On the characterization of the Taylor Expansion of λ-terms

Fanny He
Lmfi- Paris VII

09/10/2012

Contents

Acknowledgement

I would like to thank my supervisors, Pierre Boudes and Michele Pagani, for their enlightened guidance and the many insightful comments they made during the course of the internship. The interest they showed, as well as the many discussions we had, are a precious teaching I will not forget. I feel indebted to many other people that I would like to thank, especially Etienne Duchesne and Giulio Manzonetto, and Thomas Ehrhard and Alexis Saurin who first introduced me into λ-calculus, and were able to communicate their passion for this field. Completing this work would have been all the more difficult were it not for the kindness and support provided by the other members of the LIPN.

1 Introduction

The lambda-calculus was invented by Alonzo Church in the thirties, as a way to describe computable functions, alongside the recursive functions and the Turing machines. It eventually turned out that the three models are equivalent, and the Church-Turing thesis stipulates that they capture the intuitive notion of computable function. Although the model of Turing machines is a natural way to describe the computational processes, their real interest lies in the abstraction of the machine model, allowing to describe a universal notion of complexity and computability. Likewise, the interest of the λ-calculus lies in the abstraction of the essential features of functional programming, on which it allows to reason and prove properties. This core of programming features can be extended to match closely real-world programming languages, such as PCF (Programming with Computable Functions) [Plot], and recent work tend to close the gap with mainstream languages [Guha, Rey]. It is difficult to prove safety properties on these languages, making debugging hard, and therefore programs less reliable; an issue of increasing importance considering the growing number of human lives depending on computers. Giving proper semantics to general purpose programming languages will thus allow to build safer programs, and publish research papers.

In the λ-calculus, we study the interaction of functional abstraction and function application from an abstract, purely mathematical point of view: a term of λ-calculus can be a variable, an abstraction, or an application. An abstraction formalizes a function, while an application provides a term with an effective argument. A computation consists of rewriting a λ-term by the β-reduction rule, which applies the function to its arguments; one step is an atomic operation which substitutes globally all occurrences of a bound variable for a term. A λ-term can encode a partial function, and therefore the sequence of rewriting may converge to a result, or diverge. Böhm trees, infinite trees that describe the interaction of a λ-term with the external environment, were introduced in order to study the convergence of λ-terms. An important step towards the formal verification of programs is to be able to quantify the resource consumption of programs; however the β-reduction does not give information that quantifies resource consumption.

The λ-*calculus with resources* was introduced to decompose the evaluation of λ-terms, thus helping describe their resource consumption. In the resource λ-calculus, the application takes as function a resource λ-term and as argument a multiset of resource λ-terms. If the term is an abstraction, and the bound variable occurrences are in bijection with the multiset, then it can be reduced: the reduction nondeterministically assigns each element of a multiset to an occurrence of the variable, and results in a multiset of resource λ-terms consisting of all possible assignments. This reduction is *linear* since each argument is used once and only once.

The Taylor expansion of a function, a concept introduced by Brook Taylor in 1715, is a representation of a function as a possibly infinite sum of terms that are calculated from the values of the function's derivatives at a single point. Using a finite number of terms of a Taylor expansion, it is possible to approximate a function and estimate quantitatively the error in this approximation. This concept was extended in [EhrReg3] to the λ-calculus: to any λ-term we associate a set of λ-terms with resources, and this Taylor expansion describes the quantitative behaviour of the resource consumption of a program. The Taylor expansion is a quantitative refinement of Böhm trees.

There are uncountably many sets of λ-terms with resources whereas there are only countably many λ-terms; following this observation, we would like to characterize the sets of λ-terms with resources which are the Taylor expansion of some λ-term. The characterization of sets of λ-terms with resources approximating λ-terms is a step towards a semantic notion of resource consumption, and the ambitious goal to give a semantics for higher order non-deterministic calculus. This study stems on the recent advances on resource calculus [PagTas, PagMan].

First, we recall the basic notions of the λ-calculus, as well as some fundamental theorems. In section 2, we redefine Böhm trees as infinite sets of finite approximations to have a structure closer to infinite sets of resource terms, and obtain a characterization theorem for this notion of Böhm tree. In section 3, we work on resource terms and Taylor expansion of λ-terms. Following a crucial theorem that links the notions of Böhm trees and Taylor expansion, we introduce the notion of *coherence* to characterize some necessary conditions for a set of resource terms to come from the Taylor expansion of a λ-term.

2 Introduction to the λ-calculus

In this section, we recall the basic notions of the λ-calculus, as can be found in [Ehr, Sel].

2.1 Syntax

The expressions of the λ-calculus are called *λ-terms*. Let $\mathcal{V}$ be an infinite (countable) set of variables, denoted by x, y, $z \ldots$

Definition 2.1. The set of λ-terms, denoted by Λ, is given by the following grammar in *Backus-Naur Form*:

$$\Lambda : \; M, \, N \; ::= x \mid (M)N \mid \lambda x.M$$

Terms of the form x are called *variables*, $(M)N$ *applications*, and $\lambda x.M$ *λ-abstractions* (short abstractions).

An equivalent definition would be the following. Let $A = \mathcal{V} \cup \{(,), \lambda, .\}$ be an alphabet, and A^* the set of finite sequences over A. Then the set $\Lambda \subset A^*$, is the smallest subset of A^* such that $x \in \Lambda$ for all $x \in \mathcal{V}$, if M, $N \in \Lambda$ then $(M)N \in \Lambda$, and if $x \in \mathcal{V}$, and $M \in \Lambda$ then $\lambda x.M \in \Lambda$. As the Backus-Naur Form is more convenient, we will always introduce syntactic definitions in this form.

Example 2.2. Here are some examples of λ-terms:
$\lambda x.x$, $\lambda x.(x)x$, $(\lambda x.(x)x)\lambda y.(y)y$.

Remark 2.3. For the sake of readability, we apply the following conventions:

- Applications associate to the left: we write $(M)NP$ instead of $((M)N)P$. We extend this to $n \in \mathbb{N}$ applications: we can write $(M_1)M_2 \ldots M_n$ instead of $(\ldots((M_1)M_2)\ldots)M_n$.
- We can add some parenthesis to underline the structure of a λ-term. For example, we can write $(\lambda x.(x)x)(\lambda x.(x)x)$ instead of $(\lambda x.(x)x)\lambda x.(x)x$.
- We write $\lambda x_1 \ldots x_n.M$ instead of $\lambda x_1 \ldots \lambda x_n.M$.
- For a term M, we write $(M)^2N$ instead of $(M)(M)N$. More generally, we write $(M)^nN$ instead of $\underbrace{(M)\ldots(M)}_{\text{n times}}N$.

Example 2.4. We write $\lambda f\,x.(f)^2x$ instead of $\lambda f.\lambda x.(f)(f)x$.

2.2 Free and bound variables, α-equivalence

The terms $I = \lambda x.x$ and $\lambda y.y$, both represent the identity application $i : x \longmapsto x$, thus are intuitively equivalent. We therefore formalize this notion of equivalence.

Definition 2.5. An occurrence of the variable $x \in \mathcal{V}$ inside a term of the form $\lambda x.M$ is *bound*.

