Krivine Machine and Taylor Expansion in a Non-uniform Setting

Antoine Allioux
Institut de Recherche en Informatique Fondamentale
Paris, France
antoine.allioux@gmail.com

The Krivine machine is an abstract machine implementing the linear head reduction of λ-calculus. Ehrhard and Regnier gave a resource sensitive version returning the annotated form of a λ-term accounting for the resources used by the linear head reduction. These annotations take the form of terms in the resource λ-calculus.

We generalize this resource-driven Krivine machine to the case of the algebraic λ-calculus. The latter is an extension of the pure λ-calculus allowing for the linear combination of λ-terms with coefficients taken from a semiring. Our machine associates a λ-term M and a resource annotation t with a scalar α in the semiring describing some quantitative properties of the linear head reduction of M.

In the particular case of non-negative real numbers and of algebraic terms M representing probability distributions, the coefficient α gives the probability that the linear head reduction actually uses exactly the resources annotated by t. In the general case, we prove that the coefficient α can be recovered from the coefficient of t in the Taylor expansion of M and from the normal form of t.

1 Introduction

The Krivine machine is an abstract machine implementing the linear head reduction [1] on the pure λ-calculus. Ehrhard and Regnier gave a resource sensitive version [3] returning the annotated form of a λ-term accounting for the resources used by the linear head reduction. These annotations take the form of terms in the resource λ-calculus. As an example, the ordinary term $((\lambda x.(x)x)\lambda x.x)c_0$ which reduces to the constant c_0 is annotated by the following resource term $\langle\langle\lambda x.\langle x\rangle x^1\rangle(\lambda x.x)^2\rangle c_0^1$. This resource term informs us that $\lambda x.x$ is used twice during the reduction and x and c_0 are used once.

We generalize this resource-driven Krivine machine to the case of the algebraic λ-calculus[1]. The latter is an extension of the pure λ-calculus allowing for the linear combination of λ-terms with coefficients taken from a semiring. Some properties enjoyed by the ordinary λ-calculus do not hold anymore in the case of the algebraic λ-calculus and some results become nontrivial. Our machine associates a λ-term M and a resource annotation t with a scalar α in the semiring describing some quantitative properties of the linear head reduction of M. We will only consider terms reducing to a multiple of a constant for the sake of convenience.

In the particular case of non-negative real numbers and of terms M representing probability distributions, the coefficient α gives the probability that the linear head reduction actually uses exactly the resources annotated by t. In the general case, we prove that the coefficient α can be recovered from the coefficient of t in the Taylor expansion of M and from the normal form of t. A more detailed report concerning this work can be found at `http://allioux.iiens.net/taylor/report.pdf`.

[1] This machine has been implemented and is available online at `http://allioux.iiens.net/taylor/`.

I. Cervesato and M. Fernández (Eds): Fourth International Workshop on Linearity 2016 (LINEARITY'16)
EPTCS 238, 2017, pp. 24–32, doi:10.4204/EPTCS.238.3

Algebraic equalities of the $\mathbb{S}$-module		
$M+\mathbf{0} \equiv_{\text{alg}} M$	$(M+N)+P \equiv_{\text{alg}} M+(N+P)$	$M+N \equiv_{\text{alg}} N+M$
$\alpha(M+N) \equiv_{\text{alg}} \alpha M+\alpha N$	$\alpha M+\beta M \equiv_{\text{alg}} (\alpha+\beta)M$	$\alpha(\beta M) \equiv_{\text{alg}} (\alpha\beta)M$
$1M \equiv_{\text{alg}} M$	$\mathbf{0}M \equiv_{\text{alg}} \mathbf{0}$	$\alpha\mathbf{0} \equiv_{\text{alg}} \mathbf{0}$
Linear properties		
$\lambda x.(M+N) \equiv_{\text{alg}} \lambda x.M+\lambda x.N$	$\lambda x.(\alpha M) \equiv_{\text{alg}} \alpha(\lambda x.M)$	$\lambda x.\mathbf{0} \equiv_{\text{alg}} \mathbf{0}$
$(\mathbf{0})M \equiv_{\text{alg}} \mathbf{0}$	$(\alpha M)N \equiv_{\text{alg}} \alpha(M)N$	$(M+N)P \equiv_{\text{alg}} (M)P+(N)P$

Table 1: Algebraic equalities of the algebraic λ-calculus

2 Algebraic lambda calculus

The algebraic λ-calculus is an extension of the pure λ-calculus allowing for the linear combination of λ-terms. More precisely, we endow it with a structure of left $\mathbb{S}$-module where $\mathbb{S}$ is a semiring. We shall follow the presentation of the algebraic λ-calculus given in [5].

