Krivine machine and Taylor expansion of λ-terms in a non-uniform setting [1]

Antoine Allioux[2]

Institut de Recherche en Informatique Fondamentale
Paris, France

Abstract

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 [3]. 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

[1] Last revision: 3 October 2016

[2] Email: antoine.allioux@gmail.com

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

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

Table 1
Algebraic equalities of the algebraic λ-calculus

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.

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.

2.1 Grammar

We shall follow the presentation of the algebraic λ-calculus given in [8].

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

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

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 [8].

2.2 The Krivine abstract machine

The Krivine machine [5] is an abstract machine performing the weak head linear reduction on untyped λ-terms. The machine K we shall define will implement the

head linear reduction on algebraic terms with the restriction that they reduce to a multiple of the constant c_0 which is a new term we are adding to the grammar of the algebraic λ-calculus for simplicity reasons. 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.

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 the closure 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 $\mathrm{FV}(M) \subseteq \mathrm{Dom}(E)$ where $\mathrm{FV}(M)$ denotes the free variables of M and $\mathrm{Dom}(E)$ 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 head linear reduction, we reduce according to the structure of the first closure so we give it a special status. The set of the algebraic states is denoted $\mathcal{S}(\Lambda_{\mathbb{S}})$.

A state is a snapshot of the abstract machine at a given time and represents the dissection of a *unique* λ-term. It is possible to recover this λ-term and we define a function $\mathrm{T} : \mathcal{S}(\Lambda_{\mathbb{S}}) \to \Lambda_{\mathbb{S}}$ doing precisely that.

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

$$\mathrm{T}(M, E) = M[\mathrm{T}(E(x))/x]_{x\in\mathrm{Dom}(E)} \tag{2}$$

$$\mathrm{T}(M, E, (\Gamma_1, \ldots, \Gamma_n)) = (\ldots(\mathrm{T}(M, E))\mathrm{T}(\Gamma_1)\ldots)\mathrm{T}(\Gamma_n) \tag{3}$$

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 the pair (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.

- $K_0(M, E, \Pi) = \mathbf{0}$,
- $K_{n+1}(c_0, E, \emptyset) = c_0$,
- $K_{n+1}(x, E, \Pi) = K_n(E(x), \Pi)$ 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:

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

And finally $K = \lim_{n\to\infty} Kx_n$.

3 Resource lambda calculus

We recall the syntax and the reduction of the resource λ-calculus Δ which has been defined in [4]. Indeed, we will define the Taylor expansion of an algebraic term in terms of a sum of resource λ-terms. This flavour of λ-calculus is linear in the argument in the sense that during the reduction, the application has to utilize all its arguments once, no more, no less. This is why the argument of an application will be represented as a multiset of terms. Firstly we shall make explicit our notion of multiset.

3.1 Multisets

We define the set of finite multisets over a set I with coefficients in $\mathbb{N}$, $M_{\text{fin}}(I)$, to be $\mathbb{N}\langle I\rangle$: the free monoid over I. As an example, we denote the multiset composed of the element $s \in I$ with multiplicity $p \in \mathbb{N}$ and of the element $t \in I$ with multiplicity $q \in \mathbb{N}$ the following way: $s^p t^q$. For two multisets S and T, we refer to the multiset union using the multiplicative notation ST. Therefore we choose to denote the empty multiset by the multiplicative unit 1. As multisets can also be seen as total functions from I to $\mathbb{N}$, for a multiset T and an element $t \in I$ we will write $T(t)$ to refer to the multiplicity of t in T. The cardinal of T, denoted $|T|$, is equal to $\sum_{t\in\text{supp}(T)} T(t)$. We denote $M_n(I)$ the restriction of $M_{\text{fin}}(I)$ to multisets whose cardinal is equal to n. The set of elements of T whose multiplicity is nonzero is called the support of T and is denoted $\text{supp}(T)$. Finally, we define the multinomial coefficient of T as follows:

$$[T] = \frac{|T|!}{\prod_{t\in\text{supp}(T)} T(t)!}$$

This coefficient is the number of distinct enumerations of the occurrences of the elements of T.

For example, $[a^2b] = \dfrac{3!}{2!1!} = 3$ and $[abc] = 3! = 6$.

3.2 Grammar

The resource λ-calculus Δ shares its grammar with the ordinary λ-calculus with the exception that the application takes multisets of terms. We denote it using angle brackets instead of parentheses.

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

$$\Delta : s, t, u, \ldots ::= x \mid \lambda x.t \mid \langle t\rangle S \quad , \tag{4}$$

where $x, y, \cdots \in \mathcal{V}$, 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 $s, t, \ldots$ The set of finite multisets of simple terms is defined to be $M_{\text{fin}}(\Delta)$ and we denote it $\Delta^!$. We call its elements simple *poly-terms* and we refer to

them using upper case letters $S, T, \ldots$ When a term t can either be a simple term or a simple poly-term we will say it belongs to $\Delta^{(!)} = \Delta \cup \Delta^!$.

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

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 cannot be expressed in the syntax of the resource λ-calculus contrarily to the algebraic λ-calculus. We refer to (poly-)terms using the letters $\mathcal{S}, \mathcal{T}, \ldots$ and we define the support $\mathrm{supp}(\mathcal{S})$ to be the set of simple (poly-)terms whose coefficient is not null in $\mathcal{S}$. We refer to the coefficient of s in $\mathcal{S}$ by the notation $\mathcal{S}_s$.

Finally we extend the grammar of the resource λ-calculus to all (poly-)terms by linearity: Let $\mathcal{S} \in \mathbb{S}\langle \Delta \rangle$ and let $\mathcal{T} \in \mathbb{S}\langle \Delta^! \rangle$, $\lambda x.\mathcal{S} = \sum_{s \in \mathrm{supp}(\mathcal{S})} \mathcal{S}_s \lambda x.s$, $\langle \mathcal{S} \rangle \mathcal{T} = \sum_{s \in \mathrm{supp}(\mathcal{S}), T \in \mathrm{supp}(\mathcal{T})} \mathcal{S}_s \mathcal{T}_T \langle s \rangle T$ and $\mathcal{S}T = \sum_{s \in \mathrm{supp}(\mathcal{S})} \mathcal{S}_s sT$ where $\sum$ is the sum of the $\mathbb{S}$-module. We give some examples to illustrate these rules: $\lambda x.(t+u) = (\lambda x.t) + (\lambda x.u)$, $\langle s + t \rangle T = \langle s \rangle T + \langle t \rangle T$ and $(s+t)s^2t = s^3t + s^2t^2$.

3.3 Linear substitution and reduction

The linear substitution is based on the notion that substituting terms must be used once and only once. As an example, take the following ordinary substitution: $M[N/x]$. N can be copied as many times as there are occurrences of x in M – possibly none. In the resource λ-calculus we forbid the copy or the deletion of any substituting term during the substitution. This raises the question of the allocation of the resource if there are more than one occurrence of a variable to substitute. As we shall see, the linear substitution form the sums of all the possible allocations.

For any simple terms s and t, the partial substitution of x in s by t is denoted $\dfrac{\partial s}{\partial x} \cdot t$ and is defined as follows:

- $\dfrac{\partial y}{\partial x} \cdot t = \begin{cases} t & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$,
- $\dfrac{\partial(\lambda y.s)}{\partial x} \cdot t = \lambda y. \left(\dfrac{\partial s}{\partial x} \cdot t \right)$ assuming $x \neq y$ and y not free in t,
- $\dfrac{\partial \langle s \rangle T}{\partial x} \cdot t = \left\langle \dfrac{\partial s}{\partial x} \cdot t \right\rangle T + \langle s \rangle \dfrac{\partial T}{\partial x} \cdot t$,
- $\dfrac{\partial sT}{\partial x} \cdot t = \left(\dfrac{\partial s}{\partial x} \cdot t \right) T + s \left(\dfrac{\partial T}{\partial x} \cdot t \right)$.