In this case, we say that λx is the *binder*, and that M is the *scope* of the binder.

Definition 2.6. If a variable occurrence is not bound, it is *free*. For every $M \in \Lambda$, $FV(M)$ is the *set of free variables* of M, defined by structural induction:

- $FV(x) = \{x\}$,
- $FV((M)N) = FV(M) \cup FV(N)$,
- $FV(\lambda x.M) = FV(M) \backslash \{x\}$.

Example 2.7. In the term $M = (\lambda x.((x)\lambda y.(y)x))x$, the occurrence of y is bound, the two first occurrences of x are bound, but the third occurrence of x is free. The set of free variables of M is $\{x\}$.

Definition 2.8. A term M is *closed* if $FV(M) = \emptyset$ and Λ^0 is the set of closed λ-terms.

In order to formalize the notion of equivalence, we have to define what it means to simply rename a variable in a λ-term. Let M, N be two λ-terms. If M and N are identical, we write $M \equiv N$.

Definition 2.9. Let y be a variable and M be a λ-term. We say that y belongs to M whenever it appears in M.

Definition 2.10. Let x, y be two variables and M be a λ-term. By induction on M, we define the *renaming* of x as y in M:

- $x \ll y/x \gg \equiv y$,
- $z \ll y/x \gg \equiv z$ if $x \neq z$,
- $(M)N \ll y/x \gg \equiv (M \ll y/x \gg)N \ll y/x \gg$,
- $(\lambda x.M) \ll y/x \gg \equiv \lambda y.(M \ll y/x \gg)$,
- $(\lambda z.M) \ll y/x \gg \equiv \lambda z.(M \ll y/x \gg)$, if $x \neq z$.

Example 2.11. We have $(\lambda y.x) \ll y/x \gg \equiv \lambda y.x \ll y/x \gg \equiv \lambda y.y$.

We remark that this kind of renaming replaces all occurrences of x by y, whether x is free or bound. We can now formally characterize what it means for two terms M, N to be the same, modulo the renaming of bound variables.

Definition 2.12. Let R be a relation between λ-terms. We recall that R is an *equivalence* relation if it satisfies these three rules:

Reflexivity: $\dfrac{}{M\,R\,M}$ Symmetry: $\dfrac{M\,R\,N}{N\,R\,M}$ Transitivity: $\dfrac{M\,R\,N \quad N\,R\,P}{M\,R\,P}$

We say that R is λ-*compatible* if it satisfies these two rules:

$$\frac{M\,R\,M' \quad N\,R\,N'}{(M)N\,R\,(M')N'} \qquad \frac{M\,R\,M'}{\lambda x.M\;R\;\lambda x.M'}$$

Finally, R is a *congruence* if it is an equivalence and if it satisfies λ-compatibility.

Definition 2.13. The *α-equivalence* is the smallest congruence relation $=_\alpha$ on λ-terms, satisfying the following rule:

$$\frac{y \notin M}{\lambda x.M =_\alpha \lambda y.(M \ll y/x \gg)}$$

Remark 2.14. We will always use Barendregt's variable convention, which assumes without loss of generality that bound variables have been renamed to be distinct.

Example 2.15. We have $(\lambda x.(x)x)\lambda y.(y)y =_\alpha (\lambda x.(x)x)\lambda x.(x)x$.

2.3 Substitution

After having defined a renaming operation, we turn to a less trivial operation, called substitution. Considering a variable y and two terms M, N, we would like the substitution $M\{N/x\}$ to satisfy two conditions:

- Only the free occurrences of x should be replaced, as the names of bound variables should not effect the result of a substitution. For instance, we would like the result of $((x)\lambda x.(x)y)\{y/x\}$ to be $(y)\lambda x.(x)y$, not $(y)\lambda y.(y)y$.
- We need to avoid capturing free variables. For example, if $M \equiv \lambda x.(y)x$, and $N \equiv \lambda z.(x)z$, we want the free occurrence of x in N to remain free in $M\{N/y\}$. In order not to capture a free variable, we rename the bound variable before the substitution:

$$M\{N/y\} =_\alpha (\lambda x'.(y)x')\{N/y\} \equiv \lambda x'.(N)x' \equiv \lambda x'.(\lambda z.(x)z)x'$$

Remark 2.16. In the substitution operation, we often have to rename a bound variable x by the name of a variable which has not been used yet. That is why we need the set $\mathcal{V}$ of variables to be infinite. We say that we use a *fresh* variable to rename x.

Now we formally define the notion of substitution:

Definition 2.17. The substitution of N for free occurrences of x in M, denoted by $M\{N/x\}$, is defined as follows:

- $x\{N/x\} \equiv N$,
- $y\{N/x\} \equiv y$ if $x \neq y$,
- $((M)P)\{N/x\} \equiv (M\{N/x\})P\{N/x\}$,

- $(\lambda x.M)\{N/x\} \equiv \lambda x.M$,
- $(\lambda y.M)\{N/x\} \equiv \lambda y.(M\{N/x\})$ if $x \neq y$ and $y \notin FV(N)$,
- $(\lambda y.M)\{N/x\} \equiv \lambda y'.(M \ll y'/y \gg \{N/x\})$ if $x \neq y$, $y \in FV(N)$, and y' fresh.

Example 2.18. $(\lambda y.x)\{y/x\} \equiv \lambda y'.x \ll y'/y \gg \{y/x\} \equiv \lambda y'.y$.

The substitution is well-defined modulo α-equivalence. From now on, we identify λ-terms modulo α-equivalence, and use indifferently $=$ or $=_\alpha$.

2.4 β-reduction, β-normal form

We evaluate λ-terms by applying functions to arguments; this process is called β-reduction. A *redex* (or β-redex) is a term of the form $(\lambda x.M)N$. The *β-reduction*, is a process that reduces a redex $(\lambda x.M)N$ to $M\{N/x\}$. Each step of β-reduction is denoted by $\longrightarrow_\beta$. A term without any redexes is in *β-normal form* (short normal form).

Definition 2.19. Formally, a single-step β-reduction, denoted by $\longrightarrow_\beta$, is the smallest relation on λ-terms which satisfies the rules:

$$\frac{}{(\lambda x.M)N \longrightarrow_\beta M\{N/x\}} \qquad \frac{M \longrightarrow_\beta M'}{(M)N \longrightarrow_\beta (M')N}$$

$$\frac{N \longrightarrow_\beta N'}{(M)N \longrightarrow_\beta (M)N'} \qquad \frac{M \longrightarrow_\beta M'}{\lambda x.M \longrightarrow_\beta \lambda x.M'}$$

Example 2.20. We have $M = ((\lambda x.(x)x)\lambda y.y)y \longrightarrow_\beta ((\lambda y.y)\lambda y.y)y \longrightarrow_\beta (\lambda y.y)y \longrightarrow_\beta y$. The last term, y, has no redexes and thus is in normal form.

Example 2.21. The term $\Omega = (\lambda x.(x)x)\lambda x.(x)x$ reduces to itself, thus Ω does not reduce to a normal form.

