Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms

Thomas Ehrhard and Laurent Regnier*
Institut de Mathématiques de Luminy (UMR 6206)
and
Preuves, Programmes et Systèmes (UMR 7126)
`Thomas.Ehrhard@pps.jussieu.fr` and `Laurent.Regnier@iml.univ-mrs.fr`

January 6, 2006

Abstract

We show that, given an ordinary lambda-term M and a normal resource lambda-term u which appears in the normal form of the Taylor expansion of M, the unique resource term t of the Taylor expansion of M whose normal form contains u can be obtained by running the Krivine abstract machine on M.

This result, combined with a previous result of the same authors, alows to show that, in the ordinary lambda-calculus, Taylor expansion and normalization commute — by normalizing an ordinary lambda-term, we mean here computing its Böhm tree.

Introduction

After having introduced the differential lambda-calculus in [ER03], we studied in [ER05] a subsystem of the differential lambda-calculus which turns out to be very similar to resource oriented versions of the lambda-calculus previously introduced and studied by various authors [Bou93, BCL99, Kfo00]: the *resource lambda-calculus*.

Resource lambda-calculus as the target language of the complete Taylor expansion of lambda-terms. Our viewpoint on this resource lambda-calculus is that it is the sublanguage of the differential lambda-calculus where the *complete*[1]Taylor expansions of ordinary lambda-terms can be written.

Indeed, the only notion of application available in this resource calculus consists in taking a term s (of type $A \to B$ if the calculus is typed) and a finite number of terms $s_1, \dots, s_n$ (of type A) and applying s to the multiset consisting of the terms s_i, written multiplicatively $s_1 \dots s_n$. This application is written $\langle s\rangle\,(s_1 \dots s_n)$. In ordinary differential calculus, this operation would correspond to taking the nth derivative of s at 0, which is a symmetric n-linear map, and applying this derivative to the tuple $(s_1, \dots, s_n)$.

Defining a beta-reduction in this calculus (as in the original differential lambda-calculus) requires the possibility of adding terms, because the analogue of substitution is a notion of formal partial derivative whose inductive definition is based on Leibniz rule[2], and the expression $\langle s\rangle\,(s_1 \dots s_n)$ is linear in s, $s_1, \dots, s_n$; the connection between ordinary linearity and this syntactical notion of linearity is discussed in the introduction of [ER03]. The logical significance of this derivative, and the analogue in linear logic of this resource lambda-calculus are discussed in [ER04], where *differential interaction nets* are introduced. The striking fact is that

*This work has been supported by the ACI project GEOCAL.

[1]By complete, we mean that all applications in the lambda-terms are Taylor expanded.

[2]In [ER04], it is shown that Leibniz rule is more pecisely connected to the interaction between derivation and contraction.

this new structure appears in this linear logic setting as new operations associated to the exponentials, completely dual to the traditional *structural* operations (weakening, contraction), and to dereliction.

In constrast, ordinary lambda-calculus has a notion of application which is linear in the function but not in the argument, for which we used the notation $(M)\,N$ (parenthesis around the function, not around the argument). The connection between these two applications is given by the Taylor formula.

Taylor expansion and normalization. In [ER05], we explained how to Taylor expand arbitrary ordinary lambda-terms as (generally infinite) linear combinations of resource lambda-terms with rational coefficients. We showed moreover that, when normalizing the resource terms which occur in such a Taylor expansion, one gets – generally infinitely many – finite linear combinations of normal resource terms (with positive integers as coefficients) which *do not overlap*; so it makes sense to sum up all these linear combinations. Moreover, the numerical coefficients "behave well" during the reduction, in a sense which is made precise in the corresponding statement, Theorem 4 in the present paper.

Overview

We show that this sum s of normal resource terms obtained by normalizing the Taylor expansion of an ordinary lambda-term M is simply the Taylor expansion of the Böhm tree of M (the extension of Taylor expansion to Böhm trees is straightforward). Thanks to the results obtained in [ER05], this reduces to showing that a normal resource term appears in s with a nonzero coefficient iff it appears with a nonzero coefficient in the Taylor expansion of the Böhm tree of M. The "only if" part of this equivalence is fairly straightforward, whereas the "if" part requires the introduction of a version of the Krivine machine which also provides an appealing computational interpretation of the result.

Krivine machine. Usually, the Krivine machine [Kri05] is described as an abstract environment machine which performs the weak linear head reduction on lambda-terms: given a term M which is beta-equivalent to a variable x, starting from the state $(M, \emptyset, \emptyset)$ (empty environment and empty stack[3]), after a certain number of steps, the machine will produce the result $(x, E, \emptyset)$ where the resulting variable x is not bound by the environment E.

This computation can be understood as a special kind of reduction of lambda-terms which cannot be described exactly as a beta-reduction because, at each reduction step, only the *leftmost occurrence* of variable in the term is substituted. Rather than giving a formal definition[4], it is simpler to contemplate an example, so consider the term $M_0 = (\lambda x\,(x)\,(x)\,y)\,\lambda z\,z$. After one step of linear head reduction, we get $M_1 = (\lambda x\,(\lambda z\,z)\,(x)\,y)\,\lambda z\,z$. Observe that the argument and the lambda of the main redex are still there, and that the function still contains an occurrence of the variable x. Now the leftmost variable occurrence is z and the term M_1 reduces to $M_2 = (\lambda x\,(\lambda z\,(x)\,y)\,(x)\,y)\,\lambda z\,z$. The leftmost occurrence of variable is x again and we get $M_3 = (\lambda x\,(\lambda z\,(\lambda z\,z)\,y)\,(x)\,y)\,\lambda z\,z$ which reduces to $M_4 = (\lambda x\,(\lambda z\,(\lambda z\,y)\,y)\,(x)\,y)\,\lambda z\,z$. We arrive to a term M_4 whose redexes are all K-redexes[5] and reduces to the variable y.

This is exactly this kind of computation that the Krivine machine performs, with the restriction that one does not reduce under the lambda's, in some sense (whence the word "weak").

We extend the Krivine machine in two directions[6].

- First, we accept to reduce under lambda's.
- Second, when the Krivine machine arrives to a state (x, E, Π) where the environment E does not bind x and Π is a non-empty stack, it classically stops with an error. Here instead we continue the computation by running the machine on each element of Π. This corresponds, in the linear head reduction process, to reducing within the arguments of the head variable when a head normal form has been reached.

[3] The stack is there as usual for pushing the arguments of applications.

[4] By the way, the best formal definition available is certainly the Krivine machine itself.

[5] A K-redex is a redex $(\lambda x\,M)\,N$ such that x does not occur free in M. In M_4, the outermost redex is not a K-redex, but becomes a K-redex after reduction of the internal K-redexes.