We denote the composition of partial substitutions as follows:

$$\frac{\partial^n s}{\partial x^n} \cdot (t_1, \ldots, t_n) = \frac{\partial}{\partial x} \left(\ldots \left(\frac{\partial s}{\partial x} \cdot t_1 \right) \ldots \right) \cdot t_n$$

When the x's do not appear free in the t_i's, the result does not depend on the order of the substitutions and we forget the commas: $\frac{\partial^n s}{\partial x^n} \cdot (t_1 \ldots t_n)$. This partial substitution is extended to all finite (poly-)terms by linearity.

From this elementary substitution, we derive the linear substitution $\partial_x(s, T)$ which returns the iterated substitutions of the free occurrences of x in s by the elements of T if, and only if, the cardinal of the bag T is equal to the number of free occurrences of the variable x in s. Otherwise the result is 0, the additive unit of $\mathbb{S}\langle\Delta\rangle$. It is defined as follows:

$$\partial_x(s, t_1 \dots t_n) = \frac{\partial^n s}{\partial x^n} \cdot (t_1 \dots t_n)[0/x] \tag{5}$$

We extend this notation to the linear substitution of several variables:

$$\partial_{x_1,\dots,x_n}(s, T_1, \dots, T_n) = \partial_{x_n}(\dots \partial_{x_1}(s, T_1), \dots, T_n) \tag{6}$$

This substitution does not depend on the order of the iterated substitutions as the variables $x_1, x_2, \dots$ 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 it reduces as follows:

$$\langle\lambda x.s\rangle T \to_\beta \partial_x(s, T) \tag{7}$$

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 $s \to_\beta \mathcal{S}$ with $s \in \Delta$ and $\mathcal{S} \in \mathbb{S}\langle\Delta\rangle$:

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

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.

3.4 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 $\mathcal{V}$ to resource closures. e_0 is the empty environment mapping any variable to the 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 variable 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 $1 \coloneqq (1, e_0)$, e_0 being the empty environment. We use letters $c, c_1, \dots$ for general resource closures and $\gamma, \gamma_1, \dots$ 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. The set of resource states is denoted $\mathcal{S}(\Delta)$.

We also define $\mathrm{T_D} : \mathcal{S}(\Delta) \to \mathbb{S}\langle\Delta^{(!)}\rangle$ similarly to T (see (2)) on resource closures and extends it to resource states. For any resource closure (T, e) and any resource

state $(t, e, (c_1, \ldots, c_n))$:

$$\mathrm{T_D}(T, e) = \partial_{x_1,\ldots,x_n}(T, \mathrm{T_D}(e(x_1)), \ldots, \mathrm{T_D}(e(x_n))) \tag{8}$$

$$\mathrm{T_D}(t, e, (c_1, \ldots, c_n)) = \langle \ldots \langle \mathrm{T_D}(t, e) \rangle \mathrm{T_D}(c_1) \ldots \rangle \mathrm{T_D}(c_n) \tag{9}$$

4 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 $\mathbb{Q}^+$ as we need it to express the Taylor expansion. Taylor expanding an algebraic term then comes down to expanding its applications according to the following formula:

$$((P)Q)^* = \sum_{n=0}^{\infty} \frac{1}{n!} \langle P^* \rangle Q^{*n} \tag{10}$$

where M^* denotes the Taylor expansion of the algebraic term M and where

$$Q^{*n} = \sum_{T \in M_n(\mathrm{supp}(Q^*))} [T] \left(\prod_{t \in \mathrm{supp}(T)} Q_t^{*T(t)} \right) T \tag{11}$$

The coefficient of T in Q^{*n}, namely $[T] \prod_{t \in \mathrm{supp}(T)} Q_t^{*T(t)}$, corresponds to the number of distinct enumerations of the elements t of T weighted by the multiplicity of each t in Q^*.

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.

An alternative definition of the Taylor expansion will be given in the next sections by inductively defining the coefficient of a resource term in the Taylor expansion of an algebraic term.

4.1 Weights and multiplicities

We shall define the coefficient of a resource term in the Taylor expansion of an algebraic term. To this effect, we recall the coefficient m described in [4] accounting for the intrinsic contribution of t to its coefficient in the Taylor expansion of an algebraic term M and we introduce the weights w which accounts for the dependance in M of this coefficient.

Definition 4.1 The multiplicity of a resource term is inductively defined as follows:

- $m(x) = 1$
- $m(\lambda x.t) = m(t)$
- $m(\langle t \rangle T) = m(t) \prod_{t \in \mathrm{supp}(T)} T(t)! m(t)^{T(t)}$

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 on M. The weights w account for this phenomenon and is one of the contributions of this report.

Definition 4.2 Let M be an algebraic term and let t be a resource term, the weight of t in the Taylor expansion of M is noted $w(t, M)$. It is inductively defined as follows:

- $w(x, x) = 1$
- $w(\lambda x.t, \lambda x.M) = w(t, M)$
- $w(\langle t\rangle T, (M)N) = w(t, M) \prod_{t \in \text{supp}(T)} w(t, N)^{T(t)}$
- $w(t, \alpha M) = \alpha w(t, M)$
- $w(t, M + N) = w(t, M) + w(t, N)$

It can be noted that the coefficient $w(t, M)$ is nonzero if and only if $t \in M^*$.

These coefficients defined, we can define the Taylor expansion of any algebraic term.

4.2 Generalized expression

The general expression of the Taylor expansion is defined inductively on algebraic terms as follows:

Definition 4.3 [Taylor expansion] The Taylor expansion of any algebraic term in $\Lambda_{\mathbb{S}}$ is defined as follows:

- $x^* = x$
- $(\lambda x.M)^* = \lambda x.M^*$
- $((M)N)^* = \sum_{n=0}^{\infty} \frac{1}{n!} \langle M^*\rangle N^{*n}$
- $(\alpha M)^* = \alpha M^*$
- $(M + N)^* = M^* + N^*$

The sum induced when Taylor expanding the application is well-defined. This has been proved in [4].

This definition leads to the following lemma giving a general form for the Taylor expansion using the coefficients m and w.

Lemma 4.4 (Taylor expansion)

$$M^* = \sum_{t \in \Delta} \frac{w(t, M)}{m(t)} t \tag{12}$$

Proof. We will prove the only non-obvious case which is the one of the application by induction on $\langle P^*\rangle Q^{*n}$.

$$
\begin{aligned}
&\sum_{n=0}^{\infty} \frac{1}{n!} \langle P^* \rangle Q^{*n} \\
&\quad \overset{(12)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \left\langle \sum_{s \in \Delta} \frac{w(s,P)}{m(s)} s \right\rangle \left(\sum_{t \in \Delta} \frac{w(t,Q)}{m(t)} t \right)^n \\
&\quad \overset{(11)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \left\langle \sum_{s \in \Delta} \frac{w(s,P)}{m(s)} s \right\rangle \left(\sum_{T \in M_n(\Delta)} [T] \left(\prod_{t \in \mathrm{supp}(T)} \left(\frac{w(t,Q)}{m(t)} \right)^{T(t)} \right) T \right) \\
&\quad = \sum_{n=0}^{\infty} \sum_{s \in \Delta} \sum_{T \in M_n(\Delta)} \frac{1}{n!} \frac{w(s,P)}{m(s)} [T] \left(\prod_{t \in \mathrm{supp}(T)} \frac{w(t,Q)^{T(t)}}{m(t)^{T(t)}} \right) \langle s \rangle T \\
&\quad = \sum_{s \in \Delta} \sum_{T \in M_{\mathrm{fin}}(\Delta)} \frac{w(s,P)}{m(s)} \left(\prod_{t \in \mathrm{supp}(T)} \frac{w(t,Q)^{T(t)}}{T(t)! m(t)^{T(t)}} \right) \langle s \rangle T \\
&\quad \overset{(4.2)}{\overset{(4.1)}{=}} \sum_{\langle s \rangle T \in \Delta} \frac{w(\langle s \rangle T, (P)Q)}{m(\langle s \rangle T)} \langle s \rangle T \\
&\quad \overset{(12)}{=} ((P)Q)^*
\end{aligned}
$$

□

Taylor expanding λ-terms can be compared to denotational semantics as it transforms terms enjoying a rich dynamics and potentially not strongly normalizing to a possibly infinite sum of resource terms having a much restricted dynamics. The difference with denotational semantic being that resource terms still have a dynamics albeit one enjoying different properties like finiteness as we know that resource terms are strongly normalizing.