Example 2.22. The term $Y = \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x$ is such that $Y \longrightarrow_\beta \lambda f.(f)(\lambda x.(f)(x)x)\lambda x.(f)(x)x \longrightarrow_\beta \lambda f.(f)(f)(\lambda x.(f)(x)x)\lambda x.(f)(x)x \ldots$, thus Y does not reduce to a normal form.

Example 2.23. Let $Y' = (\lambda x.z)Y$. Then Y' can have infinite reductions like, for any $n \in \mathbb{N}$, $(\lambda x.z)(\lambda f.(f)(f)^n(\lambda x.(f)(x)x)\lambda x.(f)(x)x)$, if at each step we reduce the redex $(\lambda x.(f)(x)x)\lambda x.(f)(x)x$. But Y' can also have a reduction that erases all redexes, by considering the redex $(\lambda x.z)Y$, giving the normal form z. We see in this example that different reductions can start from a single term, depending on which redex to reduce. Furthermore, the confluence theorem (in [Ehr] Theorem 1.2.10, 2.34, 2.32) will prove that eventually that if there exists a normal form, then it is unique, and furthermore all sequences of reductions can always be completed to reach the normal form.

Definition 2.24. The relation $\twoheadrightarrow_\beta$ is the *reflexive transitive closure* of $\longrightarrow_\beta$, therefore $M \twoheadrightarrow_\beta M'$ if M reduces to M' in 0 or more steps. The reflexive transitive *symmetric* closure of $\longrightarrow_\beta$, called *β-equivalence*, is written $=_\beta$.

Example 2.25. We have $(\lambda yx.(y)x)\lambda z.z \longrightarrow_\beta \lambda x.(\lambda z.z)x \longrightarrow_\beta \lambda x.x$, we write $(\lambda yx.(y)x)\lambda z.z \twoheadrightarrow_\beta \lambda x.x$, and $\lambda x.x =_\beta (\lambda yx.(y)x)\lambda z.z$.

Remark 2.26. The relation $=_\beta$ is a congruence.

We now introduce some usual terminology of rewriting theory.

Definition 2.27. Let R be a binary relation on a set T of terms. We call R^* the reflexive transitive closure of R.

- $t \in T$ is *R-normal* if there is no $t' \in T$ such that $t\,R\,t'$.
- $t \in T$ is *R-weakly normalizable* if there exists a sequence $t_1 = t, t_2, \dots, t_n$ such that $\forall i \in \{1, \dots, n\}$, $t_i\,R\,t_{i+1}$ and t_n is R-normal. We say that t_n is a *normal form* of t.
- $t \in T$ is *strongly normalizable* if there is no sequence $(t_i)_{i \in \mathbb{N}\backslash\{0\}}$ such that $t_1 = t$ and $\forall i \in \mathbb{N}\backslash\{0\}$, $t_i\,R\,t_{i+1}$.
- (T, R) enjoys *weak* (resp. *strong*) *normalization* if any element of T is R-weakly (resp.strongly) normalizable.
- (T, R) enjoys *strong confluence* if, for every $t, t_1, t_2 \in T$, if $t\,R\,t_i$ for $i = 1, 2$, there exists $t' \in T$ such that $t_i\,R\,t'$ for $i = 1, 2$.
- (T, R) enjoys *local confluence* if, for every $t, t_1, t_2 \in T$, if $t\,R\,t_i$ for $i = 1, 2$, there exists $t' \in T$ such that $t_i\,R^*\,t'$ for $i = 1, 2$.
- (T, R) enjoys *confluence* if (T, R^*) enjoys strong confluence.

Remark 2.28. Strong normalization implies weak normalization, as well as strong confluence implies local confluence, but in each case, the converse is false (see examples 2.29, 2.30, 2.31).

Example 2.29. In (Λ, β), the term $(\lambda x.z)Y$ is weakly normalizable to z, but not strongly normalizable, as we have the infinite sequence:

$$(t_i)_{n \in \mathbb{N}\backslash\{0\}} = ((\lambda x.z)(\lambda f.(f)^n(\lambda x.(f)(x)x)\lambda x.(f)(x)x))_{n \in \mathbb{N}\backslash\{0\}}$$

such that $t_1 = t$ and $\forall i \in \mathbb{N}\backslash\{0\}$, $t_i\,R\,t_{i+1}$, as seen in 2.23.

Example 2.30. In (Λ, β), for any $n \in \mathbb{N}$, the term $(\lambda x.x)^n\lambda x.x$ is strongly normalizable, therefore weakly normalizable.

Example 2.31. In (Λ, β), Y is not weakly normalizable, therefore not strongly normalizable.

As some λ-terms are not weakly normalizable, (Λ, β) does not enjoy weak normalization. Therefore, (Λ, β) does not enjoy strong normalization.

Lemma 2.32. If (T, R) enjoys confluence, and if $t \in T$ has a normal form, then t has a unique normal form.

Proof. Let T be a set of terms, and $R \subseteq T^2$, such that (T, R) enjoys confluence. Let $t \in T$. Suppose that there exists $t', t'' \in T$, normal forms of t. Then $t \twoheadrightarrow_R t'$ and $t \twoheadrightarrow_R t''$. By confluence, there exists $t''' \in T$ such that $t' \twoheadrightarrow_R t'''$ and $t'' \twoheadrightarrow_R t'''$. However, as t', t'' are R-normal, then $t' = t'' = t'''$, and we have the unicity of the normal form. □

Corollary 2.33. If the β-normal form of a λ-term exists, it is unique.

We recall the Church-Rosser theorem from [Ehr]:

Theorem 2.34. [Church-Rosser] The rewriting system (Λ, β) enjoys confluence.

The crucial Newman's lemma follows:

Lemma 2.35. [Newman] If the relation R is strongly normalizing and locally confluent, then R is confluent.

Example 2.36. The rewriting system (Λ, β) is locally confluent, whereas the rewriting system $(\{a, b, c\}, \{(a, b), (a, c)\})$ is not locally confluent.

2.5 Solvability, head normal form

In section 3 on Böhm trees, we will see that Böhm trees are designed to understand to notion of solvability. The notions of solvability and head normal form are important, as they define in the λ-calculus the notion of convergence.

Definition 2.37. Let $M \in \Lambda^0$. M is *solvable* if there exists $n \in \mathbb{N}$, and $N_1, \ldots, N_n \in \Lambda$, such that $(M)\, N_1 \ldots N_n =_\beta \lambda x.x$.

The *closure* of a λ-term M is a λ-term N such that $N = \lambda x_1 \ldots x_n.M$ and N is closed. We now consider an arbitrary λ-term:

Definition 2.38. Let $M \in \Lambda$. M is *solvable* if a closure $\lambda x_1 \ldots x_n.M$ of M is solvable.

This is independent on the order of $x_1, \ldots, x_n$.

A term $M \in \Lambda$ is *unsolvable* if M is not solvable.

Example 2.39. The term $K = \lambda xy.x$ is solvable. We have $(K)II =_\beta I$, where $I = \lambda x.x$.

Example 2.40. The term $\Omega = (\lambda x.(x)x)\lambda x.(x)x$ is unsolvable. For any $N_1, \ldots,$ N_n, if $(\Omega)N_1 \ldots N_n \twoheadrightarrow_\beta M$, then $M = (\Omega)N'_1 \ldots N'_n$, with $N_i \twoheadrightarrow_\beta N'_i$, for $i \leq n$.

