule, the context $\Gamma, \tau(\Delta_1, \dots \Delta_n)$ is defined just as in the corresponding rule of Fig. 5b.

Notice how, as expected, the type system of Fig. 10 results from superposing Fig. 5b and Fig. 8, just as non-idempotent intersection types (Fig. 3) result from superposing the standard system of simple types for delayed substitution terms (ignoring explicit substitutions) and Fig. 2a. Therefore, a similar result as Proposition 15, which is about non-idempotent intersection types, holds for polyadic types:

Proposition 29 *A type derivation $\Gamma \vdash M : A$ in polyadic types (Fig. 10) is the same thing as a linear polyadic approximation $t \sqsubset M$ (Fig. 8) such that $\Gamma \vdash t : A$ (Fig. 5b).*

5.6 Complexity of the IAM

We define the *size* of exponential signatures of the IAM inductively: if the sizes of $\sigma_1, \dots, \sigma_k$ are $n_1, \dots, n_k$, respectively, then the size of $i\langle\sigma_1, \dots, \sigma_k\rangle$ is $1 + \sum_{i=1}^{k} n_i$.

Let φ be a GoI path of the IAM. We write:

- $\mathrm{len}(\varphi)$ for its length (*i.e.*, the number of transitions of the IAM);

- $\mathrm{h}(\varphi)$ for the maximum length of the multiplicative stacks appearing in φ;

- $\mathrm{e}(\varphi)$ for the maximum size of exponential signatures appearing in φ.

The first two quantities may be defined identically for a GoI path ψ of the pIAM. The third quantity is replaced by the length of the longest address appearing in ψ, which is denoted by $\mathrm{a}(\psi)$.

Let M be a closed λ-term such that $M \to^* *$. Obviously, $*$ has a unique principal path, so by Lemma 21 M too has a unique principal path, which we denote by M_*. The same may be said of any closed simply-typed linear polyadic term $t : \mathsf{unit}$, because typability implies $t \to^* *$, and we use the same notation t_*. Notice that M_* (resp. t_*) is just the execution of the IAM (resp. pIAM) from the initial state $\blacktriangle M \mid \{\cdot\} \mid \mathsf{q} \mid \cdot$ (resp. $\blacktriangle t \mid \{\cdot\} \mid \mathsf{q}$).

Lemma 30 *Let M be a closed λ-term such that $M \to^* *$ and let ψ be a GoI path of the pIAM such that $\psi \sqsubset M_*$. Then:*

1. $\mathrm{len}(M_*) = \mathrm{len}(\psi)$;

2. $\mathrm{h}(M_*) = \mathrm{h}(\psi)$;

3. $\mathrm{e}(M_*) = \mathrm{a}(\psi)$.

Proof. Point (1) is immediate from the definition of approximation relation. Also by definition, we have that if $\blacklozenge N \mid \mathsf{C} \mid S \mid B$ and $\lozenge u \mid \mathsf{c} \mid s$ are two corresponding states of M_* and ψ (*i.e.*, the n-th state of M_* and the n-th state of ψ for some $0 \le n \le \mathrm{len}(M_*)$), then $\lozenge u \mid \mathsf{c} \mid s \sqsubset \blacklozenge N \mid \mathsf{C} \mid S \mid B$, which implies $s \sqsubset S$, which in turn implies, again by definition, that S and s have equal length, yielding point (2); moreover, if the i-th element of S is an exponential signature σ, then the i-th element of s is an address α which is the linearization of σ, so the length of α is exactly the size of σ, proving point (3). $\square$

Let A be a polyadic type. Recall that in §5.3 we defined $\|A\|$ to be the number of occurrences of unit in A. We now define the *height* of A as follows:

$$\mathrm{hgt}(\mathsf{unit}) := 1$$
$$\mathrm{hgt}(\tau(A_1,\dots,A_n) \multimap B) := \max\left(\max_{1\le i\le n} \mathrm{hgt}(A_i), \mathrm{hgt}(B)\right) + 1$$