4.3 Taylor expansion of algebraic states

In order to prove some properties of the quantitative Krivine machine we are going to introduce in the next section, we need to define the Taylor expansion of algebraic states which should be consistent with the Taylor expansion of its corresponding algebraic term.

We give a quantitative definition in the form of a characterization of the multiplicity of a resource state in the Taylor expansion of an algebraic state.

We extend environments to environments mapping variables to sums of closures as follows: $e[x \mapsto c_1 + c_2] = e[x \mapsto c_1] + e[x \mapsto c_2]$. We extend resource closures to all (poly-)terms and to sums of environments by linearity: $(s + t, e) = (s, e) + (t, e)$ and $(s, e_1 + e_2) = (s, e_1) + (s, e_2)$. We extend stack of closures to stack of sum of closures as follows: $(c_1 + c_2) :: \pi = c_1 :: \pi + c_2 :: \pi$. Finally we extend states to states of sum of closures and sum of stacks: $(c_1 + c_2, \pi) = (c_1, \pi) + (c_2, \pi)$ and $(c, \pi_1 + \pi_2) = (c, \pi_1) + (c, \pi_2)$.

Definition 4.5 (Taylor expansion of algebraic states)

- Given an environment E:

$$E^* = \prod_{x \in \mathrm{Dom}(E)} [x \mapsto E(x)^*] \tag{13}$$

- Given an algebraic closure (M, E),

$$(M, E)^* = \sum_{n=0}^{\infty} \frac{1}{n!} (M^{*n}, E^*) \tag{14}$$

- Expansion of an algebraic stack:

$$\emptyset^* = \emptyset \tag{15}$$
$$(\Gamma :: \Pi)^* = \Gamma^* :: \Pi^* \tag{16}$$

- Finally, given an algebraic state (M, E, Π):

$$(M, E, \Pi)^* = (M^*, E^*, \Pi^*) \tag{17}$$

We also define the following notations $(M, E)^*_{(T,e)} = M^*_T E^*_e$ and $(M, E, \Pi)^*_{(t,e,\pi)} = M^*_t E^*_e \Pi^*_\pi$ where for any sum S, S_t is the coefficient of t in S.

It remains to show the correctness of this definition, namely that

$$\mathrm{T_D}((M, E)^*) = \sum_{n=0}^{\infty} \frac{1}{n!} \mathrm{T}(M, E)^{*n} \tag{18}$$

which will be the object of Lemma 4.7 in which $\mathrm{T_D}$ will be extended to sums of resource closures.

But in order to prove this lemma, we need the following preliminary lemma:

Lemma 4.6

$$(M[N/x])^* = \sum_{n=0}^{\infty} \frac{1}{n!} \partial_x (M^*, N^{*n}) \tag{19}$$

Proof. (Sketch) First, ∂_x is well-defined on linear combinations of resource terms. This result stems from Lemma 15 in [4]. Indeed, a consequence of this lemma is that there are only finitely many $s \in \mathrm{supp}(M^*)$ such that for any $t \in \Delta, t \in \mathrm{supp}(\partial_x(s, N^{*n}))$.

This lemma is proved by structural induction on M but we only detail the case $M = (P)Q$. In this case we use the fact that $\langle \_ \rangle (\_)^k$ is $k+1$-linear and the generalization of the Leibniz rule to affirm that the following equality holds:

$$\begin{aligned} \partial_x(\langle P^* \rangle Q^{*k}, N^{*n}) = \sum_{\substack{p_0, \ldots, p_k \in \mathbb{N} \\ p_1 + \cdots + p_k = n}} \binom{n}{p_0 \ldots p_k} \langle \partial_x(P^*, N^{*p_0}) \rangle \\ (\partial_x(Q^*, N^{*p_1}) \ldots \partial_x(Q^*, N^{*p_k})) \end{aligned} \tag{20}$$

We now prove the lemma:

$$
\begin{aligned}
&\sum_{n=0}^{\infty} \frac{1}{n!} \partial_x(((P)Q)^*, N^{*n}) \\
&\quad \overset{(4.3)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{k!} \partial_x(\langle P^* \rangle Q^{*k}, N^{*n}) \\
&\quad \overset{(20)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{k!} \sum_{\substack{p_0,\dots,p_k \in \mathbb{N} \\ p_0+\cdots+p_k=n}} \binom{n}{p_0 \dots p_k} \langle \partial_x(P^*, N^{*p_0}) \rangle (\partial_x(Q^*, N^{*p_1}) \\
&\qquad\qquad \dots \partial_x(Q^*, N^{*p_k})) \\
&\quad = \sum_{k=0}^{\infty} \sum_{p_0,\dots,p_k \in \mathbb{N}} \frac{1}{k! p_0! \dots p_k!} \langle \partial_x(P^*, N^{*p_0}) \rangle (\partial_x(Q^*, N^{*p_1}) \dots \partial_x(Q^*, N^{*p_k})) \\
&\quad \overset{(\text{I.H.})}{=} \sum_{k=0}^{\infty} \frac{1}{k!} \langle P[N/x]^* \rangle Q[N/x]^{*k} \\
&\quad \overset{(4.3)}{=} ((P[N/x])Q[N/x])^* \\
&\quad = ((P)Q)[N/x]^*
\end{aligned}
$$

□

This permits to prove the correctness of our definition of the Taylor expansion of an algebraic closure.

Lemma 4.7 *For any algebraic closure* (M, E)*,*

$$\mathrm{T_D}((M, E)^*) = \sum_{n=0}^{\infty} \frac{1}{n!} T(M, E)^{*n} \tag{21}$$

Proof. (Sketch) For the same reasons as stated in the proof of Lemma 4.6, $\mathrm{T_D}$ is defined on linear combinations of resource terms thanks to Lemma 15 in [4]. This is due to the fact that $\mathrm{T_D}$ is just syntaxic sugar for ∂_x.