Remark 2.41. One can notice that a closed term M is solvable if and only if, for any $P \in \Lambda$, there exists $N_1, \ldots, N_n \in \Lambda$ such that $(M)N_1 \ldots N_n =_\beta P$.

We deduce a property on unsolvability (proven in [Bar], section 8.3): if a λ-term M is unsolvable, then so are $(M)N$, $M\{N/x\}$, and $\lambda x.M$, for any variable x, and $N \in \Lambda$.

For any $M \in \Lambda$, M is of one of the following two forms ([Bar] 8.3.8.):

- $M = \lambda x_1 \ldots x_n.(x)M_1 \ldots M_m$ with $n, m \geq 0$,
- $M = \lambda x_1 \ldots x_n.(\lambda x.M_0)M_1 \ldots M_m$ with $n \geq 0, m \geq 1$.

Definition 2.42. [Head Normal Form] Let $M \in \Lambda$. Then M is in *head normal form* (or is a head normal form or is hnf) if $M = \lambda x_1 \ldots x_n.(x)M_1 \ldots M_m$, where x is a variable, $n, m \geq 0$. We call x the *head variable*. If $M = \lambda x_1 \ldots x_n.((\lambda x.P)Q)M_1 \ldots M_n$, we call $((\lambda x.P)Q)$ the *head redex* of M.

We write HNF the *set of head normal forms*.

Definition 2.43. The term $M \in \Lambda$ has a head normal form if there exists a $M' \in$ HNF such that $M =_\beta M'$.

We now define the *head reduction*, denoted by β_h:

Definition 2.44. Let $M, M' \in \Lambda$. Then $M \longrightarrow_{\beta_h} M'$ if M has a head redex and M' is obtained by reducing the head redex of M.

Remark 2.45. Normal forms for β_h-reduction are head normal forms. The head normal forms are the normal forms of head reduction.

Remark 2.46. The head reduction is deterministic, as at each step, there is at most one redex to reduce.

Remark 2.47. If $M \twoheadrightarrow_\beta M'$, and if M' is a hnf, then the head reduction on M is finite, and stops on a hnf N', which can be different from M', because of the arguments on the head variable, but $M' =_\beta N'$. We call N' the *principal head normal form*.

Example 2.48. The term $\lambda x.(x)Y$ is in hnf but is not β-normalizable. $Y = \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x$ is head normalizable but not β-normalizable. Ω is not head normalizable.

The following lemma explains how head reduction is so important. Head reduction is an effective procedure. More precisely, head reduction is an effective convergent procedure:

Lemma 2.49. Let $M \in \Lambda$. Then M has a hnf if and only if the head reduction path of M terminates.

We finally recall the following theorem (in [Bar], th. 8.3.14):

Theorem 2.50. Let $M \in \Lambda$. Then M is solvable if and only if M has a hnf.

This theorem is important, as it states the equivalence of two notions: solvability and head-normalization. Solvability gives the right notion of convergence in λ-calculus. However, solvability is defined by using an existential quantifier (see 2.37), and therefore gives no procedure to find, for a λ-term $M \in \Lambda^0$, the list of λ-terms $N_1, \dots, N_n$ such that $(M)N_1 \dots N_n =_\beta \lambda x.x$. The notion of head reduction allows to overcome the difficulty: head-reduction is an effective procedure, hence the theorem says that a term is solvable if and only if such a procedure terminates.

3 Böhm Trees resulting from λ-terms

3.1 Recall on Böhm trees

Now that we have recalled the necessary notions of λ-calculus for the next sections, we introduce a few definitions related to *Böhm trees* and set up some notations. We first introduce the notion of *elementary Böhm trees* (EBT's).

Definition 3.1. EBT : $b, c ::= \Omega \mid \lambda x_0 \dots x_{n-1}.(y)b_0 \dots b_{k-1}$;

Following this definition, we make two observations:

Remark 3.2. By the second rule, applied to $n = k = 0$, variables are EBT's.

Remark 3.3. It is usual to see Ω as a constant, not in Λ, representing divergence for β-reduction. However, when we want to have EBT $\subseteq \Lambda$, without loss of generality, we can consider $\Omega = (\lambda x.(x)x)\ \lambda x.(x)x$.

Remark 3.4. Actually, we will silently suppose that EBT and sets of EBT's are defined modulo α-equivalence. In particular, two sets $\mathcal{B}, \mathcal{B}'$ of EBT's are α-equivalent whenever there is a sequence σ of renaming of bound variables in $\mathcal{B}$, such that $\mathcal{B}' = \{b\sigma \mid b \in \mathcal{B}\}$.

We now define an order relation on EBT's.

Definition 3.5. The relation $\sqsubseteq$ is defined by induction on EBT's:

- $\Omega \sqsubseteq b$ for all EBT b;
- $\lambda x_0 \dots x_{n-1}.(y)b_0 \dots b_{k-1} \sqsubseteq c$ if $c = \lambda x_0 \dots x_{n-1}.(y)c_0 \dots c_{k-1}$ with $b_j \sqsubseteq c_j$ for all j.

Example 3.6. We have $\Omega \sqsubseteq \lambda x.(y)\Omega$ and $\Omega \sqsubseteq \lambda x.(y)x$.

Example 3.7. The order $\sqsubseteq$ is partial. Let y be a variable, $b_0 \in$ EBT: we have $(y)\, b_0\, \Omega \not\sqsubseteq (y)\, \Omega\, b_0$ and $(y)\, \Omega\, b_0 \not\sqsubseteq (y)\, b_0\, \Omega$, however $(y)\, \Omega\, \Omega \sqsubseteq (y)\, b_0\, \Omega$, $(y)\, \Omega\, \Omega \sqsubseteq (y)\, \Omega\, b_0$.

Our aim is now to associate with any λ-term M, a non-empty set of EBT's, called the Böhm tree of M, denoted by $BT(M)$. First, we define a family of functions from λ-terms to EBT's.

Definition 3.8. For all $n \in \mathbb{N}$, we have $BT_n(M)$:

- $BT_0(M) = \Omega$;
- $BT_{n+1}(\lambda x_0 \dots x_{p-1}.(y)M_0 \dots M_{l-1}) = \lambda x_0 \dots x_{p-1}.(y)BT_n(M_0) \dots BT_n(M_{l-1})$;
- $BT_{n+1}(\lambda x_0 \dots x_{p-1}.((\lambda y.Q)R)M_0 \dots M_{l-1}) = BT_n(\lambda x_0 \dots x_{p-1}.(Q\{R/y\})M_0 \dots M_{l-1})$.

Remark 3.9. We have that for all $n \in \mathbb{N}$, $BT_n(M) \sqsubseteq BT_{n+1}(M)$.

We can now describe $BT(M)$ as a map from λ-terms to sets (actually $\sqsubseteq$-ideals) of EBT's:

Definition 3.10. Let $M \in \Lambda$. Then $BT(M)$ is the $\sqsubseteq$-downwards closure of $\{BT_n(M), n \in \mathbb{N}\}$.