We also define the *pattern height* of a type A as follows:

$$\mathrm{ph}(\mathsf{unit}) := 0$$
$$\mathrm{ph}(\tau(A_1,\dots,A_n) \multimap B) := \max\left(\max_{1\le i\le n} \mathrm{ph}(A_i), \mathrm{ph}(B), \mathrm{hgt}(\tau)\right)$$

Let t be a closed simply-typed linear polyadic term. We write:

- $\|t\|$ for the sum of all $\|A\|$ for each occurrence of subterm of t of type A;

- $\mathrm{hgt}(t)$ and $\mathrm{ph}(t)$ for the maximum of $\mathrm{hgt}(A)$ and $\mathrm{ph}(A)$, respectively, with A ranging over the types of subterms of t.

Lemma 31 *For every closed linear polyadic term t : unit, we have:*

1. $\mathrm{len}(t_*) = 2\|t\| - 1$;
2. $\mathrm{h}(t_*) = \mathrm{hgt}(t)$;
3. $\mathrm{a}(t_*) = \mathrm{ph}(t)$.

Proof. For point 1, Lemma 25 tells us that t_* goes through each occurrence of subterm of t of type A exactly $2\|A\|$ times, once "upwards" and once "downwards". So t_* goes through exactly $2\|t\|$ states, hence the result.

Points 2 and 3 are immediate consequences of Lemma 24. □

Lemma 32 (quantitative approximation system for the IAM) *Let us define c_0 to be the function from closed linear polyadic terms to $\mathbb{N}^3$ such that*

$$c_0(t) := (2\|t\| - 1, \mathrm{hgt}(t), \mathrm{ph}(t)).$$

Let c_1 be the function from closed reductions over closed λ-terms $\rho : M \to^* N$ to $\mathbb{N}^3$ such that

$$c_1(\rho) := \begin{cases} c & \text{if } N \neq * \\ (\mathrm{len}(M_*), \mathrm{h}(M_*), \mathrm{e}(M_*)) & \text{if } N = * \end{cases}$$

where $c \in \mathbb{N}^3$ is some fixed triple whose value is irrelevant. With these cost functions, the span of Lemma 28 is a quantitative Böhm approximation system.

Proof. We need to prove that, given an arrow $\chi : (\psi \sqsubset M_*) \to (* \sqsubset *)$ of $\mathcal{A}px_{\mathrm{IAM}}$ (where we abusively denoted by $*$ the unique principal path of the linear polyadic term / λ-term $*$), we have $c_0(\mathrm{tm}(\psi)) = c_1(\mathrm{tm}(p_2(\chi)))$. Notice that, since $\mathrm{tm}(\psi)$ is typable by definition and $\mathrm{tm}(\psi) \to^* *$, we have $\mathrm{tm}(\psi)$: unit. Therefore, we may write

$$\begin{aligned} c_0(\mathrm{tm}(\psi)) &= (2\|\mathrm{tm}(\psi)\| - 1, \mathrm{hgt}(\mathrm{tm}(\psi)), \mathrm{ph}(\mathrm{tm}(\psi))) && \text{by definition,} \\ &= (\mathrm{len}(\psi), \mathrm{h}(\psi), \mathrm{a}(\psi)) && \text{by Lemma 31,} \\ &= (\mathrm{len}(M_*), \mathrm{h}(M_*), \mathrm{e}(M_*)) && \text{by Lemma 30,} \\ &= c_1(\mathrm{tm}(p_2(\chi))) && \text{by definition,} \end{aligned}$$

which proves the claim. □

In the following, we apply the notations $\|\cdot\|$, $\mathrm{hgt}(\cdot)$ and $\mathrm{ph}(\cdot)$ to polyadic type derivations (Fig. 10), knowing that these, by definition, are isomorphic to simply-typed linear polyadic terms.