[6] These extensions are fairly standard and are part of the folklore.

We call K this extended machine. When fed with a triple $(M, E, \emptyset)$ where E does not bind the free variables of M, this machine produces the Böhm tree of M (all finite approximations being obtained in a finite number of steps).

A more informative version of the machine. Then we define a version $\widehat{\mathsf{K}}$ of that machine where a "tracing mechanism" is added. The idea is to count precisely how many times the various parts of the term M have been used, starting from the state $(M, \emptyset, \emptyset)$, for reaching the state $(x, E', \emptyset)$ (when one knows that M is equivalent to the variable x). This information is summarized as a resource term which has the same shape as M (or, equivalently, appears in the Taylor expansion of M with a nonzero coefficient). For example, in the example of M_0, the corresponding resource term is $\langle\lambda x\,\langle x\rangle\,\langle x\rangle\,y\rangle\,(\lambda z\,z)^2$, which appears with coefficient $\frac{1}{2}$ in the Taylor expansion of M_0.

But there is no reason for limiting our attention to lambda-terms equivalent to a variable: when M reduces to a Böhm tree B, we just add a parameter to our Krivine machine, which is a resource term u occurring in the Taylor expansion of B. Then $\widehat{\mathsf{K}}(M, \emptyset, \emptyset, u)$ produces a resource term s which appears in the Taylor expansion of M and, in some sense, counts how much of M the machine uses for producing u. This resource term s will depend on M and on u: the larger will be u, the larger will be s.

This machine also gives us a proof for the "if" part of our main result (see the beginning of this "Overview" section), because u appears with a nonzero coefficient in the normal form of the resource term s produced by the machine.

Implementation. We implemented this modified Krivine machine in `OCaml`, you can try this program at the followin URL:

`http://iml.univ-mrs.fr/~regnier/taylor/`

1 Ordinary notions

Böhm trees. We recall a few definitions, more or less standard in lambda-calculus. An *elementary Böhm tree* (EBT) is a normal term in the lambda-calculus extended with the constant Ω subject to the following equations: $(\Omega)\,M = \Omega$ and $\lambda x\,\Omega = \Omega$. In other words:

- Ω is an elementary Böhm tree;
- if $x, x_1, \ldots, x_n$ are variables and $B_1, \ldots, B_k$ are elementary Böhm trees, then $\lambda x_1 \ldots x_n\,(x)\,B_1 \ldots B_k$ is an elementary Böhm tree.

The following clauses define an order relation on EBT's:

- $\Omega \leq B$ for all EBT B;
- $\lambda x_1 \ldots x_n\,(x)\,B_1 \ldots B_k \leq C$ if $c = \lambda x_1 \ldots x_n\,(x)\,C_1 \ldots C_k$ with $B_j \leq C_j$ for all j.

A (general) Böhm tree is now defined as an ideal of elementary Böhm trees, in other word, it is a set B of EBT's such that:

- if $B \leq C \in \mathsf{B}$ then $B \in \mathsf{B}$;
- $\mathsf{B} \neq \emptyset$ (equivalently, $\Omega \in \mathsf{B}$);
- if $B, B' \in \mathsf{B}$, there exists $C \in \mathsf{B}$ such that $B, B' \leq C$.

To any ordinary lambda-term M is associated its Böhm tree. We define first a family of functions from lambda-terms to EBT's.

- $\mathsf{BT}_0(M) = \Omega$;

- $\mathsf{BT}_{n+1}(\lambda x_1 \dots x_p\,(x)\,M_1 \dots M_k) = \lambda x_1 \dots x_p\,(x)\,\mathsf{BT}_n(M_1) \dots \mathsf{BT}_n(M_k)$;
- $\mathsf{BT}_{n+1}(\lambda x_1 \dots x_p\,((\lambda y\,P)\,Q)\,M_1 \dots M_k) = \mathsf{BT}_n(\lambda x_1 \dots x_p\,(P\,[Q/y])\,M_1 \dots M_k)$

It is straightforward to check that $\mathsf{BT}_n(M)$ is a non decreasing sequence of EBT's. Then the Böhm tree of M is the downwards closure of the set $\{\mathsf{BT}_n(M) \mid n \in \mathbb{N}\}$, which is an ideal of EBT's.

The Krivine Abstract Machine. If $f : S \to S'$ is a partial function, $a \in S$ and $b \in S'$, we denote by $f_{a \mapsto b}$ the partial function $g : S \to S'$ which is defined like f but for a, where it is defined and takes the value b.

By simultaneous induction, we define the two following concepts:

- A *closure* is a pair $\Gamma = (M, E)$ where M is a lambda-term and E is an environment such that $\mathrm{FV}(M) \subseteq \mathrm{Dom}\,E$.
- An *environment* is a finite partial function on variables, taking closures or the distinguished symbol free as value. We use $\mathrm{Dom_c}\,E$ for the subset of $\mathrm{Dom}\,E$ whose elements are not mapped to free.

We need also an auxiliary concept: a *stack* is a finite list Π of closures.

We first define a sequence of functions from states to EBT's.

- $\mathsf{K}_0(\Gamma, \Pi) = \Omega$;
- $\mathsf{K}_{n+1}(x, E, \Pi) = \mathsf{K}_n(E(x), \Pi)$ if $x \in \mathrm{Dom_c}(E)$;
- $\mathsf{K}_{n+1}(x, E, \Pi) = (x)\,\mathsf{K}_n(\Gamma_1, \emptyset) \dots \mathsf{K}_n(\Gamma_k, \emptyset)$ where $\Pi = (\Gamma_1, \dots, \Gamma_n)$, if $E(x) = \mathsf{free}$;
- $\mathsf{K}_{n+1}(\lambda x\,M, E, \emptyset) = \lambda x\,\mathsf{K}_n(M, E_{x \mapsto \mathsf{free}}, \emptyset)$ (assuming that $x \notin \mathrm{Dom}(E)$ and that x does not appear free in any of the terms mentioned in E);
- $\mathsf{K}_{n+1}(\lambda x\,M, E, \Gamma :: \Pi) = \mathsf{K}_n(M, E_{x \mapsto \Gamma}, \Pi)$ (with similar assumptions for x);
- $\mathsf{K}_{n+1}((M)\,N, E, \Pi) = \mathsf{K}_n(M, E, (N, E) :: \Pi)$.

Observe that the definition is correct in the sense that, in all "recursive calls" of the function K, the closures are well formed (the domain of their environment contains the free variables of their term).