Example 3.11. There are several classic examples of Böhm trees. The trivial one is $BT((\lambda x.(x)x)\lambda x.(x)x) = \{\Omega\} = BT(\lambda y.(\lambda x.(x)x)\lambda x.(x)x) = BT(((\lambda x.(x)x)\lambda x.(x)x)M)$. Let Y be a fixed-point combinator, $Y = \lambda f.(\lambda x.(f)(x)x)\ \lambda x.(f)(x)x$. Then $BT(Y) = \{\Omega\} \bigcup \bigcup_{n\in\mathbb{N}} \{\lambda f.(f)^{n+1}\Omega\}$.

Example 3.12. A Böhm tree has a lattice structure: any two elements have a *supremum* and an *infimum* (see lemma 3.18). Here is an example for the Böhm tree of a λ-term:

$$\begin{aligned} BT(\lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ x_0) = \{ & \Omega,\ \lambda x_0.\ ((x_0)\ \Omega)\Omega, \\ & \lambda x_0.((x_0)\ \lambda y.(x_0)\Omega)\ \Omega, \\ & \lambda x_0.\ ((x_0)\ \Omega)\ x_0,\ \lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ \Omega, \\ & \lambda x_0.((x_0)\ \lambda y.(x_0)\Omega)\ x_0,\ \lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ x_0\} \end{aligned}$$

$$\begin{tikzcd}
 & & \Omega \arrow[d, dash] & \\
 & & \lambda x_0.\ ((x_0)\ \Omega)\Omega \arrow[dl, dash] \arrow[dr, dash] & \\
 & \lambda x_0.((x_0)\ \lambda y.(x_0)\Omega)\ \Omega \arrow[dl, dash] \arrow[dr, dash] & & \lambda x_0.\ ((x_0)\ \Omega)\ x_0 \arrow[dl, dash] \\
\lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ \Omega \arrow[drr, dash] & & \lambda x_0.((x_0)\ \lambda y.(x_0)\Omega)\ x_0 \arrow[d, dash] & \\
 & & \lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ x_0 &
\end{tikzcd}$$

Böhm trees have been constructed so that they remain invariant by β-reduction. For instance, $BT((\lambda x.x)y) = BT(y) = \{\Omega, y\}$.

Claim 3.13. If $M =_\beta N$, then $BT(M) = BT(N)$.

Proof. By structural induction on M, we prove that if $M \longrightarrow_\beta N$, then, for all $m \in \mathbb{N}$, there exists $n \geq m$ such that $BT_m(M) =_\beta BT_n(N)$ and vice versa: for all $n \in \mathbb{N}$, there exists $m \leq n$ such that $BT_m(M) = BT_n(N)$. $\square$

Remark 3.14. The converse of claim 3.13 is false. There are λ-terms M, N such that $BT(M) = BT(N)$ but $M \neq_\beta N$. For example, $BT((\lambda x.(x)x)\lambda x.(x)x) = BT(\lambda y.(\lambda x.(x)x)\lambda x.(x)x)$.

3.2 Towards the characterization theorem

Every λ-term has a Böhm tree, however, not every set of EBT's comes from a λ-term. For instance, $\mathcal{B} = \{\lambda x.x\}$ is not the Böhm tree image of any λ-term, as well as the ideal $\mathcal{B} = \{\Omega, (x_1)\,\Omega, (x_1)\,(x_2)\,\Omega, (x_1)\,(x_2)\,(x_3)\,\Omega, \dots\}$.

In this section, we give three conditions that characterize exactly the Böhm tree image subset of EBT.

Let $\mathcal{B}$ be a subset of EBT's.

The first condition for $\mathcal{B}$ to come from a λ-term is to be a $\sqsubseteq$-ideal (see definition 3.15). The set $\{\lambda x.x\}$ is not coming from a λ-term, since it is not $\sqsubseteq$-closed.

The second condition for $\mathcal{B}$ is to have a finite number of free variables (see definition 3.21). The ideal $\mathcal{B} = \{\Omega, (x_1)\,\Omega, (x_1)\,(x_2)\,\Omega, (x_1)\,(x_2)\,(x_3)\,\Omega, \dots\}$ is for instance not coming from a λ-term, and indeed $FV(\mathcal{B})$ is infinite.

The third condition for $\mathcal{B}$ is to be recursively enumerable (see definition 3.34). To give an intuition, we can construct an ideal $\mathcal{B}$ based on the Halting problem, and this ideal $\mathcal{B}$ is not recursively enumerable. For every $n \in \mathbb{N}$, we want $b_n \in \mathcal{B}$, where

$$b_n = \lambda xy.(u_0)\dots(u_n)\Omega = \begin{cases} \lambda xy.(u_0)(u_1)\dots(u_{n-1})(x)\Omega & \text{if program } n \text{ halts on input } n \\ \lambda xy.(u_0)(u_1)\dots(u_{n-1})(y)\Omega & \text{otherwise} \end{cases}$$

There is no $M \in \Lambda$ such that $BT(M) = \mathcal{B}$, because $BT(M)$ is recursively enumerable, since it is defined by an effective procedure (see lemma 3.36), while $\mathcal{B}$ is not, deciding the Halting problem.

3.2.1 Ideal

An ideal is a set $\mathcal{B}$ of EBT's, such that $\mathcal{B}$ verifies a number of properties for the order $\sqsubseteq$:

Definition 3.15. [Ideal] An *ideal* $\mathcal{B} \subseteq EBT$ is a set of elementary Böhm trees such that the following conditions hold:

- $\Omega \in \mathcal{B}$;
- if $b \sqsubseteq c \in \mathcal{B}$ then $b \in \mathcal{B}$;
- if $b, b' \in \mathcal{B}$, there exists $c \in \mathcal{B}$ such that $b, b' \sqsubseteq c$.

Example 3.16. The singleton $\{\Omega\}$ is an ideal. It is the minimum ideal with respect to the set theoretical inclusion on the ideals of EBT's.

Example 3.17. Let $(x_n)_{x\in\mathbb{N}}$ be an enumeration of distinct variables. Then $\{\Omega, (x_1)\,\Omega, (x_1)\,(x_2)\,\Omega, (x_1)\,(x_2)\,(x_3)\,\Omega, \dots\}$ is an ideal.

Every λ-term is associated to a Böhm tree. In particular, this Böhm tree is an ideal of EBT's.

Lemma 3.18. For all $M \in \Lambda$, $BT(M)$ is an ideal of EBT's.

Proof. Let $M \in \Lambda$. We now check that $BT(M)$ verifies the properties of a $\sqsubseteq$-ideal:

- $\Omega = BT_0(M) \in BT(M)$.
- Let $c \in BT(M)$, let $b \in EBT$, $b \sqsubseteq c$. There exists $n \in \mathbb{N}$ such that $c \sqsubseteq BT_n(M)$, and consequently $b \sqsubseteq BT_n(M)$. As $BT(M)$ is the $\sqsubseteq$-downwards closure of $\{BT_n(M), n \in \mathbb{N}\}$, we have $b \in BT(M)$.
- Let b, $b' \in BT(M)$. There exists n_1, $n_2 \in \mathbb{N}$ such that $b \sqsubseteq BT_{n_1}(M)$ and $b' \sqsubseteq BT_{n_2}(M)$. Then b, $b' \sqsubseteq BT_{\max\{n_1,n_2\}}(M) \in BT(M)$.