Theorem 33 *Let M be a closed λ-term. The following are equivalent:*

1. *$M \to^* *$ and $\mathrm{len}(M_*) = l$, $\mathrm{h}(M_*) = h$ and $\mathrm{e}(M_*) = e$;*
2. *there exists $t \sqsubset M$ such that t : unit and $l = 2\|t\| - 1$, $\mathrm{hgt}(t) = h$ and $\mathrm{ph}(t) = e$;*
3. *there is a polyadic type derivation δ of $\vdash M$: unit such that $2\|\delta\| - 1 = l$, $\mathrm{hgt}(\delta) = h$ and $\mathrm{ph}(\delta) = e$.*

Proof. The equivalence between (1) and (2) is an instance of Theorem 9, via Lemma 32. The equivalence between (2) and (3) is Proposition 29. $\square$

Corollary 34 *Let M be a closed λ-term of size n and applicative depth d such that $M \to^* *$. Then, the following are equivalent:*

1. *there exist $l, h, e \in \mathbb{N}$ such that the execution of the IAM on M may be simulated by a deterministic Turing machine in time $\mathsf{poly}(nl)$ and space $1 + (1 + (h + d)e) \log n$;*
2. *there exists $t \sqsubset M$ such that $t : \mathsf{unit}$ and $\|t\| = l$, $\mathrm{hgt}(t) = h$ and $\mathrm{ph}(t) = e$.*

Proof. For the time bound, a step of the IAM may be simulated with a polynomial slowdown including at least a linear slowdown in the size of the term being executed, so we conclude by Theorem 33.

For the space bound, simply observe that a state of the IAM executing M has size bounded by

$$1 + \log n + \mathrm{h}(M_*)\,\mathrm{e}(M_*) \log n + d\,\mathrm{e}(M_*) \log n :$$

the 1 is the direction bit; the $\log n$ is for the pointer to the current subterm of M; $\mathrm{e}(M_*) \log n$ is for representing exponential signatures, which are trees with at most $\mathrm{e}(M_*)$ nodes decorated by integers smaller than n; therefore, the second to last and last term in the sum are for representing the multiplicative stack and the exponential stack, respectively (the length of the latter being bounded by d, a standard invariant of the IAM). We then conclude by Theorem 33. $\square$

6 Approximating Turing Machines

6.1 Turing Machines

Let Σ be a finite alphabet including a blank symbol $\square$, Q a finite set of states and $[d] = \{1, \dots, d\}$ with $d > 0$. Let $H := \Sigma + (\Sigma \times Q)$. A one-tape Turing machine M on the alphabet Σ, with set of states Q and degree of non-determinism d (*i.e.*, the maximum number of choices in its transition table) may be presented by a function

$$\delta_M : H^3 \times [d] \longrightarrow H$$

such that, for all $s_{-1}, s_0, s_1 \in \Sigma$ and $1 \leq j \leq d$:

- $\delta_M(s_{-1}, s_0, s_1, j) = s_0$;
- for all $q \in Q$, exactly one of $\delta_M(s_{-1}, s_0, (s_1, q))$, $\delta_M(s_{-1}, (s_0, q), s_1)$ and $\delta_M((s_{-1}, q), s_0, s_1)$ belongs to $\Sigma \times Q$ (the other being all in Σ).

A Turing machine M may be thought of as acting on infinite strings $\phi : \mathbb{Z} \to H$ verifying that

- $\phi(i) = \square$ except for finitely many $i \in \mathbb{Z}$;

- there exists a unique $i \in \mathbb{Z}$ such that $\phi(i) \in \Sigma \times Q$. Such an i is the current position of the head on the tape, and $\pi_2\phi(i)$ is the current state.

We call such infinite strings *infinitary configurations*. Given $\phi, \phi' : \mathbb{Z} \to H$ and $1 \leq j \leq d$, we write $\phi \xrightarrow{j} \phi'$ just if, for all $i \in \mathbb{Z}$,

$$\phi'(i) := \delta_M(\phi(i-1), \phi(i), \phi(i+1), j).$$

The constraints on δ_M guarantee that ϕ' is an infinitary configuration whenever ϕ is.