2.1 Grammar

Let x be a variable in $\mathscr{V}$, the set of variables, and let α be a scalar in $\mathbb{S}$. The grammar of the algebraic λ-calculus is the following:

$$\Lambda_{\mathbb{S}} : M,N ::= x \mid \lambda x.M \mid (M)N \mid \alpha M \mid M+N \mid \mathbf{0} \tag{1}$$

We denote $\equiv_{\text{alg}}$ the equivalence relation described in Table 1 making $\Lambda_{\mathbb{S}}$ into a left $\mathbb{S}$-module and providing linear properties to terms. We consider the terms of the quotient set $\Lambda_{\mathbb{S}}/\equiv_{\text{alg}}$ up to α-conversion and we call them *algebraic terms*. We define free variables and α-conversion as in [5].

2.2 Algebraic states

The behaviour of the Krivine machine is defined on some structures we call *states* with which we can associate a unique algebraic term rather than on algebraic terms directly. A state is a snapshot of the abstract machine at a given time and represents the dissection of a *unique* λ-term.

Algebraic environment An algebraic environment is a finite partial function E mapping variables to closures. We introduce the notation $E_{x\mapsto\Gamma}$ to refer to the environment which behaves like E for variables other than x and which maps x to Γ.

Algebraic closure An algebraic closure Γ is a pair (M,E) composed of an algebraic term $M \in \Lambda_{\mathbb{S}}$ and of an environment E such that $\text{FV}(M) \subseteq \text{Dom}(E)$ where $\text{FV}(M)$ denotes the free variables of M and $\text{Dom}(E)$ denotes the domain of E.

Algebraic state An algebraic state is a nonempty stack of closures. We choose to denote states as triples (M,E,Π) where (M,E) is the first closure of the stack and Π is the stack of the remaining closures. Indeed, our Krivine machine implementing the linear head reduction, we reduce according to the structure of the first closure and we give it a special status. We refer to the set of the algebraic states by $\mathscr{S}(\Lambda_{\mathbb{S}})$.

The intuition behind algebraic states can be made explicit by defining the function $\mathrm{T} : \mathscr{S}(\Lambda_{\mathbb{S}}) \to \Lambda_{\mathbb{S}}$ which given an algebraic state returns its unique associated algebraic term.

Given any algebraic closure (M,E) and any stack of algebraic closures $\Gamma_1,\dots,\Gamma_n$ with $n \geq 0$, we first define T on closures and then extend it to states as follows:

$$\mathrm{T}(M,E) = M[\mathrm{T}(E(x))/x]_{x\in \mathrm{Dom}(E)}$$
$$\mathrm{T}(M,E,(\Gamma_1,\dots,\Gamma_n)) = (\dots(\mathrm{T}(M,E))\mathrm{T}(\Gamma_1)\dots)\mathrm{T}(\Gamma_n)$$

2.3 Krivine machine

In this particular section the semiring $\mathbb{S}$ is complete — its sum is infinitary.

We give a description of the Krivine machine as the limit K of the sequence $(K_n)_{n\in\mathbb{N}}$ defined by induction on (n,M) lexicographically ordered where n is a non-negative integer and M is an algebraic term. The induction on n turns the reduction of M into a finite process even for non-normalizing terms. We also enrich the grammar of the algebraic λ-calculus with the constant c_0 as we restrict our study to closed terms reducing to this constant.