Similarly to the previous case, we use the fact that $(\_)^k$ is k-linear and the generalization of the Leibniz rule to affirm the following equality holds:

$$\partial_x(M^{*n}, S^p) = \sum_{\substack{p_1,\dots,p_n \in \mathbb{N} \\ p_1+\cdots+p_n=p}} \binom{p}{p_1 \dots p_k} (\partial_x(M^*, S^{p_1}) \dots \partial_x(M^*, S^{p_k})) \tag{22}$$

We perform our induction on E:

- If $E = \emptyset$ the result obviously holds.
- If $E = [x \mapsto \Gamma]$,

$$
\begin{aligned}
&\mathrm{T_D}((M, E)^*) \\
&\quad \overset{(14)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \mathrm{T_D}(M^{*n}, E^*)
\end{aligned}
$$

$$\overset{(8)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \partial_x(M^{*n}, \mathrm{T_D}(E^*(x)))$$
$$\overset{(\text{I.H.})}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \partial_x(M^{*n}, \sum_{p=0}^{\infty} \frac{1}{p!} \mathrm{T}(E(x))^{*p})$$
$$= \sum_{n=0}^{\infty} \sum_{p=0}^{\infty} \frac{1}{n!} \frac{1}{p!} \sum_{\substack{p_1,\ldots,p_n \in \mathbb{N} \\ p_1+\cdots+p_n=p}} \binom{p}{p_1 \ldots p_n} (\partial_x(M^*, \mathrm{T}(E(x))^{*p_1}) \ldots \partial_x(M^*, \mathrm{T}(E(x))^{*p_n}))$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{p_1,\ldots,p_n \in \mathbb{N}} \frac{1}{p_1! \ldots p_n!} (\partial_x(M^*, \mathrm{T}(E(x))^{*p_1}) \ldots \partial_x(M^*, \mathrm{T}(E(x))^{*p_n}))$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!} (\sum_{p_1 \in \mathbb{N}} \frac{1}{p_1!} \partial_x(M^*, \mathrm{T}(E(x))^{*p_1}) \cdots \sum_{p_n \in \mathbb{N}} \frac{1}{p_n!} \partial_x(M^*, \mathrm{T}(E(x))^{*p_n}))$$
$$\overset{(19)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} (\overbrace{M[E(x)/x]^* \ldots M[E(x)/x]^*}^{n \text{ times}})$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!} M[E(x)/x]^{*n}$$
$$\overset{(2)}{=} \sum_{n} \frac{1}{n!} \mathrm{T}(M, E)^{*n}$$

- If $E = E'[x_1 \mapsto \Gamma]$,

$$\mathrm{T_D}((M, E)^*)$$
$$\overset{(14)}{=} \sum_{n=0}^{\infty} \frac{1}{n!} \mathrm{T_D}(M^{*n}, E^*)$$
$$\overset{(8)}{=} \sum_{n} \frac{1}{n!} \partial_{x_1,\ldots,x_k}(M^{*n}, \mathrm{T_D}(E^*(x_1)), \ldots, \mathrm{T_D}(E^*(x_k)))$$
$$= \sum_{n} \frac{1}{n!} \partial_{x_2,\ldots,x_k}(\partial_{x_1}(M^{*n}, \mathrm{T_D}(E^*(x_1))), \mathrm{T_D}(E^*(x_2)), \ldots, \mathrm{T_D}(E^*(x_k)))$$
$$= \sum_{n} \frac{1}{n!} \partial_{x_2,\ldots,x_k}(M[E(x_1)/x_1]^{*n}, \mathrm{T_D}(E^*(x_2)), \ldots, \mathrm{T_D}(E^*(x_k)))$$
$$\overset{(\text{I.H.})}{=} \sum_{n} \frac{1}{n!} M[E(x_1)/x_1, \ldots, E(x_k)/x_k]^{*n}$$
$$\overset{(2)}{=} \sum_{n} \frac{1}{n!} \mathrm{T}(M, E)^{*n}$$

□

It is easy to generalize this result to algebraic states in order to show:

$$\mathrm{T_D}((M, E, (\Gamma_1, \ldots, \Gamma_n))^*) = \langle \mathrm{T}(M, E) \rangle \left(\sum_{k=0}^{\infty} \frac{1}{k!} \mathrm{T}(\Gamma_1)^{*k} \right) \cdots \left(\sum_{k=0}^{\infty} \frac{1}{k!} \mathrm{T}(\Gamma_n)^{*k} \right)$$

The head closure has a particular status, this is a particular case of the equa-

tion (21) where n is fixed to 1.

5 Quantitative Krivine machine (qKAM)

5.1 Welcome to the machine

We shall define our quantitative Krivine machine **(qKAM)** K which draws its inspiration from the one described in [3]. We shall adapt this machine in order to handle the algebraic λ-calculus. In particular this machine will deal with the coefficients of multiplicity which were not implemented by the previously mentioned machine. It will, given an algebraic term and a resource term, output the multiplicity of this resource term in the Taylor expansion of the algebraic term modulo a coefficient linked to the reduction of the algebraic term.

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 which we also define to be equal to its own Taylor expansion.

In the first place we need to establish a measure on resource states then characterize its evolution according to the transitions of K. This will enable us to do inductions on the size of resource terms.

Definition 5.1 (Measure on resource states) We define a measure on resource states (t, e, π) inductively on their shape such that $\texttt{size}(t, e, \pi) = \texttt{size}(t) + \texttt{size}(e) + \texttt{size}(\pi)$.

- Size of a resource term:
  - $\texttt{size}(x)\texttt{=1}$
  - $\texttt{size}(\lambda x.N)\texttt{=1+size}(N)$
  - $\texttt{size}(\langle t \rangle T)\texttt{=1+size}(t)\texttt{+size}(T)$
  - $\texttt{size}(t_1 \dots t_n)\texttt{=}\sum_{i=1}^{n} \texttt{size}(t_i)$
- Size of a resource environment:
  - $\texttt{size}(e_0)\texttt{=}0$
  - $\texttt{size}(e)\texttt{=}\sum_{x \in \mathrm{Dom}(e)} \texttt{size}(\mathrm{e(x)})$
- Size of a resource stack:
  - $\texttt{size}([])\texttt{=}0$
  - $\texttt{size}(\gamma :: \pi) \texttt{ = size}(\gamma) \texttt{ + size}(\pi)$

Theorem 5.2 *The following results link the size of the resource states (t, e, π) with the ones corresponding to the transitions of K.*

- $\texttt{size}(c_0, e_0, \emptyset) = 1$*;*
- $\texttt{size}(x, e, \pi) = \texttt{size}(e(x), \pi)+1$*, assuming e is such that* $\forall y, y \neq x \implies e(y) = 1$*;*
- $\texttt{size}(\lambda x.u, e, \gamma :: \pi') = \texttt{size}(u, e_{x \mapsto \gamma}, \pi') + 1$*, assuming* $e(x) = 1$*;*
- $\texttt{size}(\langle t \rangle T, e'e'', \pi) = \texttt{size}(t, e', (T, e'') :: \pi) + 1$.

Proof. Easy proof using the definition 5.1. □

This theorem proves that the size of a resource term strictly decreases during the transitions of K which we are going to introduce in the next definition. This

allows us to perform inductions on the size of resource terms.

We are now able to define the **qKAM** K [4] which is the main contribution of this report.

Definition 5.3 (Quantitative Krivine machine)

The quantitative Krivine machine is defined as a matrix $K \in \mathbb{S}^{\mathcal{S}(\Lambda_{\mathbb{S}})\times\mathcal{S}(\Delta)}$. It is defined by induction on the pair $(\mathtt{size}(t,e,\pi), \mathtt{size}(M,E,\Pi))$ lexicographically ordered. The coefficient in K associated with the pair $((M,E,\Pi),(t,e,\pi))$ is denoted $K(M,E,\Pi)_{(t,e,\pi)}$.