Lemma 1 *Let $S = (\Gamma, \Pi)$ be a state. Then $(\mathsf{K}_n(S))_{n \in \mathbb{N}}$ is a non decreasing sequence of EBT's.*

This is easy to check. We define $\mathsf{K}(S)$ as the downwards closure of the set $\{\mathsf{K}_n(S) \mid n \in \mathbb{N}\}$; this set is a Böhm tree.

We define another toral function T, from closures to lambda-terms. Given a closure $\Gamma = (M, E)$, we set

$$\mathsf{T}(\Gamma) = M\,[\mathsf{T}(E(x))/x]_{x \in \mathrm{Dom_c}\,E}$$

This is a definition by induction on the height of closures, seen as finitely branching trees. We extend this mapping to states:

$$\mathsf{T}(\Gamma, (\Gamma_1, \dots, \Gamma_n)) = (\mathsf{T}(\Gamma))\,\mathsf{T}(\Gamma_1) \dots \mathsf{T}(\Gamma_n)\,.$$

The main, standard, property of the Krivine machine is as follows.

Theorem 2 *Let S be a state. Then*

$$\mathsf{K}(S) = \mathsf{BT}(\mathsf{T}(S))\,.$$

This "soundness" result shows in particular that the Krivine machine computes the Böhm tree of lambda-terms: $\mathsf{BT}(M) = \mathsf{K}(M, E, \emptyset)$, where E is any environment mapping all the free variables of M to the value free. However the way it computes it is not the standard head reduction, but the linear head reduction of [DR99], also known as De Bruijn mini-reduction [DB87]. This reduction is much more "elementary" than the standard notion, since each substitution performed by the machine is linear (only head occurrences of variables are substituted).

2 Resource notions

2.1 Notations

Let E be a set. A multiset on E is a function $m : E \to \mathbb{N}$. The support $\mathrm{supp}(m)$ of m is the set of all $a \in E$ such that $m(a) \neq 0$. The multiset m is finite if $\mathrm{supp}(m)$ is finite. The number $m(a)$ is the multiplicity of a in m. We denote by $\mathcal{M}_{\mathrm{fin}}(E)$ the set of all finite multisets on E.

2.2 The resource lambda-calculus.

We give a short account of the resource lambda-calculus, as developped in [ER05]. We recall the syntax and terminology of [ER05]. As usual we are given a countable set of variables.

Simple terms and poly-terms.

- If x is a variable, then x is a simple term.
- If x is a variable and t is a simple term, then $\lambda x\, t$ is a simple term.
- If t is a simple term and T is a simple poly-term, then $\langle t\rangle\, T$ is a simple term.
- A simple poly-term is a multiset of simple terms. We use multiplicative notations for these multisets: 1 denotes the empty poly-term, if t is a simple term, we use also t for denoting the simple poly-term whose only element is t, and if S and T are simple poly-terms, we use ST for the multiset union (sum) of S and T.

We use the greek letters $\sigma, \tau \dots$ for simple terms or poly-terms when we do not want to be specific. The size of a simple resource (poly-)term σ is a nonnegative integer $\mathsf{S}(\sigma)$, defined by induction: $\mathsf{S}(x) = 1$, $\mathsf{S}(\lambda x\, s) = 1 + \mathsf{S}(s)$, $\mathsf{S}(\langle s\rangle\, S) = 1 + \mathsf{S}(s) + \mathsf{S}(S)$ and last, for poly-terms, $\mathsf{S}(s_1 \dots s_n) = \mathsf{S}(s_1) + \cdots + \mathsf{S}(s_n)$. Observe that $\mathsf{S}(\sigma) = 0$ holds iff σ is the poly-term 1.

We use Δ for the set of all simple terms and $\Delta^!$ for the set of all simple poly-terms.

Linear combinations and reduction. Let R be a set of scalars, a commutative semi-ring with multiplicative unit 1 (so that there is a canonical semi-ring homomorphism from $\mathbb{N}$ to R, under which we consider nonnegative integers as elements of R, although this homomorphism has no reason to be injective). If A is a set, we use $R\langle A\rangle$ for the free module over R generated by A. If $\alpha \in R\langle A\rangle$, we use $\mathrm{Supp}(\alpha)$ for the set of all $a \in A$ such that $\alpha_a \neq 0$.

A redex is a simple term of the shape $r = \langle \lambda x\, s\rangle\, S$. There is a "small-step" notion of reduction for these redexes, but we do not consider it here, we focus on the big-step reduction which stipulates that r reduces to $0 \in R\langle\Delta\rangle$ if the cardinality of the multiset S is distinct from the number of free occurrences of x in s, and otherwise reduces to

$$\partial_x(s, S) = \sum_{f \in \mathfrak{S}_d} s\left[s_1, \dots, s_d / x_{f(1)}, \dots, x_{f(d)}\right] \in R\langle\Delta\rangle$$

where $S = s_1 \dots s_d$ and $x_1, \dots, x_d$ are the d free occurrences of x in s. In this expression, $\mathfrak{S}_d$ stands for the group of all permutations on the set $\{1, \dots, d\}$.

This notion of reduction extends to all simple (poly-)terms, using the fact that all constructions of the syntax are linear. For instance, if $s_1, \dots, s_n \in \Delta$ and for each i, s_i reduces to $s'_i \in R\langle\Delta\rangle$, then the simple poly-term $s_1 \dots s_n$ reduces to $\prod_{i=1}^n s'_i \in R\langle\Delta^!\rangle$.

This notion of reduction is a relation $\rightsquigarrow$ from $\Delta^{(!)}$ to $R\langle\Delta^{(!)}\rangle$; it is extended to a relation from $R\langle\Delta^{(!)}\rangle$ to itself by linearity (the linear span of $\rightsquigarrow$ in the product space $R\langle\Delta^{(!)}\rangle \times R\langle\Delta^{(!)}\rangle$). This relation is confluent, and strongly normalizing when $R = \mathbb{N}$. We use Δ_0 for the set of all normal simple terms, and NF for the normalization map $\mathbb{N}\langle\Delta^{(!)}\rangle \to \mathbb{N}\langle\Delta_0^{(!)}\rangle$, which is linear.

The following lemma is straightforward and will be useful in the sequel.