- $K_0(M,E,\Pi) = \mathbf{0}$,
- $K_{n+1}(c_0,E,\emptyset) = c_0$,
- $K_{n+1}(x,E,\Pi) = K_n(E(x),\Pi)$ if $x \in \mathrm{Dom}(E)$,
- $K_{n+1}(\lambda x.M,E,\Gamma :: \Pi) = K_n(M,E_{x\mapsto\Gamma},\Pi)$ assuming $x \notin \mathrm{Dom}(E)$,
- $K_{n+1}((M)N,E,\Pi) = K_n(M,E,(N,E) :: \Pi)$.

These rules, excluding the first two ones, are the ones of the original Krivine machine. As the algebraic λ-calculus is just an extension of the ordinary λ-calculus, it suffices to add the two following rules to the description of the Krivine machine to handle it:

- $K_{n+1}(\alpha M,E,\Pi) = \alpha K_{n+1}(M,E,\Pi)$,
- $K_{n+1}(M+N,E,\Pi) = K_{n+1}(M,E,\Pi) + K_{n+1}(N,E,\Pi)$.

Finally we set $K = \lim_{n\to\infty} K_n$.

3 Resource lambda calculus

We recall the syntax and the reduction of the resource λ-calculus which has been defined in [4]. Indeed, we will define the Taylor expansion of an algebraic term in terms of a sum of resource terms.

3.1 Grammar

The resource λ-calculus shares its syntax with the ordinary λ-calculus with the exception that the application takes multisets of terms as argument. We use the multiplicative notation to denote multisets so the multiplicative unit 1 is the empty multiset. For example s^2t is the multiset formed of two occurrences of s and one occurrence of t. Multisets are commutative. The multiset union of S and T is denoted ST. The multiplicity of an element t in a multiset T is given by $T(t)$. The support of T, denoted $\mathrm{supp}(T)$, is the set of elements of T whose multiplicity is nonzero.

The grammar of *simple terms* in the resource λ-calculus is:

$$\Delta : s,t,u ::= x \mid \lambda x.t \mid \langle t\rangle S \tag{2}$$

where $x,y,\dots \in \mathscr{V}$, the set of variables and where S is a finite multiset of simple terms. We denote the set of simple terms Δ and we refer to its elements using lower case letters $s,t,\dots$. The set of finite multisets of simple terms is denoted $\Delta^!$. We call its elements simple *poly*-terms and we refer to them using upper case letters $S,T,\dots$. When a term t can either be a simple term or a simple poly-term we say it is in $\Delta^{(!)} = \Delta \cup \Delta^!$.

When denoting an application, we use the Krivine notation which we recall: for any simple term t and any simple poly-terms $S_1,\dots,S_n$, we denote the application $\langle \dots \langle t\rangle S_1 \dots \rangle S_n$ by the simplified form $\langle t\rangle S_1 \dots S_n$.

The module $\mathbb{S}\langle\Delta^{(!)}\rangle$ is the set of linear combinations of simple (poly-)terms with coefficients in $\mathbb{S}$. We call its elements (poly-)terms in opposition to *simple* (poly-)terms which are not part of a linear combination. These combinations can not be expressed in the syntax of the resource λ-calculus contrarily to the algebraic λ-calculus. We refer to (poly-)terms using the letters $\mathscr{S},\mathscr{T},\dots$. We denote $\mathscr{S}_s$ the coefficient of the (poly-)term s in $\mathscr{S}$. Finally we extend the grammar of the resource λ-calculus to all (poly-)terms by multilinearity so that: $\lambda x.(t+u) = \lambda x.t + \lambda x.u$, $\langle s+t\rangle T = \langle s\rangle T + \langle t\rangle T$ and $(s+t)T = sT + tT$.

3.2 Linear substitution and reduction

The reduction of the resource λ-calculus is based on a particular notion of linear substitution. What distinguishes linear substitutions from classical substitutions is that, in the former case, substituted terms have to be used once and only once whereas this restriction does not apply in the latter case.

For a variable x and a term s, we define $\deg_x(s)$ to be the number of occurrences of x in s and we call it the degree in x of s.

Let $\hat{s}$ be the resource term obtained from a resource term s by renaming its different occurrences of x to $x_1,\dots,x_n$ with $n = \deg_x(s)$. $\hat{s}$ is such that for all $i \in [\![1,n]\!]$, $\deg_{x_i}(\hat{s}) = 1$ and $s = \hat{s}[x/x_1,\dots,x_n]$.