$\square$

3.2.2 A finite set of free variables

We now describe the set of free variables of an EBT:

Definition 3.19. Let $b \in EBT$. The set $FV(b)$ of free variables of b is:

- $\emptyset$ if $b = \Omega$;
- $\left(\{y\} \cup \bigcup_{i=0}^{k-1} FV(b_i)\right) \setminus \{x_0, \dots, x_{n-1}\}$ if $b = \lambda x_0 \dots x_{n-1}.(y)b_0 \dots b_{k-1}$.

Example 3.20. We have $FV(((x_0)\,\lambda y.(x_0)y)\,x_0) = \{x_0\}$.

We extend this definition to a set of EBT's:

Definition 3.21. Let $\mathcal{B} \subseteq EBT$. Then $FV(\mathcal{B}) = \bigcup_{a\in\mathcal{B}} FV(a)$.

Example 3.22. Let $M = ((x_0)\ \lambda y.(x_0)y)\ x_0$. We have:

$$FV(BT(((x_0)\ \lambda y.(x_0)y)\ x_0)) = \bigcup_{a \in BT(M)} FV(a) = \{x_0\}$$

Remark 3.23. It is easy to observe that if $\mathcal{B}_1 \subseteq \mathcal{B}_2$, then $FV(\mathcal{B}_1) \subseteq FV(\mathcal{B}_2)$. As an immediate consequence, $FV(BT(M)) = \bigcup\limits_{n \in \mathbb{N}} FV(BT_n(M))$.

Lemma 3.24. For all $M \in \Lambda$, we have $FV(BT(M)) \subseteq FV(M)$. It follows that $FV(BT(M))$ is finite.

Proof. We only have to show that for $M \in \Lambda$ and $n \in \mathbb{N}$, we have $FV(BT_n(M)) \subseteq FV(M)$. Let $M \in \Lambda$.

If $n = 0$, we have $FV(BT_0(M)) = FV(\Omega) = \emptyset \subseteq FV(M)$.

If $n = m + 1$, there are two cases:

- $M = \lambda x_0 \dots x_{p-1}(y)M_0 \dots M_{l-1}$, $FV(M) = (\{y\} \cup \bigcup\limits_{i=0}^{l-1} FV(M_i)) \backslash \{x_0, \dots, x_{p-1}\}$,

  and $FV(BT_{m+1}(M)) = (\{y\} \cup \bigcup\limits_{i=0}^{l-1} FV(BT_m(M_i))) \backslash \{x_0, \dots, x_{p-1}\}$ Then by induction hypothesis,

  $$FV(BT_{m+1}(M)) \subseteq (\{y\} \cup \bigcup_{i=0}^{l-1} FV(M_i)) \backslash \{x_0, \dots, x_{p-1}\} = FV(M)$$

- $M = \lambda x_0 \dots x_{p-1}.((\lambda y.Q)R)M_0 \dots M_{l-1}$
  We have $M \longrightarrow_\beta \lambda x_0 \dots x_{p-1}.(Q\{R/y\})M_0 \dots M_{l-1}$. Remember that Böhm trees are invariant by β-reduction (claim 3.13) and by definition, $FV(BT_{m+1}(M)) = FV(BT_m(\lambda x_0 \dots x_{p-1}.(Q\{R/y\})M_0 \dots M_{l-1}))$. Then by induction hypothesis,

  $$FV(BT_{m+1}(M)) \subseteq FV(\lambda x_0 \dots x_{p-1}.(Q\{R/y\})M_0 \dots M_{l-1}) \subseteq FV(M)$$

□

Remark 3.25. The converse is false. We have $FV(BT(((\lambda x.(x)x)\lambda x.(x)x)x)) = \emptyset \neq FV(((\lambda x.(x)x)\lambda x.(x)x)\ x) = \{x\}$.

3.2.3 Recursive enumerability

Now we want to characterize what is an "effective Böhm tree". In order to do that, we use the notion of a recursively enumerable subset of $\mathbb{N}$. So we encode

Λ into $\mathbb{N}$ by a Gödel number mapping # that is any effective bijection between $\Lambda/_{=_\alpha}$ and $\mathbb{N}$. One way of defining # is by using the De Bruijn notation, which gives a system of canonical representatives for $=_\alpha$. We will omit such details, so we fix once and for all a recursive Gödel numbering $\# : \Lambda \longrightarrow \mathbb{N}$.

Remark 3.26. The Gödel number are applied to EBT's with the convention that $\#\Omega = \#(\lambda x.(x)x)\ \lambda x.(x)x$.

Remark 3.27. The inverse function of a recursive injection is a partial recursive function. In particular, $\#^{-1}$ is an effective bijection from $\mathbb{N}$ to Λ.

We recall a standard encoding of integers in λ-calculus.

Definition 3.28. [Church Numeral] Let $n \in \mathbb{N}$. Then, we set $\underline{n} = \lambda fx.(f)^n x$.

We also define the Gödel numbering representing $M \in \Lambda$, $\lceil M \rceil$:

Definition 3.29. [Gödel Number] Let $M \in \Lambda$. Then, we set $\lceil M \rceil = \underline{\#M}$, where $\#M$ is the Gödel number of M.

Definition 3.30. We denote the set of finite sequences of $\mathbb{N}$ by $(\mathbb{N})^*$, where ε is the empty sequence, $< i >$ is the sequence having only one element, $i \in \mathbb{N}$, and $\sigma \cdot \sigma'$ is the concatenation of σ and σ'.

Let $\#'$ be an effective injection from $(\mathbb{N})^*$ to $\mathbb{N}$.

Definition 3.31. For every $\sigma \in (\mathbb{N})^*$, $\lceil \sigma \rceil = \underline{\#'\sigma}$.

Remark 3.32. There is little danger of confusion between $\lceil M \rceil$ and $\lceil \sigma \rceil$, since the first is defined on Λ while the second is defined on $(\mathbb{N})^*$.

We now define *recursive enumerability* for Böhm Trees:

Definition 3.33. Let $\mathcal{B} \subseteq EBT$. We define $\#\mathcal{B} = \{\#a \mid a \in \mathcal{B}\} \subseteq \mathbb{N}$.

Definition 3.34. Let $\mathcal{B} \subseteq EBT$. Then $\mathcal{B}$ is R.E. if and only if $\#\mathcal{B}$ is R.E., i.e. either $\mathcal{B} = \emptyset$, or there exists a recursive function $\phi : \mathbb{N} \longrightarrow \mathbb{N}$ such that $\#\mathcal{B} = \{\phi(n) \mid n \in \mathbb{N}\}$.

Remark 3.35. The notion of a recursively enumerable subset of EBT's does not depend on the chosen Gödel enumeration.

We remark that all examples in the previous section of $\mathcal{B} = BT(M)$ satisfy the three conditions : $\mathcal{B}$ is a $\sqsubseteq$-ideal, $FV(\mathcal{B})$ is finite, and $\mathcal{B}$ is R.E.