Lemma 3 *Let $s_1, \ldots, s_p$ be simple terms and $T_1, \ldots, T_p$ be simple poly-terms. Then*

$$\mathrm{Supp}(\prod_{i=1}^{p} \partial_x(s_i, T_i)) \subseteq \mathrm{Supp}(\partial_x(s_1 \ldots s_p, T_1 \ldots T_p)) .$$

Taylor expansion of ordinary lambda-terms. Let us give an intuition of the resource lambda-calculus, explaining why it is related to the idea of Taylor expansion. Usually, when f is a sufficiently regular function from a vector space E to a vector space F (finite dimensional spaces, or Banach spaces, typically), at all point $x \in E$, f has nth derivatives for all $n \in \mathbb{N}$, and these derivatives are maps $f^{(n)} : E \times E^n \to F$ with the same regularity as f and such that $f^{(n)}(x, u_1, \ldots, u_n) = f^{(n)}(x) \cdot (u_1, \ldots, u_n)$ is n-linear and symmetric in $u_1, \ldots, u_n$. When one is lucky, and at least locally around 0, the Taylor formula holds:

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} f^{(n)}(0) \cdot (u, \ldots, u) .$$

If we want to Taylor-expand lambda-terms, which after all are functions, we need to extend the language with explicit differentials, or more precisely a construction of *differential application* of a term M to n terms $N_1, \ldots, N_n$, as we did in [ER03] (a simplified version of that calculus is now available in [Vau05]). The idea is that if M represents a function f from E to F and if $N_1, \ldots, N_n$ represent n vectors $u_1, \ldots, u_n \in E$, then this new construction $\mathrm{D}^n M \cdot (N_1, \ldots, N_n)$ will represent the function from E to F which maps x to $f^{(n)}(x) \cdot (u_1, \ldots, u_n)$, and therefore this construction is linear and symmetric in the N_i's.

The Taylor expansion of a single lambda-calculus application $(M)\, N$ would then read

$$\sum_{n=0}^{\infty} \frac{1}{n!} \left(\mathrm{D}^n M \cdot (N, \ldots, N)\right) 0 .$$

If we want now to Taylor expand *all* the applications occurring in a lambda-term, we see that the usual lambda-calculus application in its generality will become useless: only application to 0 is needed. This is exactly the purpose of the construction $\langle s \rangle\, s_1 \ldots s_n$ of the resource lambda-calculus; with the notations of the differential lambda-calculus, the expression $\langle s \rangle\, s_1 \ldots s_n$ stands for $(\mathrm{D}^n s \cdot (s_1, \ldots, s_n))\, 0$.

So the resource lambda-calculus is a "target language" for completely Taylor expanding ordinary lambda-terms. The expansion of a term M will be an infinite linear combination of resource terms, with rational coefficients (actually, inverses of positive integers). Let us use M^* for the complete Taylor expansion of M. By what we said, this operation should obey

$$(M)\, N^* = \sum_{n=0}^{\infty} \frac{1}{n!} \langle M^* \rangle\, (N^*)^n$$

as well as $x^* = x$ and $(\lambda x\, M)^* = \lambda x\, M^*$. From these equations, we obtain, applying the multinomial formula, that

$$M^* = \sum_{s \in \mathcal{T}(M)} \frac{1}{\mathrm{m}(s)} s$$

where $\mathcal{T}(M) \subseteq \Delta$ is defined inductively by $\mathcal{T}(x) = \{x\}$, $\mathcal{T}(\lambda x\, M) = \{\lambda x\, s \mid s \in \mathcal{T}(M)\}$ and $\mathcal{T}((M)\, N) = \{\langle s \rangle\, S \mid s \in \mathcal{T}(M) \text{ and } S \in \mathcal{M}_{\mathrm{fin}}(\mathcal{T}(N))\}$. The positive number $\mathrm{m}(\sigma)$ associated to each (poly-)term σ is called its *multiplicity coefficient*; see the definition and properties of these numbers in [ER05]. We can recall now the main result proven in that paper.

Theorem 4 *Let M be an ordinary lambda-term.*

1. *If $s, s' \in \mathcal{T}(M)$ and s and s' are not α-equivalent, then* $\mathrm{Supp}(\mathsf{NF}(s)) \cap \mathrm{Supp}(\mathsf{NF}(s')) = \emptyset$.

2. *If $s \in \mathcal{T}(M)$ and $u \in \mathrm{Supp}(\mathsf{NF}(s))$, then the coefficient $\mathsf{NF}(s)_u$ of u in $\mathsf{NF}(s)$ (remember that this coefficient must be a positive integer) is equal to $\mathrm{m}(s)/\mathrm{m}(u)$.*

For proving the first part of this theorem, we have been led to introduce a coherence relation $\frown$ on resource terms and on resource poly-terms, in such a way that $\mathcal{T}(M)$ is a maximal clique for this coherence relation, for any ordinary term M, and we have showed that NF is in some sense "stable" with respect to this coherence relation.

The second part is based on the observation that $\mathrm{m}(s)$ has a simple combinatorial interpretation: it is the order of the group of permutations of variable occurrences of s which respect the variables associated to these occurrences and leave s unchanged; this group is non trivial in general because the elements of a simple poly-term commutes with each other ($ST = TS$). For instance, if $s = \langle x\rangle\,(\langle y\rangle\, z^2)^3$, there are $3! \times (2!)^3 = 48$ such permutations.

Given an ordinary lambda-term M, it makes sense therefore to apply NF to each of the simple terms occurring in its Taylor expansion, defining

$$\mathsf{NF}(M^*) = \sum_{s\in\mathcal{T}(M)} \frac{1}{\mathrm{m}(s)}\,\mathsf{NF}(s)\,.$$

Indeed by Theorem 4, if u is a normal simple term, there is at most one $s \in \mathcal{T}(M)$ such that $\mathsf{NF}(s)_u \neq 0$. Moreover, if such a simple term s exists, the coefficient of u in the sum above is

$$\mathsf{NF}(M^*)_u = \frac{1}{\mathrm{m}(s)}\,\mathsf{NF}(s)_u = \frac{1}{\mathrm{m}(u)}$$

by Theorem 4 again.

We want to prove that this sum is equal to $\mathsf{BT}(M)^*$, the Taylor expansion of the Böhm tree of M. To give a meaning to this notion, we need first to define $\mathcal{T}(B)$ when B is an EBT: the definition is the same as for ordinary lambda-terms, with the additional clause that $\mathcal{T}(\Omega) = \emptyset$. For instance $\mathcal{T}((x)\,\Omega) = \{\langle x\rangle\,1\}$.

Observe that $B \leq C \Rightarrow \mathcal{T}(B) \subseteq \mathcal{T}(C)$.

We generalize this notion to arbitrary Böhm trees: $\mathcal{T}(\mathsf{B}) = \bigcup_{B\in\mathsf{B}} \mathcal{T}(B)$ (this is a directed union since B is an ideal). Of course, all these resource terms are normal. The obtained set is still a clique of $(\Delta, \frown)$, but this clique has no reason to be maximal anymore (think of the clique associated with $\Omega!$). Given a Böhm tree B, it makes sense finally to define its Taylor expansion, as we did for ordinary lambda-terms:

$$\mathsf{B}^* = \sum_{b\in\mathcal{T}(\mathsf{B})} \frac{1}{\mathrm{m}(b)}\,b\,.$$

2.3 Resource closures and resource stacks.

We adapt now the concepts of closure and stack to the framework of the resource lambda-calculus, introducing multi-set based versions thereof. We stick to our multiplicative conventions for denoting multi-sets.