Let s be a simple term and let $t_1 \dots t_n$ be any poly-term with n being a non-negative integer, the linear substitution of s by $t_1 \dots t_n$ is defined as follows:

$$\partial_x(s,t_1 \dots t_n) = \begin{cases} \displaystyle\sum_{f\in\mathfrak{S}^n} \hat{s}[t_{f(1)}/x_1,\dots,t_{f(n)}/x_n] & \text{if } \deg_x(s) = n \\ \mathbf{0} \in \mathbb{S}\langle\Delta\rangle & \text{if } \deg_x(s) \neq n \end{cases} \tag{3}$$

with $\mathfrak{S}^n$ being the group of permutations on the set $\{1,\dots,n\}$. This construction can be extended to simple (poly-)terms.

We extend this notation to the linear substitution of several variables. For all poly-terms $T_1,\dots,T_n$ with n being a non-negative integer,

$$\partial_{x_1,\dots,x_n}(s,T_1,\dots,T_n) = \partial_{x_n}(\dots\partial_{x_1}(s,T_1),\dots,T_n) \tag{4}$$

This substitution does not depend on the order of the iterated substitutions as the variables $x_1,\dots,x_2$ are pairwise distincts.

We derive the β-reduction relation for the resource λ-calculus from this linear substitution. A redex in the resource λ-calculus is of the form $\langle\lambda x.s\rangle T$ and reduces as follows: $\langle\lambda x.s\rangle T \to_\beta \partial_x(s,T)$.

We extend this relation to $\mathbb{S}\langle\Delta^{(!)}\rangle \times \mathbb{S}\langle\Delta^{(!)}\rangle$ by defining it as being the least relation closed under the following rules, assuming $s \to_\beta \mathscr{S}$ with $s \in \Delta$ and $\mathscr{S} \in \mathbb{S}\langle\Delta\rangle$:

$\langle s\rangle T \to_\beta \langle\mathscr{S}\rangle T \qquad \langle u\rangle sT \to_\beta \langle u\rangle \mathscr{S} T \qquad \lambda x.s \to_\beta \lambda x.\mathscr{S} \qquad s+u \to_\beta \mathscr{S}+u$

This relation is confluent and strongly normalizing for $\mathbb{S} = \mathbb{N}$ as proved in [2] and we derive NF, the unique normalization map $\mathbb{N}\langle\Delta^{(!)}\rangle \to \mathbb{N}\langle\Delta_0^{(!)}\rangle$, where Δ_0 stands for the set of normal simple terms.

3.3 Resource states

Similarly to the case of the algebraic λ-calculus, we define resource closures, resource environments and resource states in a mutually recursive fashion.

Resource environment A resource environment is a total function from the set of variables $\mathscr{V}$ to resource closures. e_0 is the empty environment mapping any variable in $\mathscr{V}$ to the empty closure 1. We use the notation $[x \mapsto c]$ to refer to the environment which maps the variable x to the closure c and all the other variables to the closure 1. Given two environments e' and e'', we define their pointwise concatenation $e'e''$ such that for all variables x, $e'e''(x) = e'(x)e''(x)$.

Resource closure A resource closure is defined as a pair $c = (T, e)$ where T is a simple poly-term and e is a resource environment. A resource closure is said to be elementary when its multiset T is a singleton. The empty closure is $1 = (1, e_0)$. We use letters $c, c_1, \ldots$ for general resource closures and $\gamma, \gamma_1, \ldots$ for elementary resource closures.

Resource state A resource state is a triple (t, e, π) where (t, e) is an elementary resource closure and where π is a stack of resource closures. We denote the set of resource states $\mathscr{S}(\Delta)$.

4 Quantitative Krivine machine and Taylor expansion

4.1 Quantitative Krivine machine

In turn we define our quantitative Krivine machine (**qKAM**) K[2] which draws its inspiration from the one described in [3]. This definition is the main contribution of this paper. It is important to note that for the sake of convenience we will only consider closed algebraic terms which reduce to the constant c_0. From now on, we therefore enrich the syntax of the resource λ-calculus with this same constant c_0.