What is usually called a *configuration* of M is just a finite string $\gamma \in H^*$ containing exactly one symbol of type $\Sigma \times Q$. We say that M has a *transition* from γ to γ', and we write $\gamma \xrightarrow{j} \gamma'$, just if:

- γ and γ' have equal length m;
- there exist two infinitary configurations ϕ, ϕ' such that $\phi \xrightarrow{j} \phi'$ and $k \in \mathbb{Z}$ such that, for all $1 \leq i \leq m$, $\gamma_i = \phi(k+i)$ and $\gamma'_i = \phi'(k+i)$, where γ_i, γ'_i is the i-th symbol of γ, γ', respectively, whereas, for all $j \in \mathbb{Z}$, $j < k+1$ or $j > k+m$ implies $\phi(j) = \phi'(j) = \square$.

We denote by $\mathcal{NT}ur$ the free category on the graph:

- whose nodes are either the Boolean constants 0 and 1 or pairs (M, γ) where M is a Turing machine and γ a configuration of M;
- such that there is an edge $(M, \gamma) \to (M', \gamma')$ just if $M' = M$ and $\gamma \to \gamma'$ is a transition of M, and there is an edge $(M, \gamma) \to b$ with $b = 1$ (resp. $b = 0$) just if $\gamma = (s, q)\gamma'$ and q is an accepting (resp. rejecting) state of M (*i.e.*, γ is a halting configuration and the head is placed on the first symbol).

We turn this into computational contexts by letting the observable values be the Boolean constants 0 and 1.

6.2 Turing Circuits

Given a Turing machine M with alphabet Σ, states Q and degree of nondeterminism d, *M-circuits* are defined as follows:

$$C, C' ::= x \mid C[x := C'] \mid \blacktriangledown(C) \mid \triangledown(C_1, C_2, C_3; C_4) \mid h \mid j \mid b$$

where x ranges over a countable set of variables, whereas h, j and b range over $H := \Sigma + (\Sigma \times Q)$, $[d]$ and $\{0, 1\}$ respectively. These are called the *ground constants*. In $C[x := C']$, the variable x is bound in C. This represents sharing. A circuit is *closed* if it has no free variables.

M-circuits may be typed according to the rules of Fig. 11. There are three types: symb, choice and bool, standing form the sets $H = \Sigma + (\Sigma \times Q)$, $[d]$ and $\{0, 1\}$, respectively. Notice that there are no variables of type bool and that the only terms of type choice are constants and variables. In the sharing rule,

$$\frac{A \in \{\mathsf{symb}, \mathsf{choice}\}}{x : A \vdash x : A} \qquad \frac{\Gamma \vdash C' : A \quad \Delta, x_1 : A, \ldots, x_n : A \vdash C : B}{\Gamma, \Delta \vdash C\{x/x_1, \ldots, x_n\}[x := C'] : B}$$

$$\frac{\Gamma \vdash C : \mathsf{symb}}{\Gamma \vdash \blacktriangledown(C) : \mathsf{bool}} \qquad \frac{\Gamma_i \vdash C_i : \mathsf{symb} \quad \Delta \vdash C' : \mathsf{choice}}{\Gamma_1, \Gamma_2, \Gamma_3, \Delta \vdash \triangledown(C_1, C_2, C_3; C') : \mathsf{symb}}$$

$$\frac{\Gamma_i \vdash C_i : \mathsf{symb} \quad \Delta \vdash C' : \mathsf{choice}}{\Gamma_1, \Gamma_2, \Delta \vdash \triangleright(C_1, C_2; C') : \mathsf{symb}} \qquad \frac{\Gamma_i \vdash C_i : \mathsf{symb} \quad \Delta \vdash C' : \mathsf{choice}}{\Gamma_1, \Gamma_2, \Delta \vdash \triangleleft(C_1, C_2; C') : \mathsf{symb}}$$

$$\frac{}{\vdash h : \mathsf{symb}} \qquad \frac{}{\vdash j : \mathsf{choice}} \qquad \frac{b \in \{0, 1\}}{\vdash b : \mathsf{bool}}$$

Figure 11: Typing rules for Turing circuits.

$n = 0$ is allowed, which means that sharing also encapsulates discarding. We write $C[- := C']$ to mean $C[x := C']$ where x does not appear free in C.