- A *resource environment* is a *total* function e on variables, taking resource closures or the symbol free as values. We extend pointwise the multi-set notations to resource environments, e.g. $(ee')(x) = e(x)e'(x)$ (equal to free when one of these two values is equal to free). For an environment e, we require moreover $e(x) = 1$ for almost all variables x, where 1 is the unit resource closure (see below).

  If x is a variable and c is a resource closure, we denote by $[x \mapsto c]$ the resource environment which takes the value 1 for all variables but for x, for which it takes the value c. If e is a resource environment, $e \setminus x$ denotes the resource environment which takes the same values as e but for x where it takes the value free. We use $\mathrm{Dom}_{\mathrm{c}}\, e$ for the (co-finite) set of all variables where e does not take the value free.

- A *resource closure* is a pair $c = (T, e)$ where T is a simple resource poly-term and e is a resource environment, or is the special *unit closure* 1. Intuitively, this special closure is "equal" to any closure of the shape $(1, e)$ where e maps all variables to free, to the unit closure 1 or to any closure of the shape we are now describing.

Poly-term multiplication is extended to closures in the obvious way: the unit closure 1 is neutral, and $(T, e)(T', e') = (TT', ee')$.

A resource closure (T, e) will be said to be *elementary* if T has exactly one element. All resource closures are product (in many different ways, usually) of elementary resource closures. We use the letters $c, c', \ldots$ for general resource closures and $\gamma, \gamma' \ldots$ for elementary resource closures.

Finally, a *resource stack* π is a finite sequence of resource closures.

A *resource state* is a triple (t, e, π) where t is a simple resource term, e is a resource environment and π is a resource stack. In such a resource state, the pair (t, e) will be considered as an elementary resource closure.

By mutual induction, we define $\mathcal{T}(E)$ and $\mathcal{T}(\Gamma)$, the set of all resource environments and resource closures of shape E and Γ respectively:

- $\mathcal{T}(E)$ is the set of all resource environments e such that
  - if $E(x) = \mathsf{free}$, then $e(x) = \mathsf{free}$;
  - otherwise and if $E(x)$ is defined, then $e(x) \in \mathcal{T}(E(x))$;
  - if $E(x)$ is undefined, then $e(x) = 1$.
- If $\Gamma = (M, E)$, then $\mathcal{T}(\Gamma) = (\mathcal{M}_{\mathrm{fin}}(\mathcal{T}(M)) \times \mathcal{T}(E)) \cup \{1\}$.

This extends to standard stacks and resource stacks in the obvious way, defining $\pi \in \mathcal{T}(\Pi)$. Last we set $\mathcal{T}(\Gamma, \Pi) = \mathcal{T}(\Gamma) \times \mathcal{T}(\Pi)$.

As we did for the ordinary lambda-calculus, we associate to each resource closure c a (generally not simple) resource poly-term $\mathsf{T}_{\mathrm{D}}(c) \in R\langle \Delta^! \rangle$ by the following inductive definition

$$\mathsf{T}_{\mathrm{D}}(c) = \begin{cases} 1 & \text{if } c = 1 \\ \partial_{x_1, \ldots, x_n}(T, \mathsf{T}_{\mathrm{D}}(e(x_1)), \ldots, \mathsf{T}_{\mathrm{D}}(e(x_n))) & \text{if } c = (T, e) \end{cases}$$

where $x_1, \ldots, x_n$ is any repetition-free sequence of variables which contains all the variables of $\mathrm{Dom_c}\, e$ which are free in T or satisfy $e(x) \neq 1$ (in particular, this expression is equal to 0 if there exists a variable x not free in T and such that $e(x) \neq 1$).

Due to the basic properties of partial derivatives explained in [ER05], the expression above of $\mathsf{T}_{\mathrm{D}}(c)$ does not depend on the choice of the sequence of variables $x_1, \ldots, x_n$.

Observe that when c is elementary, $\mathsf{T}_{\mathrm{D}}(c)$ can be seen as a resource term.

Last, we extend this definition to resource states (γ, π) where $\pi = (c_1, \ldots, c_k)$ is a resource stack (γ and the c_i's are therefore resource closures, and we know moreover that γ is elementary), setting

$$\mathsf{T}_{\mathrm{D}}(\gamma, \pi) = \langle \cdots \langle \mathsf{T}_{\mathrm{D}}(\gamma) \rangle\, \mathsf{T}_{\mathrm{D}}(c_1) \cdots \rangle\, \mathsf{T}_{\mathrm{D}}(c_k) \in \mathbb{N}\langle \Delta \rangle\,.$$

3 A resource driven Krivine machine

We define a new version $\widehat{\mathsf{K}}$ of the Krivine machine which, fed with an ordinary closure Γ, an ordinary stack Π and a *normal* resource term u, will return a pair $(\gamma, \pi) \in \mathcal{T}(\Gamma, \Pi)$ where γ is an elementary resource closure, or will be undefined.

As we shall see, this partial function $\widehat{\mathsf{K}}$ will be defined exactly on the triples $(\Gamma, \Pi, u))$ such that $u \in \mathcal{T}(\mathsf{BT}(\mathsf{T}(\Gamma, \Pi)))$. We use the symbol "$\uparrow$" for the result of the function when it is undefined. As before, we define by induction on n an increasing sequence of partial functions $\widehat{\mathsf{K}}_n$ and we set $\widehat{\mathsf{K}} = \bigcup_{n=0}^{\infty} \widehat{\mathsf{K}}_n$.

The base case is trivial: $\widehat{\mathsf{K}}_0(\Gamma, \Pi, t) = \uparrow$, always.

The inductive step is by case on the shape of the first element of the closure $\Gamma = (M, E)$ (remember that we assume that $\mathrm{FV}(M) \subseteq \mathrm{Dom}\, E$).