The following machine computes a coefficient associated with an algebraic state and a resource state. We remind that an algebraic state corresponds to a unique λ-term and a resource state corresponds to a unique sum of resource terms. Therefore, in the case of algebraic terms whose sums correspond to probability distributions, this coefficient will be the sum of the probabilities that each resource term in the sum describe a resource usage of the reduction of the algebraic term to c_0.

Definition 4.1. (Quantitative Krivine machine) The quantitative Krivine machine is defined as a matrix $K \in \mathbb{S}^{\mathscr{S}(\Lambda_\mathbb{S}) \times \mathscr{S}(\Delta)}$. It is defined by induction on the pair $(\text{size}(t,e,\pi), \text{size}(M,E,\Pi))$[3] lexicographically ordered. $K(M,E,\Pi)_{(t,e,\pi)}$ denotes the coefficient in K associated with the pair $((M,E,\Pi),(t,e,\pi))$.

- $K(c_0, E, \emptyset)_{(c_0,e_0,\emptyset)} = 1$,
- $K(x,E,\Pi)_{(x,e,\pi)} = K(E(x),\Pi)_{(e(x),\pi)}$ if $x \in \text{Dom}(E)$ and e is such that $\forall y \neq x, e(y) = 1$,
- $K(\lambda x.M, E, \Gamma :: \Pi)_{(\lambda x.u,e,c::\pi)} = K(M, E_{x\mapsto\Gamma}, \Pi)_{(u,e_{x\to c},\pi)}$ if $e(x) = 1$ and where w.l.o.g $x \notin Dom(E)$,

[2] This machine has been implemented and is available online at `http://allioux.iiens.net/taylor/`.

[3] The size of a term is its number of symbols.

- $K((M)N,E,\Pi)_{(\langle t\rangle T,e,\pi)} = \sum_{\substack{(e',e'')\\ e'e''=e}} K(M,E,(N,E)::\Pi)_{(t,e',(T,e'')::\pi)}$,

The major difference with the case of the ordinary λ-calculus appears in the following two cases:

- $K(\alpha M,E,\Pi)_{(t,e,\pi)} = \alpha K(M,E,\Pi)_{(t,e,\pi)}$,
- $K(M+N,E,\Pi)_{(t,e,\pi)} = K(M,E,\Pi)_{(t,e,\pi)} + K(N,E,\Pi)_{(t,e,\pi)}$,
- Otherwise $K(M,E,\Pi)_{(t,e,\pi)} = 0$.

As we do not want to deal with states directly, we hide them by defining $\hat{K}$ which takes terms instead of states.

Definition 4.2. For any algebraic term M and any resource term t,

$$\hat{K}(M)_t = K(M,\emptyset,\emptyset)_{(t,e_0,\emptyset)} \tag{5}$$

This machine is defined for all semirings and in the particular case of $\mathbb{Q}^+$ computes a coefficient we shall characterize in Theorem 4.5.

We shall give some examples of execution. Let $\Delta = \lambda x.(x)x, I = \lambda x.x, T = \lambda xy.x$ and $F = \lambda xy.y$. Consider the two examples $(\Delta)Ic_0$ and $(\Delta)(pI+qF)c_0$, where $p,q \in \mathbb{S}$.

$$\hat{K}((\Delta)Ic_0) = \begin{cases} \langle\langle\lambda x.\langle x\rangle x\rangle(\lambda x.x)^2\rangle c_0 \mapsto 1 \\ \_ \mapsto 0 \end{cases}$$

Table 2 exposes the succession of states taken by the machine which are associated with a nonzero coefficient during the execution of this example. In fact, in this very case all the states have the coefficient 1 in K. We shall detail the transition from the 4^{th} to the 5^{th} state as this is the only one which involves a sum with several summands even though only one of these summands is nonzero.