- If $(M,E,\Pi) = (c_0,E,\emptyset)$ and $(t,e,\pi) = (c_0,e_0,\emptyset)$,

$$K(c_0,E,\emptyset)_{(c_0,e_0,\emptyset)} = 1; \tag{23}$$

- If $(M,E,\Pi) = (x,E,\Pi)$ and $(t,e,\pi) = (x,e,\pi)$ where $x \in \mathrm{Dom}(E)$ and e is such that $\forall y \neq x, e(y) = 1$,

$$K(x,E,\Pi)_{(x,e,\pi)} = K(E(x),\Pi)_{(e(x),\pi)}; \tag{24}$$

- If $(M,E,\Pi) = (\lambda x.N, E, \Gamma :: \Pi')$ and $(t,e,\pi) = (\lambda x.u, e, c :: \pi')$ where $e(x) = 1$ and w.l.o.g. $x \notin \mathrm{Dom}(E)$,

$$K(\lambda x.N,E,\Gamma :: \Pi')_{(\lambda x.u,e,c::\pi')} = K(N,E_{x\mapsto\Gamma},\Pi')_{(u,e_{x\mapsto c},\pi')}; \tag{25}$$

- If $(M,E,\Pi) = (\alpha N,E,\Pi)$,

$$K(\alpha N,E,\Pi)_{(t,e,\pi)} = \alpha K(N,E,\Pi)_{(t,e,\pi)}; \tag{26}$$

The major difference with the case of the ordinary λ-calculus appears in the following two cases where sums appear.

- If $(M,E,\Pi) = ((P)Q,E,\Pi)$ and $(t,e,\pi) = (\langle t'\rangle T,e,\pi)$,

$$K((P)Q,E,\Pi)_{(\langle t'\rangle T,e,\pi)} = \sum_{\substack{(e',e'')\\ e'e''=e}} K(P,E,(Q,E)::\Pi)_{(t',e',(T,e'')::\pi)}; \tag{27}$$

- If $(M,E,\Pi) = (P+Q,E,\Pi)$,

$$K(P+Q,E,\Pi)_{(t,e,\pi)} = K(P,E,\Pi)_{(t,e,\pi)} + K(Q,E,\Pi)_{(t,e,\pi)}; \tag{28}$$

- Otherwise,

$$K(M,E,\Pi)_{(t,e,\pi)} = 0. \tag{29}$$

As a shorthand, we define $\hat{K}$ to be such that $\hat{K}(M)_t = K(M,\emptyset,\emptyset)_{(t,e_0,\emptyset)}$.

This machine is defined for all semirings and in the particular case of $\mathbb{Q}^+$ computes a coefficient we shall characterize in Theorem 5.9. It is not clear what the meaning of these coefficients is in other cases.

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

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

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 $p, q \in \mathbb{S}$.

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

The 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 the Definition 5.3 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 the Definition 5.3.

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 the Table 2.

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

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

In this case, the different ways to execute $(\Delta)(pI + qF)c_0$ lead to two resource terms having a different shape. 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 $pI + qF$ which lead to a non-zero normal form.

5.2 Connection between the qKAM and the Taylor expansion

We shall show that for any algebraic term M and for any simple term t, $\hat{K}(M) = M_t^* \mathrm{NF}(t)_{c_0}$.

But first we need to introduce some preliminary results.

Definition 5.4 (Binomial coefficients for resource closures and resource environments) We mutually define the binomial coefficients for resource environments and resource closures as follows: for all resource closures (S', e') and (S'', e''),

$$\binom{(S'S'', e'e'')}{(S', e')} \coloneqq \left(\prod_{s \in \mathrm{supp}(S'S'')} \frac{S'S''(s)!}{S'(s)!S''(s)!} \right) \binom{e'e''}{e'}$$
$$\binom{e'e''}{e'} \coloneqq \prod_{x \in \mathrm{Dom}(e'e'')} \binom{e'e''(x)}{e'(x)}$$

Lemma 5.5 *Let (S, e) be a resource closure and let $x_1, \ldots, x_n$ be the distinct free variables of S. Let $\hat{S}$ be a simple (poly-)term obtained from S by partitioning its free variables x_i's in x_i^1's and x_i^2's. This simple (poly-)term is such that $S = \hat{S}[x_i/x_i^1, x_i^2]_{i \in [\![1,n]\!]}$ and the following holds,*

$$\mathrm{T_D}(\hat{S}[x_i/x_i^1, x_i^2]_{i \in [\![1,n]\!]}, e) = \sum_{\substack{(e',e'') \\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\hat{S}[x_i/x_i^1]_{i \in [\![1,n]\!]}, e')[x_i/x_i^2]_{i \in [\![1,n]\!]}, e'') \tag{30}$$

Proof. We prove the result by induction on the size of e.

- If $e = e_0$ then the result obviously holds.
- If $e = [x \mapsto (tT, e)]$,

$$\begin{aligned}
&\mathrm{T_D}(\hat{S}[x/x^1, x^2], [x \mapsto (tT, e)]) \\
&\quad \overset{(8)}{=} \partial_x(\hat{S}[x/x^1, x^2], \mathrm{T_D}(tT, e)) \\
&\quad \overset{\text{(I.H.)}}{\overset{(5)}{=}} \sum_{\substack{(e',e'') \\ e'e''=e}} \binom{e}{e'} \partial_x \left(\frac{\partial \hat{S}[x/x^1, x^2]}{\partial x}.\mathrm{T_D}(t, e'), \mathrm{T_D}(T, e'') \right) \\
&\quad \overset{\text{(def.)}}{=} \sum_{\substack{(e',e'') \\ e'e''=e}} \binom{e}{e'} \partial_x \left(\left(\frac{\partial \hat{S}}{\partial x^1}.\mathrm{T_D}(t, e') + \frac{\partial \hat{S}}{\partial x^2}.\mathrm{T_D}(t, e') \right) [x/x^1, x^2], \mathrm{T_D}(T, e'') \right)
\end{aligned}$$

$$
\begin{aligned}
&\overset{\text{(I.H.)}}{=} \sum_{\substack{(e',e_1'',e_2'')\\ e'e_1''e_2''=e}} \sum_{\substack{(T',T'')\\ T'T''=T}} \binom{e}{e'}\binom{e_1''e_2''}{e_1''}\binom{T'T''}{T'}\partial_{x^1,x^2}\Bigg(\frac{\partial \hat{S}}{\partial x^1}.\mathrm{T_D}(t,e')\\
&\qquad + \frac{\partial \hat{S}}{\partial x^2}.\mathrm{T_D}(t,e'),\mathrm{T_D}(T',e_1''),\mathrm{T_D}(T'',e_2'')\Bigg)\\
&\overset{\text{(linear.)}}{=} \sum_{\substack{(e',e_1'',e_2'')\\ e'e_1''e_2''=e}} \sum_{\substack{(T',T'')\\ T'T''=T}} \binom{e}{e'}\binom{e_1''e_2''}{e_1''}\binom{T'T''}{T'}\\
&\qquad \partial_{x^1,x^2}\Bigg(\frac{\partial \hat{S}}{\partial x^1}.\mathrm{T_D}(t,e'),\mathrm{T_D}(T',e_1''),\mathrm{T_D}(T'',e_2'')\Bigg)\\
&\qquad + \partial_{x^1,x^2}\Bigg(\frac{\partial \hat{S}}{\partial x^2}.\mathrm{T_D}(t,e'),\mathrm{T_D}(T',e_1''),\mathrm{T_D}(T'',e_2'')\Bigg)\\
&\overset{(5)}{=} \sum_{\substack{(e_1'',e_2'')\\ e_1''e_2''=e}} \sum_{\substack{(T',T'')\\ T'T''=T}} \binom{e}{e_1''}\binom{T'T''}{T'}\\
&\qquad (\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(tT',e_1''),\mathrm{T_D}(T'',e_2''))\\
&\qquad + \partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e_1''),\mathrm{T_D}(tT'',e_2'')))\\
&= \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'}\Bigg(\sum_{\substack{(T',T'')\\ T'T''=tT\\ t\in\mathrm{supp}(T')}} \binom{T'T''-t}{T'-t}\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))\\
&\qquad + \sum_{\substack{(T',T'')\\ T'T''=tT\\ t\in\mathrm{supp}(T'')}} \binom{T'T''-t}{T'}\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))\Bigg)\\
&= \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'}\Bigg(\sum_{\substack{(T',T'')\\ T'T''=tT\\ t\in\mathrm{supp(T')}\\ t\notin\mathrm{supp}(T'')}} \binom{T'T''-t}{T'-t}\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))\\
&\qquad + \sum_{\substack{(T',T'')\\ T'T''=tT\\ t\in\mathrm{supp}(T'')\\ t\notin\mathrm{supp}(T')}} \binom{T'T''-t}{T'}\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))\\
&\qquad + \sum_{\substack{(T',T'')\\ T'T''=tT\\ t\in\mathrm{supp}(T')\\ t\in\mathrm{supp}(T'')}} \left(\binom{T'T''-t}{T'-t}+\binom{T'T''-t}{T'}\right)\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))\Bigg)\\
&= \sum_{\substack{(e',e'')\\ e'e''=e}} \sum_{\substack{(T',T'')\\ T'T''=tT}} \binom{e}{e'}\binom{T'T''}{T'}\partial_{x^1,x^2}(\hat{S},\mathrm{T_D}(T',e'),\mathrm{T_D}(T'',e''))
\end{aligned}
$$