- If $M = x$ is a variable, we have two subcases.
  - Assume first that $x \in \mathrm{Dom_c}(E)$. If $\widehat{\mathsf{K}}_n(E(x), \Pi, u) = \uparrow$, then $\widehat{\mathsf{K}}_{n+1}(\Gamma, \Pi, u) = \uparrow$ and otherwise, let $(\gamma, \pi) = \widehat{\mathsf{K}}_n(E(x), \Pi, u)$, then
$$\widehat{\mathsf{K}}_{n+1}(M, E, \Pi, u) = (x, e, \pi) \quad \text{where} \quad e(y) = \begin{cases} \gamma & \text{if } y = x \\ \mathsf{free} & \text{if } E(y) = \mathsf{free} \\ 1 & \text{otherwise.} \end{cases}$$
  - Otherwise, we have $x \in \mathrm{Dom}(E)$ and $E(x) = \mathsf{free}$. The stack Π is a sequence $(\Gamma_1, \dots, \Gamma_k)$ of closures.
    * If $u = \langle \cdots \langle x \rangle V_1 \cdots \rangle V_k$ and for each $j = 1, \dots, k$ and $v \in \mathrm{supp}(V_j)$, there exists an elementary resource closure $\gamma_j(v)$ such that $\widehat{\mathsf{K}}_n(\Gamma_j, \emptyset, v) = (\gamma_j(v), \emptyset)$, then
$$\widehat{\mathsf{K}}_{n+1}(M, E, \Pi, u) = (x, e, \pi) \quad \text{where} \quad e(y) = \begin{cases} \mathsf{free} & \text{if } E(y) = \mathsf{free} \\ 1 & \text{otherwise.} \end{cases}$$
    and where $\pi = (c_1, \dots, c_k)$ with $c_j = \prod_{v \in \mathsf{supp}(V_j)} \gamma_j(v)^{V_j(v)}$ (this product has to be understood as a product of resource closures, in the sense defined above — remember that $V_j(v)$ is a positive integer, the multiplicity of v in the multiset V_j).
    * Otherwise, $\widehat{\mathsf{K}}_{n+1}(M, E, \Pi, u) = \uparrow$.
- Assume now that $M = \lambda x\, N$. Without loss of generality, we can assume that $E(x) = \uparrow$. Again, we have two subcases.
  - Assume first that $\Pi = \emptyset$ is the empty stack. If $u = \lambda x\, v$ and $\widehat{\mathsf{K}}_n(N, E_{x \mapsto \mathsf{free}}, \emptyset, v) = (t, e, \emptyset)$ with $e(x) = \mathsf{free}$, then
$$\widehat{\mathsf{K}}_{n+1}(M, E, \emptyset, u) = (\lambda x\, t, e_{x \mapsto 1}, \emptyset)$$
    and otherwise, $\widehat{\mathsf{K}}_{n+1}(M, E, \emptyset, u) = \uparrow$.
  - Assume next that $\Pi = \Gamma :: \Pi'$. If $\widehat{\mathsf{K}}_n(N, E_{x \mapsto \Gamma}, \Pi', u) = (t, e, \pi')$ with $e(x) \neq \mathsf{free}$, then
$$\widehat{\mathsf{K}}_{n+1}(M, E, \Pi, u) = (\lambda x\, t, e_{x \mapsto 1}, e(x) :: \pi')$$
    and otherwise, $\widehat{\mathsf{K}}_{n+1}(M, E, \emptyset, u) = \uparrow$.
- Last assume that $M = (P)\, Q$. If $\widehat{\mathsf{K}}_n(P, E, (Q, E) :: \Pi, u) = (t, e, (T, e') :: \pi)$, then
$$\widehat{\mathsf{K}}_{n+1}(M, E, \Pi, u) = (\langle t \rangle\, T, ee', \pi)$$
  and otherwise, $\widehat{\mathsf{K}}_{n+1}(M, E, \emptyset, u) = \uparrow$.

Lemma 5 *Let Γ be an ordinary closure, Π be an ordinary stack and u be a simple resource term. If $\widehat{\mathsf{K}}(\Gamma, \Pi, u)$ is defined, then u is normal and $\widehat{\mathsf{K}}(\Gamma, \Pi, u)$ is a resource state (γ, π) which belongs to $\mathcal{T}(\Gamma, \Pi)$.*

The proof is by simple inspection of the definition above.

Lemma 6 *Let Γ be an ordinary closure, Π be an ordinary stack and u be a normal simple resource term. For each $n \in \mathbb{N}$, we have the following equivalence:*
$$u \in \mathcal{T}(\mathsf{K}_n(\Gamma, \Pi)) \quad \textit{iff} \quad \widehat{\mathsf{K}}_n(\Gamma, \Pi, u) \textit{ is defined.}$$

Proof. By induction on n. For $n = 0$, this is trivial. Assume that the result is true for n, and consider the various following cases, according to the shape of Γ and Π. When needed, we give names to the elements of Π: $(\Gamma_1, \dots, \Gamma_k) = \Pi$.

- Assume that $\Gamma = (x, E)$ with $E(x) = \mathsf{free}$. Then $u \in \mathcal{T}(\mathsf{K}_{n+1}(\Gamma, \Pi))$ holds iff $u = \langle \cdots \langle x \rangle\, V_1 \cdots \rangle\, V_k$ with $V_j \in \mathcal{M}_{\mathrm{fin}}(\mathsf{K}_n(\Gamma_j, \emptyset))$ for each $j = 1, \dots, k$. By inductive hypothesis, this latter property holds iff for each j and each $v \in \mathrm{supp}(V_j)$, we have that $\widehat{\mathsf{K}}_n(\Gamma_j, \emptyset, v)$ is defined. But this in turn is equivalent to saying that $\widehat{\mathsf{K}}_{n+1}(\Gamma, \Pi, u)$ is defined.
- Assume that $\Gamma = (x, E)$ with $E(x) \neq \mathsf{free}$. Then $u \in \mathcal{T}(\mathsf{K}_{n+1}(\Gamma, \Pi))$ holds iff $u \in \mathcal{T}(\mathsf{K}_n(E(x), \Pi))$ which, by inductive hypothesis, holds iff $\widehat{\mathsf{K}}_n(E(x), \Pi, u)$ is defined, and this latter condition is equivalent to saying that $\widehat{\mathsf{K}}_{n+1}(\Gamma, \Pi, u)$ is defined.
- The cases $\Gamma = ((M)\, N, E)$ and $\Gamma = (\lambda x\, t, E)$ with $\Pi \neq \emptyset$ are handeled similarly.
- We are left with the case $\Gamma = (\lambda x\, M, E)$ with $\Pi = \emptyset$ which is straightforward.

$\square$

Lemma 7 *Let Γ be an odinary closure, Π be an ordinary stack and u be a normal simple resource term. Let $n \in \mathbb{N}$. If $\widehat{\mathsf{K}}_n(\Gamma, \Pi, u) = (\gamma, \pi)$, then $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\gamma, \pi)))$.*

Proof. When needed, we use $(\Gamma_1, \dots, \Gamma_k) = \Pi$. The proof is by simple induction on n.

For $n = 0$, this is trivial. So assume that $\widehat{\mathsf{K}}_{n+1}(\Gamma, \Pi, u) = (\gamma, \pi)$. We have several cases to consider, depending on Γ and Π.