Let $\mathbf{S}_1$ be the algebraic state $((x)x,\{x\mapsto(\lambda x.x,\emptyset)\},[(c_0,\emptyset)])$ and let $\mathbf{S}_2$ be the algebraic state $(x,\{x\mapsto(\lambda x.x,\emptyset)\},[(x,\{x\mapsto(\lambda x.x,\emptyset)\});(c_0,\emptyset)])$.

Then the transition from the 4^{th} to the 5^{th} state in Table 2 given by Definition 4.1 is:

$$\begin{aligned} K(\mathbf{S}_1)_{(\langle x\rangle x,\{x\mapsto((\lambda x.x)^2,e_0)\},[(c_0,e_0)])} = {} & K(\mathbf{S}_2)_{(x,\{x\mapsto(\lambda x.x,e_0)\},[(x,\{x\mapsto(\lambda x.x,e_0)\});(c_0,e_0)])} \\ & + K(\mathbf{S}_2)_{(x,\{x\mapsto 1\},[(x,\{x\mapsto((\lambda x.x)^2,e_0)\});(c_0,e_0)])} \\ & + K(\mathbf{S}_2)_{(x,\{x\mapsto((\lambda x.x)^2,e_0)\},[(x,\{x\mapsto 1\});(c_0,e_0)])} \end{aligned}$$

But both $K(\mathbf{S}_2)_{(x,\{x\mapsto 1\},[(x,\{x\mapsto((\lambda x.x)^2,e_0)\});(c_0,e_0)])}$ and $K(\mathbf{S}_2)_{(x,\{x\mapsto((\lambda x.x)^2,e_0)\},[(x,\{x\mapsto 1\});(c_0,e_0)])}$ are equal to 0 according to Definition 4.1.

That is why we only show the pair of states $(\mathbf{S}_2,(x,\{x\mapsto(\lambda x.x,e_0)\},[(x,\{x\mapsto(\lambda x.x,e_0)\});(c_0,e_0)]))$ in Table 2.

We will not give the full breakdown of the execution of the machine for the next example.

$$\hat{K}((\Delta)(pI+qF)c_0) = \begin{cases} \langle\lambda x.\langle x\rangle x\rangle I^2\rangle c_0 \mapsto p^2 \\ \langle\lambda x.\langle x\rangle 1\rangle F\rangle c_0 \mapsto q \\ \_ \mapsto 0 \end{cases}$$

There are two non-deterministic reductions of $(\Delta)(pI+qF)c_0$ which lead to c_0. The first one with multiplicity p^2 and the second one with multiplicity q which correspond to the two non-deterministic choices induced by the sum $pI+qF$.

Algebraic state			Resource state		
Term	Env.	Stack	Term	Env.	Stack
$((\lambda x.(x)x)\lambda x.x)c_0$	$\emptyset$	$[]$	$\langle\langle\lambda x.\langle x\rangle x\rangle(\lambda x.x)^2\rangle c_0$	e_0	$[]$
$(\lambda x.(x)x)\lambda x.x$	$\emptyset$	$[(c_0,\emptyset)]$	$\langle\lambda x.\langle x\rangle x\rangle(\lambda x.x)^2$	e_0	$[(c_0,e_0)]$
$\lambda x.(x)x$	$\emptyset$	$[(\lambda x.x,\emptyset);(c_0,\emptyset)]$	$\lambda x.\langle x\rangle x$	e_0	$[((\lambda x.x)^2,e_0);(c_0,e_0)]$
$(x)x$	$\{x\mapsto(\lambda x.x,\emptyset)\}$	$[(c_0,\emptyset)]$	$\langle x\rangle x$	$\{x\mapsto((\lambda x.x)^2,e_0)\}$	$[(c_0,e_0)]$
x	$\{x\mapsto(\lambda x.x,\emptyset)\}$	$[(x,\{x\mapsto(\lambda x.x,\emptyset)\});(c_0,\emptyset)]$	x	$\{x\mapsto(\lambda x.x,e_0)\}$	$[(x,\{x\mapsto(\lambda x.x,e_0)\});(c_0,e_0)]$
$\lambda x.x$	$\emptyset$	$[(x,\{x\mapsto(\lambda x.x,\emptyset)\});(c_0,\emptyset)]$	$\lambda x.x$	e_0	$[(x,\{x\mapsto(\lambda x.x,e_0)\});(c_0,e_0)]$
x	$\{x\mapsto(x,\{x\mapsto(\lambda x.x,\emptyset)\})\}$	$[(c_0,\emptyset)]$	x	$\{x\mapsto(x,\{x\mapsto(\lambda x.x,e_0)\})\}$	$[(c_0,e_0)]$
x	$\{x\mapsto(\lambda x.x,\emptyset)\}$	$[(c_0,\emptyset)]$	x	$\{x\mapsto(\lambda x.x,e_0)\}$	$[(c_0,e_0)]$
$\lambda x.x$	$\emptyset$	$[(c_0,\emptyset)]$	$\lambda x.x$	e_0	$[(c_0,e_0)]$
x	$\{x\mapsto(c_0,\emptyset)\}$	$[]$	x	$\{x\mapsto(c_0,e_0)\}$	$[]$
c_0	$\emptyset$	$[]$	c_0	e_0	$[]$