The evaluation rules are as follows:

$$\begin{array}{rcll} \blacktriangledown((s, q)) & \to & \begin{cases} 1 & \text{if } q \text{ is an accepting state of } M \\ 0 & \text{if } q \text{ is a rejecting state of } M \end{cases} & \\ \triangledown(h_{-1}, h_0, h_1; j) & \to & \delta_M(h_{-1}, h_0, h_1, j) & \\ C[x := h] & \to & C\{h/x\} & h \text{ is a ground constant} \end{array}$$

where $C\{h/x\}$ denotes, as usual, the substitution of h to all free occurrences of x in C.

We let $\mathcal{B}ool\mathcal{C}irc_\bullet$ be the posetal category whose

- objects are either the Boolean constants $0, 1$ or pairs (M, C) such that M is a Turing machine and C a closed M-circuit other than a Boolean ground constant;
- there is an arrow $(M, C) \to (M', C')$ just if $M = M'$ and C evaluates to C', or $(M, C) \to b$ with $b \in \{0, 1\}$ just if C evaluates to b.

6.3 Approximating Turing Machines

Given $p \in \mathbb{N}$ and $m > 0$, we are now going to define the *canonical M-circuit of depth p and width m*. Such a circuit, denoted by $C_p^m(M)$, will have exactly p free variables $y_1, \ldots, y_p$ of type choice and m free variables $x_1, \ldots, x_m$ of type symb. We write $C_p^m(M)(C_1', \ldots, C_m')$ for $C_p^m(M)\{C_1'/x_1\} \cdots \{C_1'/x_1\}$. The definition is by induction on p, as follows:

$$\begin{aligned} C_0^m(M) &:= \blacktriangledown(x_1)[- := x_2] \cdots [- := x_m] \\ C_{p+1}^m(M) &:= C_p^m(M)(\triangledown(\Box, z_1, z_2; u), \triangledown(z_1, z_2, z_3; u), \ldots, \\ &\qquad \triangledown(z_{m-2}, z_{m-1}, z_m; u), \triangledown(z_{m-1}, z_m, \Box; u))[\mathbf{z} := \mathbf{x}][u := y_{p+1}] \end{aligned}$$

where we used the abbreviation $[\mathbf{z} := \mathbf{x}]$ for $[z_1 := x_1] \cdots [z_m := x_m]$.

A *closed canonical M-circuit* is either a Boolean ground constant or of the form

$$C_m^p(M)(\gamma; \mathbf{j}) := C_m^p(M)\{\gamma_1/x_1\} \cdots \{\gamma_m/x_m\}\{j_1/y_1\} \cdots \{j_p/y_p\}$$

where γ is a configuration of M of length m and γ_i is the ground constant corresponding to the i-th symbol of γ, and $\mathbf{j} \in [d]^p$ with j_i being the ground constant corresponding to i-th element of $\mathbf{j}$.

We denote by $\mathcal{CanCirc}_\bullet$ the subposet of $\mathcal{BoolCirc}_\bullet$ of closed canonical circuits and Boolean constants. We turn this into a computational context by letting the Boolean constants be the only observable values.

Let now M be a Turing machine, with degree of non-determinism equal to d, let γ be a configuration of M of length m, and let C be an M-circuit. We write $(M, C) \sqsubset (M, \gamma)$ just if $C = C_m^p(M)(\gamma; \mathbf{j})$ for some $p \in \mathbb{N}$ and $\mathbf{j} \in [d]^p$. From this, we define a category $\mathcal{A}px_{\mathrm{NTur}}$ whose:

- objects are either pairs $b \sqsubset b$ with $b \in \{0, 1\}$ or $(M, C) \sqsubset (M, \gamma)$ as above;
- an arrow $((M, C) \sqsubset (M, \gamma)) \to ((M, C') \sqsubset (M, \gamma'))$ is a sequence of transitions of M
$$\gamma \xrightarrow{j_1} \gamma_1 \xrightarrow{j_2} \cdots \xrightarrow{j_{n-1}} \gamma_{n-1} \xrightarrow{j_n} \gamma'$$
where the length of the configurations is m and there exist $p \in \mathbb{N}$, $\mathbf{j}' \in [d]^p$ such that, if we set $\mathbf{j} = j_n \cdots j_1$, we have $C = C_m^{p+n}(M)(\gamma; \mathbf{j}'\mathbf{j})$ and $C' = C_m^p(M)(\gamma'; \mathbf{j}')$. An arrow $((M, C) \sqsubset (M, \gamma)) \to (b \sqsubset b)$ with $b \in \{0, 1\}$ is a sequence of transitions of M
$$\gamma \xrightarrow{j_1} \gamma_1 \xrightarrow{j_2} \cdots \xrightarrow{j_{n-1}} \gamma_{n-1} \xrightarrow{j_n} \gamma'$$
where the length of the configurations is m and, if we set $\mathbf{j} = j_n \cdots j_1$, we have $C = C_m^n(M)(\gamma; \mathbf{j}) \to^* b$.
- Composition is concatenation (in both cases above, $n = 0$ is allowed).

Lemma 35 *Equipped with the obvious projection functors, the span*

$$\begin{tikzcd} & \mathcal{A}px_{\mathrm{NTur}} \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\ \mathcal{CanCirc}_\bullet & & \mathcal{NTur} \end{tikzcd}$$

is an approximation system with observable values (which, we recall, are the Boolean constants).

6.4 Running Time of Turing Machines via Circuits

Lemma 36 (quantitative approximation system for Turing machines) *Let c_0 be the function assigning to a closed canonical M-circuit C the pair $(p, m) \in \mathbb{N}^2$*

where p and m are the height and width of the canonical M-circuit underlying C, respectively.

Let c_1 be the function assigning to an arrow f of $\mathcal{NT}ur$ the pair $(l, m) \in \mathbb{N}^2$ where l and m are the number of transitions and the length of the configurations appearing in f, respectively.

With these cost functions, the span of Lemma 28 is a quantitative approximation system.

Proof. Immediate from the definitions. $\square$

In the following, given $w \in (\Sigma \setminus \{\square\})^*$ of length n and $m \geq n$, we define the configuration $\mathsf{init}_k(w) := (w_1, q_0)w_2 \cdots w_n\square^{m-n}$, where w_i is the i-th letter of w and q_0 the initial state of M. A configuration of length $m > 0$ is accepting if it is of the form $(s, q)w$ for some $s \in \Sigma$, $w \in \Sigma^{m-1}$. We say that M accepts w if there is m and a finite number of transitions from $\mathsf{init}_m(w)$ to an accepting state.

Theorem 37 *Let M be a Turing machine with alphabet Σ and degree of non determinism d. Then, the following are equivalent:*

- *M accepts $w \in (\Sigma \setminus \{\square\})^*$ in l steps and using at most m cells of its tape;*
- *there exists $\mathbf{j} \in [d]^l$ such that $C^l_m(M)(\mathsf{init}_m(w); \mathbf{j}) \to^* 1$.*

The above result is an instance of Theorem 9, via Lemma 36, although of course it may be proved directly and immediately from the definitions. The goal here is not to do more work than necessary to prove an essentially trivial result, but to show that it also fits in our framework of approximations. This is remarkable because Theorem 37 has a number of important consequences.

We start by observing that, if we make the innocuous assumption that, for any halting state q of any Turing machine M, we have, for all $s, s_{-1}, s_0, s_1 \in \Sigma$ and $j \in [d]$, $\delta_M((s, q), s_0, s_1, j) = s_0$, $\delta_M(s_{-1}, (s_q), s_1, j) = (s, q)$ and $\delta_M(s_{-1}, s_0, (s, q), j) = s_0$ (*i.e.*, M acting on halting configurations does nothing), we obtain the following:

Lemma 38 *If $C^p_m(\gamma; \mathbf{j}) \to^* b$ with $b \in \{0, 1\}$, then for any $k \in \mathbb{N}$ and any $\mathbf{j}' \in [d]^k$, $C^{p+k}_m(\gamma; \mathbf{j}'\mathbf{j}) \to^* b$.*

With the help of the above result, the first application of Theorem 37 is the famous Cook-Levin theorem:

Theorem 39 (Cook-Levin) SAT *is* NP*-complete with respect to logspace reductions.*

Proof. Membership is obvious, the non-trivial part is showing hardness. There is a well-known logspace reduction from CIRCUIT SAT to SAT (see *e.g.* [Pap94]), so it is enough to show that CIRCUIT SAT is NP-hard with respect to logspace reductions. We remind that CIRCUIT SAT is the following problem: given a Boolean circuit, is there a way to set its inputs so that it evaluates to 1?