$$= \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\hat{S}[x/x^1], e')[x/x^2], e'')$$

The penultimate equality is obtained by applying Pascal's formula $\binom{T-t_1}{T'-t_1} + \binom{T-t_1}{T'} = \binom{T}{T'}$ and by noticing that

· when $t_1 \in T'$ and $t_1 \notin T''$, $T(t_1) = T'(t_1)$ therefore $\binom{T-t_1}{T'-t_1} = \binom{T}{T'}$

· when $t_1 \notin T'$ and $t_1 \in T''$, $T'(t_1) = 0$ therefore $\binom{T-t_1}{T'} = \binom{T}{T'}$

- If the environment is $e[x_j \mapsto c]$ with e such that $e(x_j) = 1$,

$$\begin{aligned}
&\mathrm{T_D}(\hat{S}[x_i/x_i^1, x_i^2]_{i\in[\![1,n]\!]}, e[x_j \mapsto c]) \\
&\quad \overset{(5)}{=} \mathrm{T_D}(\partial_{x_j}(\hat{S}[x_i/x_i^1, x_i^2])_{i\in[\![1,n]\!]}, \mathrm{T_D}(c)), e) \\
&\quad = \sum_{\substack{(c',c'')\\ c'c''=c}} \binom{c}{c'} \mathrm{T_D}(\partial_{x_j^1,x_j^2}(\hat{S}, \mathrm{T_D}(c'), \mathrm{T_D}(c''))[x_i/x_i^1, x_i^2]_{i\in[\![1,n]\!], i\neq j}, e) \\
&\quad \overset{\text{(I.H.)}}{=} \sum_{\substack{(c',c'')\\ c'c''=c}} \binom{c}{c'} \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\partial_{x_j^1,x_j^2}(\hat{S}, \mathrm{T_D}(c'), \mathrm{T_D}(c'')) \\
&\qquad\qquad [x_i/x_i^1]_{i\in[\![1,n]\!], i\neq j}, e')[x_i/x_i^2]_{i\in[\![1,n]\!], i\neq j}, e'') \\
&\quad = \sum_{\substack{(c',c'')\\ c'c''=c}} \binom{c}{c'} \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\hat{S}[x_i/x_i^1]_{i\in[\![1,n]\!]}, \\
&\qquad\qquad e'[x_j \mapsto c'])[x_i/x_i^2]_{i\in[\![1,n]\!]}, e''[x_j \mapsto c'']) \\
&\quad = \sum_{\substack{(e',e'')\\ e'e''=e[x_j\mapsto c]}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\hat{S}[x_i/x_i^1]_{i\in[\![1,n]\!]}, e')[x_i/x_i^2]_{i\in[\![1,n]\!]}, e'')
\end{aligned}$$

□

Corollary 5.6 *For all resource closures $(S'S'', e)$,*

$$\mathrm{T_D}(S'S'', e) = \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(S', e')\mathrm{T_D}(S'', e'') \tag{31}$$

Proof. Let $x_1, \ldots, x_n$ be the distinct free variables of $S'S''$ and let $x_1^1, \ldots, x_n^1$ and $x_1^2, \ldots, x_n^2$ be fresh variables which do not appear in $S'S''$.

Let $\hat{S'} = S'[x_i^1/x_i]_{i\in[\![1,n]\!]}$ and $\hat{S''} = S''[x_i^2/x_i]_{i\in[\![1,n]\!]}$.

$$\begin{aligned}
\mathrm{T_D}(S'S'', e) &= \mathrm{T_D}(\hat{S'}\hat{S''}[x_i/x_i^1, x_i^2]_{i\in[\![1,n]\!]}, e) \\
&\overset{(30)}{=} \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\mathrm{T_D}(\hat{S'}\hat{S''}[x_i/x_i^1]_{i\in[\![1,n]\!]}, e')[x_i/x_i^2]_{i\in[\![1,n]\!]}, e'') \\
&\quad \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(\hat{S'}[x_i/x_i^1]_{i\in[\![1,n]\!]}, e')\mathrm{T_D}(\hat{S''}[x_i/x_i^2]_{i\in[\![1,n]\!]}, e'')
\end{aligned}$$

$$= \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \mathrm{T_D}(S',e')\mathrm{T_D}(S'',e'')$$

□

Corollary 5.7 *For all resource closures* $(\langle t\rangle T, e)$,

$$\mathrm{T_D}(\langle t\rangle T, e) = \sum_{\substack{(e',e'')\\ e'e''=e}} \binom{e}{e'} \langle \mathrm{T_D}(t,e')\rangle \mathrm{T_D}(T,e'') \tag{32}$$

Proof. Let $x_1,\dots,x_n$ be the distinct free variables of $\langle t\rangle T$ and let $x_1^1,\dots,x_n^1$ and $x_1^2,\dots,x_n^2$ be fresh variables which do not appear in $\langle t\rangle T$.

Let $\hat{t} = t[x_i^1/x_i]_{i\in[\![1,n]\!]}$ and $\hat{T} = T[x_i^2/x_i]_{i\in[\![1,n]\!]}$.

$$\begin{aligned}
\mathrm{T_D}(\langle t\rangle T, e) &= \mathrm{T_D}(\langle \hat{t}\rangle \hat{T}[x_i/x_i^1, x_i^2]_{i\in[\![1,n]\!]}, e)\\
&\overset{(30)}{=} \sum_{\substack{(e',e'')\\ e'e''=e}} \mathrm{T_D}(\mathrm{T_D}(\langle \hat{t}\rangle \hat{T}[x_i/x_i^1]_{i\in[\![1,n]\!]}, e')[x_i/x_i^2]_{i\in[\![1,n]\!]}, e'')\\
&= \sum_{\substack{(e',e'')\\ e'e''=e}} \langle \mathrm{T_D}(\hat{t}[x_i/x_i^1]_{i\in[\![1,n]\!]}, e')\rangle \mathrm{T_D}(\hat{T}[x_i/x_i^2]_{i\in[\![1,n]\!]}, e'')\\
&= \sum_{\substack{(e',e'')\\ e'e''=e}} \langle \mathrm{T_D}(t,e')\rangle \mathrm{T_D}(T,e'')
\end{aligned}$$

□

It is finally time to exhibit the main theorem of this report linking the behaviour of the **qKAM** with the Taylor expansion of algebraic terms.