- Assume first that $\Gamma = (x, E)$ with $E(x) = \mathsf{free}$. Then by our hypothesis and by the definition of $\widehat{\mathsf{K}}$, we know that $u = \langle \cdots \langle x \rangle\, V_1 \cdots \rangle\, V_k$ and that for each $j = 1, \dots, k$ and $v \in \mathrm{supp}(V_j)$, there is an elementary resource closure $\gamma_j(v)$ such that $\widehat{\mathsf{K}}_n(\Gamma_j, \emptyset, v) = (\gamma_j(v), \emptyset)$. Moreover, we know that $\gamma = (x, e)$ with
$$e(y) = \begin{cases} \mathsf{free} & \text{if } E(y) = \mathsf{free} \\ 1 & \text{otherwise,} \end{cases}$$
so that $\mathsf{T}_{\mathrm{D}}(\gamma) = x$. By inductive hypothesis, for each $j = 1, \dots, k$ and $v \in \mathrm{supp}(V_j)$, we have $v \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\gamma_j(v), \emptyset)))$. But we also know that $\pi = (c_1, \dots, c_k)$ with $c_j = \prod_{v \in \mathrm{supp}(V_j)} \gamma_j(v)^{V_j(v)}$, so that $V_j \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(c_j)))$ and finally $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\gamma, \pi)))$ as required.
- Assume next that $\Gamma = (x, E)$ with $x \in \mathrm{Dom}_{\mathrm{c}}\, E$. Then we know that $\widehat{\mathsf{K}}_n(E(x), \Pi, u)$ is defined, with value (γ', π) and that $\gamma = (x, e)$ with e given by
$$e(y) = \begin{cases} \gamma' & \text{if } y = x \\ \mathsf{free} & \text{otherwise and if } E(y) = \mathsf{free} \\ 1 & \text{otherwise.} \end{cases}$$
By inductive assumption, we have $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\gamma', \pi)))$ and we conclude since $\mathsf{T}_{\mathrm{D}}(\gamma) = \mathsf{T}_{\mathrm{D}}(\gamma')$.
- The case $\Gamma = (\lambda x\, M, E)$ and $\Pi = \emptyset$ is straightforward.
- Assume that $\Gamma = (\lambda x\, M, E)$ and $\Pi = \Gamma :: \Pi'$. Assume also that $x \notin \mathrm{Dom}(E)$. Then we know that $\widehat{\mathsf{K}}_n(M, E_{x \mapsto \Gamma}, \Pi', u)$ is defined, with value (t, e, π'), and $\widehat{\mathsf{K}}_{n+1}(\Gamma, \Pi, u) = (\lambda x\, t, e_{x \mapsto 1}, e(x) :: \pi')$ (that is $\gamma = (\lambda x\, t, e_{x \mapsto 1})$ and $\pi = e(x) :: \pi'$). Let us set $(c_1, \dots, c_k) = \pi'$. By inductive assumption, $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(t, e, \pi')))$. One sees easily that
$$\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\lambda x\, t, e_{x \mapsto 1}, e(x) :: \pi')) = \mathsf{NF}(\langle \dots \langle \langle \lambda x\, \mathsf{T}_{\mathrm{D}}(t, e_{x \mapsto \mathsf{free}}) \rangle\, \mathsf{T}_{\mathrm{D}}(e(x)) \rangle\, \mathsf{T}_{\mathrm{D}}(\gamma_1) \dots \rangle\, \mathsf{T}_{\mathrm{D}}(\gamma_k))\,.$$

Let $x_1, \dots, x_p$ be an enumeration without repetitions of all the free variables of t which do belong to $\mathrm{Dom_c}\,(e_{x\mapsto\mathsf{free}})$, so that the variable x does not occur in that list.

We have $\mathsf{T}_{\mathrm{D}}(t, e_{x\mapsto\mathsf{free}}) = \partial_{x_1,\dots,x_p}(t, \mathsf{T}_{\mathrm{D}}(e(x_1)), \dots, \mathsf{T}_{\mathrm{D}}(e(x_p)))$ by definition of T_{D} and of $e_{x\mapsto\mathsf{free}}$ and hence

$$\begin{aligned}
&\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\lambda x\, t, e_{x\mapsto 1}, e(x) :: \pi')) \\
&\quad = \mathsf{NF}(\big\langle \cdots \big\langle \partial_{x_1,\dots,x_p,x}(t, \mathsf{T}_{\mathrm{D}}(e(x_1)), \dots, \mathsf{T}_{\mathrm{D}}(e(x_p)), \mathsf{T}_{\mathrm{D}}(e(x)))\big\rangle\, \mathsf{T}_{\mathrm{D}}(\gamma_1) \cdots \big\rangle\, \mathsf{T}_{\mathrm{D}}(\gamma_k)) \\
&\quad = \mathsf{NF}(\mathsf{T}_{\mathrm{D}}(t, e, \pi'))
\end{aligned}$$

so that $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(\gamma, \pi)))$.

- Assume last that $\Gamma = ((M)\,N, E)$. We know that $\widehat{\mathsf{K}}_n(M, E, (N, E) :: \Pi, u)$ is defined, with value $(t, e, (S, e') :: \pi)$, and $\mathsf{K}_{n+1}(\Gamma, \Pi) = (\langle t\rangle\, S, ee', \pi)$, that is $\gamma = (\langle t\rangle\, S, ee')$. By inductive hypothesis, we know that $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(t, e, (S, e') :: \pi)))$. Applying Lemma 3 and the definition of T_{D} we get the inclusion
$$\mathrm{Supp}(\mathsf{T}_{\mathrm{D}}(t, e, (S, e') :: \pi)) \subseteq \mathrm{Supp}(\mathsf{T}_{\mathrm{D}}(\langle t\rangle\, S, ee', \pi)),$$
which is strict in general. We conclude, applying the NF operator to both sides.

□

4 Normal form of the Taylor expansion

Lemma 8 *Let M and N be ordinary lambda-terms and let x be a variable. Let $s \in \mathcal{T}(M)$ and $S \in \mathcal{M}_{\mathrm{fin}}(\mathcal{T}(N))$. Then* $\mathrm{Supp}(\partial_x(s, S)) \subseteq \mathcal{T}(M\,[N/x])$.

The proof is a straightforward induction on M, using Lemma 3 when M is an application.

Lemma 9 *Let M be an ordinary lambda-term and let $s \in \mathcal{T}(M)$. Then* $\mathrm{Supp}(\mathsf{NF}(s)) \subseteq \mathcal{T}(\mathsf{BT}(M))$.