So let $L \in \mathsf{NP}$. We need to exhibit a logspace-computable function r such that, for every $w \in \{0, 1\}^*$, $w \in L$ iff $r(w) \in$ CIRCUIT SAT. Since $L \in \mathsf{NP}$, there is a non-deterministic Turing machine M deciding L in polynomial time. Let Σ, Q, d and p be the alphabet, set of states, degree of non-determinism and polynomial bounding the running time of M, respectively.

Now, the key observation is that, since δ_M is a finite function, modulo an (arbitrary) encoding of the elements of $H := \Sigma + (\Sigma \times Q)$ as binary strings, δ_M may be computed by a finite family $(B_i(M))_{1 \leq i \leq k}$ of Boolean circuits with $3k + \log d$ inputs (supposing that integers are represented in binary), where k is an integer such that $2^k \geq |H|$, depending on the encoding. Given the binary encodings of $h_{-1}, h_0, h_1 \in H$ and $j \in [d]$, each $B_i(M)$ computes the i-th bit of the binary encoding of $\delta_M(h_{-1}, h_0, h_1, j)$. The same holds for the function $H \to \{0,1\}$ mapping $(s,q) \in \Sigma \times Q$ to 1 whenever q is accepting, and mapping anything else to 0: since it is a finite function, it may be implemented by a Boolean circuit with k inputs and one output.

If $\mathcal{T}ur\mathcal{C}irc$ and $\mathcal{B}ool\mathcal{C}irc$ are the categories of (not necessarily closed) Turing circuits and Boolean circuits, respectively, with evaluations as arrows, the above defines a functor

$$\mathrm{enc} : \mathcal{T}ur\mathcal{C}irc \longrightarrow \mathcal{B}ool\mathcal{C}irc$$

mapping every closed canonical circuit to its implementation. Notice that every free variable of type choice of C will induce $\log d$ inputs in $\mathrm{enc}(C)$.

Given $w \in (\Sigma \setminus \square)^*$ of length n, we know that M terminates in at most $p(n)$ steps, during which it cannot use more that $p(n)$ cells of the tape. So let

$$r(w) := \mathrm{enc}(C^{p(n)}_{p(n)}(M)(\mathsf{init}_{p(n)}(w); -)),$$

where the dash indicates that we leave unset the $p(n) \log d$ inputs corresponding to the $p(n)$ free variables of type choice of $C^{p(n)}_{p(n)}(M)$. Now:

$$\begin{aligned} w \in L \text{ iff } & M \text{ accepts } w \text{ in } p(n) \text{ steps using } p(n) \text{ cells} \\ \text{iff } & \text{there exists } \mathbf{j} \in [d]^{p(n)} \text{ such that } r(w)(\mathbf{j}) \to^* 1 \\ \text{iff } & r(w) \in \textsc{circuit sat} \end{aligned}$$

where by $r(w)(\mathbf{j})$ we denote the closed Boolean circuit obtained by setting the inputs of $r(w)$ to the encodings of the various elements of $\mathbf{j}$. The first double implication is given by definition and Lemma 38 (even if, in general, M might take strictly less than $p(n)$ steps, that lemma tells us that we may round up to $p(n)$ by adding some "do nothing" transitions), the second double implication is Theorem 37 and the third is again by definition.