Theorem 5.8 *For all algebraic states* (M,E,Π) *and for all resource states* (t,e,π),

$$K(M,E,\Pi)_{(t,e,\pi)} = (M,E,\Pi)^*_{(t,e,\pi)} NF(\mathrm{T_D}(t,e,\pi))_{c_0} \tag{33}$$

Proof. We proceed by induction on the pair $(\mathtt{size}(t,e,\pi), \mathtt{size}(M))$.

- If $\mathtt{size}(t,e,\pi)=1$, $(t,e,\pi) = (x,e_0,\emptyset)$. Either $x = c_0$ or $x \neq c_0$ and it is easy to check that in both cases the equation (33) holds.
- Otherwise, we proceed by induction on M:
  - If $M = c_0$, as $\mathtt{size}(t,e,\pi) > 1$, (t,e,π) must be different from $(c_0,e_0,\emptyset)$ therefore both $\mathrm{T_D}(c_0,e,\pi)$ and $K(c_0,E,\Pi)_{(c_0,e,\pi)}$ are equal to 0.
  - If $(M,E,\Pi) = (x,E,\Pi)$ where $E(x) \neq \uparrow$:
    If $(t,e,\pi) = (x,e,\pi)$ with e such that $\forall y, y \neq x \implies e(y) = 1$,

$$K(x,E,\Pi)_{(x,e,\pi)} \overset{(24)}{=} K(E(x),\Pi)_{(e(x),\pi)}$$

$$\overset{\text{(I.H.)}}{=} (E(x),\Pi)^*_{(e(x),\pi)}\mathrm{NF}(\mathrm{T_D}(e(x),\pi))_{c_0}$$
$$\overset{(17)}{=} E(x)^*_{e(x)}\Pi^*_\pi\mathrm{NF}(\mathrm{T_D}(e(x),\pi))_{c_0}$$
$$\overset{(5.1)}{=} x^*_x E^*_e\Pi^*_\pi\mathrm{NF}(\mathrm{T_D}(x,e,\pi))_{c_0}$$
$$\overset{(17)}{=} (x,E,\Pi)^*_{(x,e,\pi)}\mathrm{NF}(\mathrm{T_D}(x,e,\pi))_{c_0}$$

Otherwise,

$$K(x,E,\Pi)_{(t,e,\pi)} = (x,E,\Pi)^*_{(t,e,\pi)}NF(\mathrm{T_D}(t,e,\pi))_{c_0} = 0$$

· If $(M,E,\Pi) = (\lambda x.N, E, \Gamma :: \Pi')$ with $E(x) = \uparrow$:
If $(t,e,\pi) = (\lambda x.u, e, \gamma :: \pi')$ with e such that $e(x) = 1$,

$$K(\lambda x.N, E, \Gamma :: \Pi')_{(\lambda x.u,e,\gamma::\pi')}$$
$$\overset{(25)}{=} K(N, E_{x\mapsto\Gamma}, \Pi')_{(u,e_{x\mapsto\gamma},\pi')}$$
$$\overset{\text{(I.H.)}}{=} (N, E_{x\mapsto\Gamma}, \Pi')^*_{(u,e_{x\mapsto\gamma},\pi')}\mathrm{NF}(\mathrm{T_D}(u,e_{x\mapsto\gamma},\pi'))_{c_0}$$
$$\overset{(17)}{=} N^*_u(E_{x\mapsto\Gamma})^*_{e_{x\mapsto\gamma}}\Pi'^*_{\pi'}\mathrm{NF}(\mathrm{T_D}(u,e_{x\mapsto\gamma},\pi'))_{c_0}$$
$$\overset{(5.1)}{=} (\lambda x.N)^*_{\lambda x.u}E^*_e\Gamma^*_\gamma\Pi'^*_{\pi'}\mathrm{NF}(\mathrm{T_D}(u,e_{x\mapsto\gamma},\pi'))_{c_0}$$
$$\overset{(17)}{=} (\lambda x.N, E, \Gamma :: \Pi')^*_{(\lambda x.u,e,\gamma::\pi')}\mathrm{NF}(\mathrm{T_D}(\lambda x.u,e,\gamma :: \pi'))_{c_0}$$

Otherwise,

$$K(\lambda x.N, E, \Gamma :: \Pi')_{(t,e,\pi)} = (\lambda x.N, E, \Gamma :: \Pi')^*_{(t,e,\pi)}NF(\mathrm{T_D}(t,e,\pi))_{c_0} = 0$$

· If $(M,E,\Pi) = ((P)Q, E, \Pi)$:
If $(t,e,\pi) = (\langle t'\rangle T, e, \pi)$, the key is to notice that

$$E^*_e\mathrm{NF}(\mathrm{T_D}(\langle t'\rangle T,e,\pi))_{c_0} = \sum_{\substack{(e',e'')\\ e'e''=e}} E^*_{e'}E^*_{e''}\mathrm{NF}(\mathrm{T_D}(t',e',(T,e'')::\pi))_{c_0} \qquad (34)$$

Indeed, it suffices to notice that $\frac{E^*_{e'}E^*_{e''}}{E^*_e} = \binom{e}{e'}$ and to apply the equation (32).

$$K((P)Q,E,\Pi)_{(\langle t'\rangle T,e,\pi)}$$
$$\overset{(27)}{=} \sum_{\substack{(e',e'')\\ e'e''=e}} K(P,E,(Q,E)::\Pi)_{(t',e',(T,e'')::\pi)}$$
$$\overset{\text{(I.H.)}}{=} \sum_{\substack{(e',e'')\\ e'e''=e}} (P,E,(Q,E)::\Pi)^*_{(t',e',(T,e'')::\pi)}\mathrm{NF}(\mathrm{T_D}(t',e',(T,e'')::\pi))_{c_0}$$
$$\overset{(17)}{=} \sum_{\substack{(e',e'')\\ e'e''=e}} P^*_{t'}E^*_{e'}\left(\prod_{t\in\mathrm{supp}(T)}\frac{Q_t^{*T(t)}}{T(t)!}\right)E^*_{e''}\Pi^*_\pi\mathrm{NF}(\mathrm{T_D}(t',e',(T,e'')::\pi))_{c_0}$$

$$= P^*_{t'}\left(\prod_{t\in\mathrm{supp}(T)}\frac{Q_t^{*T(t)}}{T(t)!}\right)\Pi^*_\pi\sum_{\substack{(e',e'')\\ e'e''=e}}E^*_{e'}E^*_{e''}\mathrm{NF}(\mathrm{T_D}(t',e',(T,e'')::\pi))_{c_0}$$

$$\overset{(12)}{=} ((P)Q)^*_{\langle t'\rangle T}\Pi^*_\pi\sum_{\substack{(e',e'')\\ e'e''=e}}E^*_{e'}E^*_{e''}\mathrm{NF}(\mathrm{T_D}(t',e',(T,e'')::\pi))_{c_0}$$

$$\overset{(34)}{=} ((P)Q)^*_{\langle t'\rangle T}\Pi^*_{\pi'}E^*_e\mathrm{NF}(\mathrm{T_D}(\langle t'\rangle T,e,\pi))_{c_0}$$

$$\overset{(17)}{=} ((P)Q,E,\Pi)^*_{(\langle t'\rangle T,e,\pi)}\mathrm{NF}(\mathrm{T_D}(\langle t'\rangle T,e,\pi))_{c_0}$$

Otherwise,

$$K((P)Q,E,\Pi)_{(t,e,\pi)} = ((P)Q,E,\Pi)^*_{(t,e,\pi)}NF(\mathrm{T_D}(t,e,\pi))_{c_0} = 0$$

· If $(M,E,\Pi) = (\alpha N,E,\Pi)$,

$$\begin{aligned}K(\alpha N,E,\Pi)_{(t,e,\pi)} &\overset{(26)}{=} \alpha K(N,E,\Pi)_{(t,e,\pi)}\\ &\overset{\text{(I.H.)}}{=} \alpha(N,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}\\ &\overset{\text{(linear.)}}{=} (\alpha N,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}\end{aligned}$$

· If $(M,E,\Pi) = (P+Q,E,\Pi)$,

$$\begin{aligned}K(P+Q,E,\Pi)_{(t,e,\pi)} &\overset{(28)}{=} K(P,E,\Pi)_{(t,e,\pi)} + K(Q,E,\Pi)_{(t,e,\pi)}\\ &\overset{\text{(I.H.)}}{=} (P,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}\\ &\quad + (Q,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}\\ &\overset{\text{(linear.)}}{=} (P+Q,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}\end{aligned}$$

· Otherwise both $K(M,E,\Pi)_{(t,e,\pi)}$ and $(M,E,\Pi)^*_{(t,e,\pi)}\mathrm{NF}(\mathrm{T_D}(t,e,\pi))_{c_0}$ are equal to 0.