Lemma 3.36. Let $M \in \Lambda$. Then $BT(M)$ is R.E.

Proof. Notice that $BT(M)$ is defined by an effective procedure, hence we conclude by Church's thesis. $\square$

3.3 The characterization theorem

We now want to characterize the subsets of EBT's coming from λ-terms. Until now we have shown that if $\mathcal{B} = BT(M)$ for a certain $M \in \Lambda$, then $\mathcal{B}$ is an ideal, R.E., and with a finite number of free variables. We now prove the converse. For doing so, we need some preliminary definitions and lemmas, as well as recalling two fixed-point theorems for the λ-calculus.

Definition 3.37. Let $\mathcal{B}$ be a set of EBT's. For $i \geq 0$, we write $\mathcal{B}_i$ the projection of $\mathcal{B}$ on the i^{th} son.

$$\mathcal{B}_i = \{e \in EBT \mid \exists \lambda x_0 \ldots x_{n-1}.(y) f_0 \ldots f_{i-1} e f_{i+1} \ldots f_{k-1} \in \mathcal{B} \wedge n \geq 0 \wedge k > i\}$$

Lemma 3.38. Let $\mathcal{B}$ be a set of EBT's. If $\mathcal{B}$ is R.E., then so is $\mathcal{B}_i$ for every i.

Proof. Just remark that the procedure computing $\mathcal{B}_i$ is effective. □

Lemma 3.39. Let $\mathcal{B}$ be a $\sqsubseteq$-ideal and $\lambda x_0 \ldots x_{n-1}.(y) b_0 \ldots b_{k-1} \in \mathcal{B}$. Then, if $FV(\mathcal{B})$ is finite, $FV(\mathcal{B}_i)$ is finite. Moreover, for every $i < k$, $\mathcal{B}_i$ is a $\sqsubseteq$-ideal.

Proof. We assume that $\mathcal{B}$ is a $\sqsubseteq$-ideal, and that $FV(\mathcal{B})$ is finite. We remark that if $\mathcal{B} = \emptyset$ then there is nothing to prove. Otherwise, there exists $b = \lambda x_0 \ldots x_{n-1}.(y) e_0 \ldots e_{k-1} \in \mathcal{B}$. Let $i \in \mathbb{N}$, and $k > i$. Then for all $x \in FV(\mathcal{B}_i)$, there exists $e \in \mathcal{B}_i$ such that $x \in FV(e)$, so that $FV(\mathcal{B}_i) \subseteq FV(e) \cup \{x_0, \ldots, x_{n-1}\}$, which is finite. Finally, we show that $\mathcal{B}_i$ is a $\sqsubseteq$-ideal.

- It is clear that $\Omega \in \mathcal{B}_i$, because $\mathcal{B}$ is a $\sqsubseteq$-ideal.
- Let $e_1 \in$ EBT, $e_2 \in \mathcal{B}_i$, s.t. $e_1 \sqsubseteq e_2$. If $e_2 = \Omega$, there is nothing to prove. Otherwise, there exists $e_2' = \lambda x_0 \ldots x_{n-1}.(y) f_0 \ldots f_{i-1} e_2 f_{i+1} \ldots f_{k-1} \in \mathcal{B}$, and $e_1' = \lambda x_0 \ldots x_{n-1}.(y) f_0 \ldots f_{i-1} e_1 f_{i+1} \ldots f_{k-1} \sqsubseteq e_2'$. As $\mathcal{B}$ is a $\sqsubseteq$-ideal, $e_1' \in \mathcal{B}$ and subsequently, $e_1 \in \mathcal{B}_i$.
- Let $e_1, e_2 \in \mathcal{B}_i$. There exists $e_1' = \lambda x_0 \ldots x_{n-1}.(y) f_0 \ldots f_{i-1} e_1 f_{i+1} \ldots f_{k-1} \in \mathcal{B}$, and $e_2' = \lambda x_0 \ldots x_{n-1}.(y) g_0 \ldots g_{i-1} e_2 g_{i+1} \ldots g_{k-1} \in \mathcal{B}$. As $\mathcal{B}$ is a $\sqsubseteq$-ideal, there exists $\lambda x_0 \ldots x_{n-1}.(y) l_0 \ldots l_{i-1} l_i l_{i+1} \ldots l_{k-1} \sqsupseteq e_1', e_2'$, and subsequently, $l_i \sqsupseteq e_1, e_2$. Finally, $l_i \in \mathcal{B}_i$ and $l_i \sqsupseteq e_1, e_2$.

□

We extend the definition of $\mathcal{B}_i$ to any sequence of integers:

Definition 3.40. Let $\sigma \in (\mathbb{N})^*$. Let $\mathcal{B}$ be a subset of EBT's. Then:

$$\mathcal{B}(\sigma) = \begin{cases} \mathcal{B}, & \text{if } \sigma = \varepsilon \\ \mathcal{B}(\sigma')_i, & \text{if } \sigma = \sigma' \cdot < i > \end{cases}$$

The following lemma is crucial for our theorem.

Lemma 3.41. [Uniformity] Let $\mathcal{B}$ be a $\sqsubseteq$-ideal. If there exists b_1, b_2 such that $b_1 = \lambda x_0 \dots x_{n-1}.(y)e_0 \dots e_{k-1}$ and $b_2 = \lambda y_0 \dots y_{m-1}.(z)f_0 \dots f_{k'-1}$ then $y = z$, $k = k'$ and $n = m$.

Proof. Let $b_1,\ b_2 \in \mathcal{B}$, such that $b_1 = \lambda x_0 \dots x_{n-1}.(y)e_0 \dots e_{k-1}$, and $b_2 = \lambda y_0 \dots y_{m-1}.(z)f_0 \dots f_{k'-1}$. As $\mathcal{B}$ is a $\sqsubseteq$-ideal, there exists $b \in \mathcal{B}$ such that $b \sqsupseteq b_1,\ b_2$. By definition of $\sqsubseteq$, $b = \lambda x_0 \dots x_{n-1}.(y)g_0 \dots g_{k-1} \sqsupseteq b_1$, and $\lambda y_0 \dots y_{m-1}.(z)h_0 \dots h_{k'-1} \sqsupseteq b_2$. Subsequently $n = m$, $y = z$, and $k = k'$. $\square$

By lemmas 3.39 and 3.41, a $\sqsubseteq$-ideal describes a tree labelled by $\lambda x_0 \dots x_{n-1}.(y)$ and branching on the arguments of y. Recalling the example 3.12, we can represent the tree associated with $BT(\lambda x_0.\ ((x_0)\ \lambda y.(x_0)y)\ x_0)$ as:

$$\begin{tikzcd}
 & \lambda x_0.\ x_0 \arrow[dl, dash] \arrow[dr, dash] & \\
\lambda y.x_0 \arrow[d, dash] & & x_0 \\
y & &
\end{tikzcd}$$

The reader can find a such a tree representation in the chapter 10 of [Bar], where Böhm trees are presented as trees.