Table 2: Breakdown of the execution of the Krivine machine

4.2 Taylor expansion

In this setting we choose to restrict $\mathbb{S}$, the semiring over which is defined our algebraic λ-calculus, to any semiring having a multiplicative inverse such as $\mathbb{Q}^+$. Taylor expanding an algebraic term then comes down to expanding its applications according to the following formula:

$$((P)Q)^* = \sum_{n=0}^{\infty} \frac{1}{n!}\langle P^*\rangle Q^{*n} \tag{6}$$

where M^* denotes the Taylor expansion of the algebraic term M and where Q^{*n} is the sum of multisets of cardinality n whose elements are in the support of Q^* associated with a coefficient we will not detail here but which can be found in the report.

We justify the terminology "Taylor expansion" by pointing out that in analysis the Taylor series of an infinitely differentiable function f at 0 is $\sum_{n=0}^{\infty}\frac{1}{n!}f^{(n)}(0)x^n$. This is, indeed, quite similar to the form of the Taylor expansion of the application in the λ-calculus. See [2] for more details.

This operation can alternatively be defined by means of coefficients defined inductively on algebraic and resource terms. To this effect, we recall the coefficient m described in [4] accounting for the intrinsic contribution of a resource term t to its coefficient in the Taylor expansion of an algebraic term M and we introduce the weights w which account for the dependance in M of this coefficient.

Definition 4.3. The multiplicity m of a resource term t and the weight w of a resource term t in an algebraic term M are inductively defined as follows:

$$\begin{aligned} m(x) &= 1 \\ m(\lambda x.t) &= m(t) \\ m(\langle t\rangle T) &= m(t)\prod_{t\in\mathrm{supp}(T)} T(t)!\,m(t)^{T(t)} \end{aligned} \qquad \begin{aligned} w(x,x) &= 1 \\ w(\lambda x.t,\lambda x.M) &= w(t,M) \\ w(\langle t\rangle T,(M)N) &= w(t,M)\prod_{t\in\mathrm{supp}(T)} w(t,N)^{T(t)} \\ w(t,\alpha M) &= \alpha w(t,M) \\ w(t,M+N) &= w(t,M)+w(t,N) \end{aligned}$$

The coefficient $m(t)$ corresponds to the number of permutations of variable occurrences of t preserving the name of the variables and letting the term t unchanged. Finally, contrary to the case of the ordinary λ-calculus, the multiplicity of t in the Taylor expansion of M does not only depend on t but also depends on M. The weights w account for this phenomenon and represent one of the contributions of this paper.

We shall give some examples to enlighten the reader about these coefficients.

$$\begin{aligned} m(\langle \lambda x.x\rangle(\langle y\rangle z^3)^2) &= m(\lambda x.x)2!m(\langle y\rangle z^3)^2 \\ &= m(x)2(m(y)3!m(z)^3)^2 \\ &= 2*(3!)^2 \\ &= 2*36 = 72 \end{aligned}$$

As for the weights, their use is motivated by terms of the form $M+N$ and αM. Otherwise, if a term M is a pure λ-term and not an algebraic term then for any $t \in \Delta$, $w(t,M)$ is equal to 1 if $t \in M^*$ and 0 otherwise.