□

Using our shorthand notation $\hat{K}$ we derive the following particular case of Theorem 5.8.

Corollary 5.9 *For all algebraic terms M and for all resource terms t,*

$$\hat{K}(M)_t = M^*_t\mathrm{NF}(t)_{c_0} \tag{35}$$

5.3 Concluding remark on the computational complexity

We finish by discussing the computational complexity of the right-hand side of (35).

At first, it seemed the Corollary 5.9 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 (35) gave hope we could obtain the same result more efficiently as the right-hand side does not involve the reduction of M.

M_t^* can be computed statically by means of the coefficients m and w we have defined and it was not clear at first wether $\mathrm{NF}(t)_{c_0}$ could be computed efficiently. It turns out this problem is **NP-complete**. More precisely, we shall show that determining if a resource term reduces to a particular resource term is **NP-complete**.

Indeed, it happens that we can reduce the **CIRCUIT-SAT** [5] problem to this problem. Mairson and Terui have shown that a boolean circuit could be simulated by a linear λ-term in [7]. The linear λ-calculus can be seen as a deterministic subset of the resource λ-calculus where distinct variables only have one occurrence in λ-terms so no choice has to be made as to how to allocate the resources during the reduction – there is only one possible allocation. The resource λ-calculus then introduces the non-determinism of the allocation of resources – it involves sums corresponding to the possible allocations of resources during the reduction. Boolean circuits can be encoded in the linear λ-calculus – and therefore in the resource λ-calculus – and the properties of the resource λ-calculus allow to reduce the **CIRCUIT-SAT** problem to the problem of determining if this resource term reduces to a particular resource term. As **CIRCUIT-SAT** is known to be **NP-complete**, determining the normal form of a resource term is therefore **NP-hard** – it turns out it is even **NP-complete**.

We outline the construction given in [7]. It is possible to define booleans $\mathcal{B}$ in the linear λ-calculus. They are given the following type in the second-order intuitionistic multiplicative linear logic (**IMLL2**): $\mathcal{B} = \forall\alpha.\alpha \multimap \alpha \multimap \alpha \otimes \alpha$ with the tensor product $A \otimes B$ being defined as $A \otimes B = \forall\alpha.(A \multimap B \multimap \alpha) \multimap \alpha$. We also define the following type: $\mathbf{1} = \forall\alpha.\alpha \multimap \alpha$.

The tensor product of two terms is given by: $x \otimes y = \lambda z.zxy$.

We define the following `let` constructions which will serve later:

$\texttt{let } z \texttt{ be } x \otimes y \texttt{ in } t = z(\lambda xy.t)$

$\texttt{let } z \texttt{ be } I \texttt{ in } t = zt$

with I being the identity $\lambda x.x : \mathbf{1}$.

These constructions allows to define the weakening on booleans and the projections of the tensor product of booleans which are rules which do not hold in general in intuitionistic linear logic.

$\mathtt{w}_{\mathcal{B}}(z) = \lambda z.\texttt{let } zII \texttt{ be } x \otimes y \texttt{ in } (\texttt{let } y \texttt{ be } I \texttt{ in } x) : \mathcal{B} \multimap \mathbf{1}$

$\mathtt{fst}_{\mathcal{B}} = \lambda z.\texttt{let } z \texttt{ be } x \otimes y \texttt{ in } (\texttt{let } \mathtt{w}_{\mathcal{B}}(y) \texttt{ be } I \texttt{ in } x) : \mathcal{B} \otimes \mathcal{B} \multimap \mathcal{B}$

We are now in a position to define the booleans `true` and `false` and the usual logical connectives on booleans as well as the copy operation:

$\mathtt{true} = \lambda xy.x \otimes y : \mathcal{B}$

$\mathtt{false} = \lambda xy.y \otimes x : \mathcal{B}$

$\mathtt{not} = \lambda Pxy.Pyx : \mathcal{B} \multimap \mathcal{B}$

$\mathtt{or} = \lambda PQ.\mathtt{fst}_{\mathcal{B}}(P \texttt{ true } Q) : \mathcal{B} \multimap \mathcal{B} \multimap \mathcal{B}$

$\mathtt{copy} = \lambda P.\mathtt{fst}_{\mathcal{B}\otimes\mathcal{B}}(P(\mathtt{true} \otimes \mathtt{true})(\mathtt{false} \otimes \mathtt{false})) : \mathcal{B} \multimap \mathcal{B} \otimes \mathcal{B}$

We now have all the definitions needed to simulate a boolean circuit. In particular there is a **LOGSPACE** reduction from boolean circuits to linear λ-terms given in [6].

[5] Given a boolean circuit, determine whether there is an assignment of its inputs making the circuit outputs *true*.

For any boolean circuit C, let t_c be its associated linear λ-term seen as a resource term. The resource λ-calculus then permits to create the following term which reduces non-deterministically to `true` or `false`: $n_b = \lambda xy.\langle \lambda x.x \otimes x \rangle xy$.

Suppose t_c has k inputs, we build the following term: $\langle t_c \rangle \overbrace{\mathrm{n_b} \ldots \mathrm{n_b}}^{k \text{ times}}$. The reduction of this term will be the sum of the values returned by the circuit C on all the possible assignments of inputs. Therefore knowing if there is an assignment satisfying C amounts to knowing wether `true` belongs to the reduction of the previously defined term. Thus the problem to know if a given resource term reduces to a particular resource term is **NP-hard**. As it is easily seen to be in **NP**, it is **NP-complete**.

Acknowledgement

The author would like to thank Lionel Vaux for having suggested to use the equation (20) to prove Lemma 4.6.

References

[1] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal λ-machines. *Theoretical Computer Science*, 227(1):79–97, 1999.

[2] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, December 2003. 41 pages.

[3] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. In A. Beckmann, U. Berger, B. Löwe, and J.V. Tucker, editors, *Second Conference on Computability in Europe, CiE 2006*, LNCS 3988, pages 186–197, Swansea, United Kingdom, June 2006. Springer Berlin / Heidelberg. 12 pages.

[4] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Journal of Theoretical Computer Science*, 403(2-3):347–372, 2008.

[5] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher Order Symbolic Computation*, 20:199–207, 2007.

[6] Harry G Mairson. Functional pearl linear lambda calculus and ptime-completeness. *Journal of Functional Programming*, 14(06):623–633, 2004.

[7] Harry G Mairson and Kazushige Terui. On the computational complexity of cut-elimination in linear logic. In *Theoretical Computer Science*, pages 23–36. Springer, 2003.

[8] Lionel Vaux. The algebraic lambda calculus. *Mathematical Structures in Computer Science*, 19:1029–1059, 10 2009.