To conclude the proof, we need to check that r is logspace computable. This is clear because $|r(w)| = O(p(n)^2)$, as $C^{p(n)}_{p(n)}(M)(\mathsf{init}_{p(n)}(w); -)$ contains $p(n)^2$ occurrences of $\triangledown$, one occurrence of $\blacktriangledown$ and $O(p(n))$ ground constants, and these are all mapped to circuits of constant size. □

In the special case of deterministic Turing machines, the above proof gives another famous result:

Theorem 40 (Ladner) CIRCUIT VALUE *is* P*-complete with respect to logspace reductions.*

Proof. Recall that circuit value is the following problem: given a closed Boolean circuit (*i.e.*, with no unfixed input), does it evaluate to 1? As above, membership in P is obvious, so we need to prove P-hardness.

Now, by inspecting the proof of Theorem 39, we see that the Boolean circuit $r(w)$ has $p(n) \log d$ unset inputs. Hence, if the polytime machine M is deterministic, $d = 1$ and $r(w)$ is a closed Boolean circuit (indeed, the canonical M-circuits $C_m^p(M)$ do not depend on their free variable of type choice). So, if L happens to be decided by a polytime deterministic Turing machine, r reduces L to circuit value. But the class of problems decided by deterministic polytime Turing machines is exactly P. □

Finally, we have yet another avatar of the Cook-Levin theorem:

Theorem 41 P *is the class of languages decided by* DLOGTIME*-uniform families of Boolean circuits of polynomial size.*

Proof. That a DLOGTIME-uniform family of Boolean circuits decides a language in P is immediate. For the converse, let $L \in \mathsf{P}$ be decided by a machine M with a polynomial bound p and let, for $n \in \mathbb{N}$, C_n be the Boolean circuit $\mathrm{enc}(C_{p(n)}^{p(n)}(M))((x_1, q_0)x_2 \cdots x_n \square^{p(n)-n})$, where by this notation we mean that we fixed the inputs so that the state is the initial one and the tape is filled with blanks, except for the first n symbols, the corresponding inputs of which are left unset. By Theorem 37, $(C_n)_{n \in \mathbb{N}}$ is a family of Boolean circuits deciding L. It is of polynomial size: as already observed in the proof of Theorem 39, the size of C_n is $O(p(n)^2)$. The fact that it is DLOGTIME-uniform is a standard result that we do not wish to prove here, as it would require too many additional technical definitions. □

References

[AB17] Beniamino Accattoli and Bruno Barras. Environments and the complexity of abstract machines. In *Proceedings of PPDP*, pages 4–16, 2017.

[ABM14] Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In *Proceedings of ICFP*, pages 363–376, 2014.

[ADLV20] Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The machinery of interaction. In *Proceedings of PPDP*, pages 4:1–4:15, 2020.

[ADLV21] Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The (in)efficiency of interaction. *Proceedings of the ACM on Programming Languages*, 5(POPL), 2021.

[CT20] Jules Chouquet and Christine Tasson. Taylor expansion for call-by-push-value. In *Proceedings of CSL*, pages 16:1–16:16, 2020.

[dC07] Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Ph.d. thesis, Université de la Méditerranée – Aix-Marseille 2, 2007.

[dC18] Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Math. Struct. Comput. Sci.*, 28(7):1169–1203, 2018.

[ER06] Thomas Ehrhard and Laurent Regnier. Böhm trees, krivine's machine and the taylor expansion of lambda-terms. In *Proceedings of CiE*, pages 186–197, 2006.

[ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3):347–372, 2008.

[Gar94] Philippa Gardner. Discovering needed reductions using type theory. In *Proceedings of TACS*, pages 555–574, 1994.

[Gir89] Jean-Yves Girard. Geometry of interaction I: Interpretation of system F. In *Proceedings of Logic Colloquium 1988*, pages 221–260, 1989.

[Kfo00] A. J. Kfoury. A linearization of the lambda-calculus and consequences. *J. Log. Comput.*, 10(3):411–436, 2000.

[Maz17] Damiano Mazza. *Polyadic Approximations in Logic and Computation*. *Habilitation* thesis, Université Paris 13, 2017.

[MPV18] Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. *Proceedings of the ACM on Programming Languages*, 2(POPL:6), 2018.

[MZ15] Paul-André Melliès and Noam Zeilberger. Functors are type refinement systems. In *Proceedings POPL*, pages 3–16, 2015.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.