Consider the following example:

$$\begin{aligned} w(\langle x\rangle x^3, (x)(2x+y)+(x)(x+z)) &= w(\langle x\rangle x^3, (x)(2x+y)) + w(\langle x\rangle x^3, (x)(x+z)) \\ &= w(x,x)w(x,2x+y)^3 + w(x,x)w(x,x+z)^3 \\ &= w(x,x)(2w(x,x)+w(x,y))^3 + w(x,x)(w(x,x)+w(x,z))^3 \\ &= 2^3+1 = 9 \end{aligned}$$

Therefore, there are 9 ways to derive $\langle x\rangle x^3$ from $(x)(2x+y)+(x)(x+z)$.

Finally, the expression of the Taylor expansion can alternatively be given by the following definition:

Definition 4.4. (Taylor expansion) Given an algebraic term M, its Taylor expansion is:

$$M^* = \sum_{t\in\Delta} \frac{w(t,M)}{m(t)} t \tag{7}$$

It is easy to show this definition leads to an inductive definition of the Taylor expansion on the shape of algebraic terms which is compatible with Equation 6. This motivates our terminology.

4.3 Connection between the qKAM and the Taylor expansion

The following theorem, which is one of the main contributions of this paper, along with the definition of the **qKAM**, links the behaviour of the **qKAM** with the Taylor expansion of algebraic terms.

Theorem 4.5. *For all algebraic terms $M \in \Lambda_{\mathbb{S}}$, for all resource terms $t \in \Delta$ and provided that $\mathbb{S}$ has a multiplicative inverse,*

$$\hat{K}(M)_t = M^*_t \mathrm{NF}(t)_{c_0} \tag{8}$$

where M^*_t is the coefficient of t in M^* and $\mathrm{NF}(t)_{c_0}$ is the coefficient of c_0 in $\mathrm{NF}(t)$.

This theorem is a particular case of a more general result applying to any algebraic state and any resource state. It can be found in the report.

4.4 Computational complexity

At first, it seemed that Theorem 4.5 informed us of an efficient way to compute $K(M,\emptyset,\emptyset)$. Indeed, the Krivine machine reduces M to compute a subset of its Taylor expansion whereas the equation (8) gave hope we could obtain the same result more efficiently as the right-hand side does not involve the reduction of M. Although M^*_t can be computed statically by means of the coefficients m and w, it is folklore that determining $\mathrm{NF}(t)_{c_0}$ is **NP-complete**.

References

[1] Vincent Danos & Laurent Regnier (1999): *Reversible, irreversible and optimal λ-machines*. *Theoretical Computer Science* 227(1), pp. 79–97, doi:10.1016/S0304-3975(99)00049-3.

[2] Thomas Ehrhard & Laurent Regnier (2003): *The differential lambda-calculus*. *Theoretical Computer Science* 309(1-3), pp. 1–41, doi:10.1016/S0304-3975(03)00392-X. Available at https://hal.archives-ouvertes.fr/hal-00150572. 41 pages.

[3] Thomas Ehrhard & Laurent Regnier (2006): *Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms*. In A. Beckmann, U. Berger, B. Löwe & J.V. Tucker, editors: *Second Conference on Computability in Europe, CiE 2006*, LNCS 3988, Springer Berlin / Heidelberg, Swansea, United Kingdom, pp. 186–197, doi:10.1007/11780342_20. Available at https://hal.archives-ouvertes.fr/hal-00150273. 12 pages.

[4] Thomas Ehrhard & Laurent Regnier (2008): *Uniformity and the Taylor expansion of ordinary lambda-terms*. *Journal of Theoretical Computer Science* 403(2-3), pp. 347–372, doi:10.1016/j.tcs.2008.06.001. Available at https://hal.archives-ouvertes.fr/hal-00150275.

[5] Lionel Vaux (2009): *The algebraic lambda calculus*. *Mathematical Structures in Computer Science* 19, pp. 1029–1059, doi:10.1017/S0960129509990089. Available at http://journals.cambridge.org/article_S0960129509990089.