Proof. One defines a family of function $\mathsf{NF}_n : \Delta \to \mathbb{N}\langle\Delta\rangle$, mimicking the definition of BT. We set $\mathsf{NF}_0(t) = 0$ and

- $\mathsf{NF}_{n+1}(\lambda x_1 \dots x_p\, \langle \cdots \langle x\rangle\, S_1 \cdots \rangle\, S_k) = \lambda x_1 \dots x_p\, \langle \cdots \langle x\rangle\, \mathsf{NF}_n(S_1) \cdots \rangle\, \mathsf{NF}_n(S_k)$
- and $\mathsf{NF}_{n+1}(\lambda x_1 \dots x_p\, \langle \cdots \langle\langle \lambda x\, s\rangle\, S\rangle\, S_1 \cdots \rangle\, S_k) = \mathsf{NF}_n(\lambda x_1 \dots x_p\, \langle \cdots \langle \partial_x(s, S)\rangle\, S_1 \cdots \rangle\, S_k)$,

where NF_n is extended multiplicatively to simple poly-terms, and linearly to linear combinations of simple (poly-)terms.

One checks easily that for any $s \in \Delta$, there exists n such that $\mathsf{NF}(s) = \mathsf{NF}_n(s)$ (take any $n \geq \mathsf{S}(s)$). One concludes by proving by induction on n that $\mathrm{Supp}(\mathsf{NF}_n(s)) \subseteq \mathcal{T}(\mathsf{BT}_n(M))$, using Lemma 8. □

Now we can prove the main theorem of the paper.

Theorem 10 *Let M be an ordinary lambda-term and let u be a normal simple resource term. Then $u \in \mathcal{T}(\mathsf{BT}(M))$ if and only if there exists $s \in \mathcal{T}(M)$ such that $u \in$ $\mathrm{Supp}(\mathsf{NF}(s))$. Moreover, when this simple term s exists, it is unique.*

Proof. Assume first that $u \in \mathcal{T}(\mathsf{BT}(M))$. Let E be an environment such that $E(x) = \mathsf{free}$ for all $x \in \mathrm{FV}(M)$. Since $\mathsf{BT}(M) = \mathsf{K}(M, E, \emptyset)$ by Theorem 2, we obtain by Lemma 6 that $\widehat{\mathsf{K}}(M, E, \emptyset, u)$ is defined ant takes a value $(s, e, \emptyset)$ with $(s, e) \in \mathcal{T}(M, E)$ by Lemma 5. Moreover, by Lemma 7 we have $u \in \mathrm{Supp}(\mathsf{NF}(\mathsf{T}_{\mathrm{D}}(s, e))) = \mathrm{Supp}(\mathsf{NF}(s))$ since e takes the value free for all free variables of s (indeed, $\mathrm{FV}(s) \subseteq \mathrm{FV}(M)$).

Assume conversely that $u \in \mathrm{Supp}(\mathsf{NF}(s))$ for some $s \in \mathcal{T}(M)$. By Lemma 9, we have $u \in \mathcal{T}(\mathsf{BT}(M))$, so again $\widehat{\mathsf{K}}(M, E, \emptyset, u)$ is defined ant takes a value $(s', e, \emptyset)$ with $(s', e) \in \mathcal{T}(M, E)$ and hence $u \in \mathrm{Supp}(\mathsf{NF}(s'))$. But both s and s' belong to $\mathcal{T}(M)$ and hence they must be equal by Theorem 4 and we conclude.

The unicity statement also results from Theorem 4. □

As a consequence of this theorem and of the "quantitative" part of Theorem 4, we get immediately the announced commutation statement.

Theorem 11 *Let M be an ordinary lambda-term. Then*

$$\mathsf{BT}(M)^* = \mathsf{NF}(M^*) = \sum_{s \in \mathcal{T}(M)} \frac{1}{\mathrm{m}(s)} \mathsf{NF}(s)\,.$$

5 Concluding remarks

By Theorem 10, there exists a partial function $\mathsf{E} : \Lambda \times \Delta_0 \to \Delta$ such that $\mathsf{E}(M, u)$ is defined if and only if $u \in \mathcal{T}(\mathsf{BT}(M))$ and then takes as value the unique simple term $s \in \mathcal{T}(M)$ such that $u \in \mathrm{Supp}(\mathsf{NF}(s))$. In the proof of that theorem, we have seen how this function E can be defined, using a modified version of Krivine machine.

When $\mathsf{BT}(M)$ is a variable $\star$, the situation is particularly simple: we have $\mathcal{T}(\mathsf{BT}(M)) = \{\star\}$ and $\mathsf{E}(M, \star)$ is the unique $s \in \mathcal{T}(M)$ which has a non-zero normal form, and the normal form of s must be $\mathrm{m}(s)\star$. In that particular case, it is interseting to observe that $\mathsf{S}(s)$ is the number of steps in the reduction of M to $\star$ by the Krivine machine, which seems to be a sensible measure of the complexity of the reduction of M.

The map $\mathsf{S} \circ \mathsf{E} : \Lambda \times \Delta_0 \to \mathbb{N}$ seems therefore to provide more generally a way of measuring the complexity of the reduction of lambda-terms. The interesting point is that this measure is associated to the algebraic property stated by Theorems 10 and 11.

References

[BCL99] Gérard Boudol, Pierre-Louis Curien, and Carolina Lavatelli. A semantics for lambda calculi with resource. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.

[Bou93] Gérard Boudol. The lambda calculus with multiplicities. Technical Report 2025, INRIA Sophia-Antipolis, 1993.

[DB87] N.G. De Bruijn. Generalizing Automath by means of a lambda-typed lambda calculus. In D.W. Kueker, E.G.K. Lopez-Escobar, and C.H. Smith, editors, *Mathematical Logic and Theoretical Computer Science*, Lecture Notes in Pure and Applied Mathematics, pages 71–92. Marcel Dekker, 1987. Reprinted in: Selected papers on Automath, Studies in Logic, volume 133, pages 313-337, North-Holland, 1994.

[DR99] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Theoretical Computer Science*, 227(1-2):273–291, 1999.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003.

[ER04] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. In *Proceedings of WoLLIC'04*, Electronic Notes in Theoretical Computer Science. Elsevier Science, 2004. To appear.

[ER05] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. Technical report, Institut de mathématiques de Luminy, 2005. Submitted for publication.

[Kfo00] Assaf J. Kfoury. A linearization of the lambda-calculus. *Journal of Logic and Computation*, 10(3):411–436, 2000.

[Kri05] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 2005. To appear.

[Vau05] Lionel Vaux. The differential lambda-mu calculus. Technical report, Institut de Mathématiques de Luminy, 2005. Submitted for publication.