Let $\mathcal{B}$ be a $\sqsubseteq$-ideal. We now explicit the subtree of a node at address σ:

Definition 3.42. Let $\mathcal{B}$ be a $\sqsubseteq$-ideal. The *arity* of a node σ, $\varrho_{\mathcal{B}}(\sigma)$, is defined by: $\varrho_{\mathcal{B}}(\sigma) = k$ if $\exists \lambda x_0 \dots x_{n-1}.(g)h_0 \dots h_{k-1} \in \mathcal{B}(\sigma)$. If $\mathcal{B}(\sigma) = \{\Omega\}$, then $\varrho_{\mathcal{B}}(\sigma)$ is undefined.

Remark 3.43. For $\mathcal{B} \subseteq$ EBT the function $\varrho_{\mathcal{B}}$ must be *well-defined*, which means that for $\sigma \in (\mathbb{N})^*$, if $\lambda x_0 \dots x_{n-1}.(g)h_0 \dots h_{k-1},\ \lambda y_0 \dots y_{n-1}.(h)l_0 \dots l_{m-1} \in \mathcal{B}(\sigma)$ then $k = m$. This is the case when $\mathcal{B}$ is a $\sqsubseteq$-ideal (see lemma 3.41).

We describe the label at node β of a $\sqsubseteq$-ideal:

Definition 3.44. Let $\mathcal{B}$ be a $\sqsubseteq$-ideal.

$$\mathcal{B}\{\beta\} = \begin{cases} \lambda x_0 \dots x_{n-1}.(y), & \text{if } \beta = \varepsilon \text{ and } \lambda x_0 \dots x_{n-1}.(y)b_0 \dots b_{k-1} \in \mathcal{B} \\ \mathcal{B}_i\{\beta'\}, & \text{if } \beta = \langle i \rangle \cdot \beta' \text{ and } i < \varrho_{\mathcal{B}}(\epsilon) \\ \uparrow & \text{elsewhere} \end{cases}$$

Notice that $\mathcal{B}\{\beta\}$ is defined if and only if $\mathcal{B}(\beta)$ is different from $\emptyset$ and $\{\Omega\}$. We then define the vector of free variables at a node σ:

Definition 3.45. Given a $\sqsubseteq$-ideal $\mathcal{B}$, and a list $\ell_{\mathcal{B}}$ of the free variables of $\mathcal{B}$, if $\mathcal{B}(\sigma) \neq \emptyset$ we define $FV^{\ell_{\mathcal{B}}}(\sigma)$ as follows:

$$FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) = \begin{cases} \ell_{\mathcal{B}} & \text{If } \sigma = \varepsilon \\ & \text{Where } \lambda x_0, \dots, x_{k-1}.(y) b_0 \dots b_{l-1} \in \mathcal{B} \\ FV_{\mathcal{B}_i}^{\ell_{\mathcal{B}} \cdot <x_0, \dots, x_{k-1}>}(\sigma') & \text{ and } \sigma = \sigma' \cdot < i > \end{cases}$$

Notice that we are using Lemma 3.39 and 3.41, inferring that $\mathcal{B}_i$ is an ideal whenever $\mathcal{B}$ is, and that the prefix $\lambda x_0 \dots x_{n-1}.(y)$ is the same for all elements of $\mathcal{B}$ different from Ω.

Lemma 3.46. Given $\mathcal{B}, \mathcal{B}'$ $\sqsubseteq$-ideals, if $\forall \beta \in (\mathbb{N})^*, \mathcal{B}\{\beta\} = \mathcal{B}'\{\beta\}$, then $\mathcal{B} = \mathcal{B}'$.

Proof. We show that for all $\beta \in (\mathbb{N})^*$, for $\mathcal{B}, \mathcal{B}'$ $\sqsubseteq$-ideals such that $\mathcal{B}\{\beta\} = \mathcal{B}'\{\beta\}$, we have that $\mathcal{B} \subseteq \mathcal{B}'$ (the other inclusion being symmetrical). The proof is by induction on the length of β. Let $b \in \mathcal{B}$, we have to show that $b \in \mathcal{B}'$. If $b = \Omega$, $b \in \mathcal{B}'$. Otherwise, $b = \lambda x_0 \dots x_{n-1}.(y) b_0 \dots b_{k-1}$. Therefore, as $\forall \beta, \mathcal{B}\{\beta\} = \mathcal{B}'\{\beta\}$, then $\mathcal{B}' \neq \{\Omega\}$. In particular, $\mathcal{B}'\{\varepsilon\} = \mathcal{B}\{\varepsilon\} = \lambda x_0 \dots x_{n-1}.(y)$ consequently there exists $b' = \lambda x_0 \dots x_{n-1}.(y) b'_0 \dots b'_{k'-1} \in \mathcal{B}'$. Furthermore, $k = k'$. If we suppose for instance $k < k'$, then we are in contradiction with the hypothesis, since $\mathcal{B}'\{< k' >\}$ is defined but not $\mathcal{B}\{< k' >\}$ while $\mathcal{B}'\{< k' >\} = \mathcal{B}\{< k' >\}$.

Let $i \in \{0, \dots, k-1\}$ and $c = \lambda x_0 \dots x_{n-1}.(y) b'_0 \dots b'_{i-1} b_i b'_{i+1} \dots b'_{k-1}$; we now prove that $c \in \mathcal{B}'$.

By induction, if $\beta =< i > \cdot \beta'$, then $\mathcal{B}_i\{\beta'\} = \mathcal{B}\{< i > \cdot \beta'\} = \mathcal{B}'\{< i > \cdot \beta'\} = \mathcal{B}'_i\{\beta'\}$, and $b_i \in \mathcal{B}'_i$. Therefore, there exists

$$d = \lambda x_0 \dots x_{n-1}.(y) d_0 \dots d_{i-1} b_i d_{i+1} \dots d_{k-1} \in \mathcal{B}'$$

Let $m \in \mathcal{B}'$ such that $b', d \sqsubseteq m$, $m = \lambda x_0 \dots x_{n-1}.(y) m_0 \dots m_i \dots m_{k-1}$. Then $b'_0 \sqsubseteq m_0, \dots, b'_{i-1} \sqsubseteq m_{i-1}, b_i \sqsubseteq m_i, b'_{i+1} \sqsubseteq m_{i+1}, \dots, b'_{k-1} \sqsubseteq m_{k-1}$, and consequently $c = \lambda x_0 \dots x_{n-1}.(y) b'_0 \dots b'_{i-1} b_i b'_{i+1} \dots b'_{k-1} \in \mathcal{B}'$.

By applying this procedure for every $i \in \{0, \dots, k-1\}$, we thus obtain that $b = \lambda x_0 \dots x_{n-1}.(y) b_0 \dots b_{k-1} \in \mathcal{B}'$, which concludes the proof. $\square$

For the proof of the characterization theorem, we need two fixed point theorems, from [Bar]. Recall that Λ^0 denotes the set of closed λ-terms.

Theorem 3.47. There exists $E \in \Lambda^0$ such that for all $M \in \Lambda^0$, $E \lceil M \rceil \twoheadrightarrow M$. Furthermore, if M is unsolvable, then $E \lceil M \rceil$ is unsolvable.

Theorem 3.48. For all $G \in \Lambda^0$, there exists $M \in \Lambda^0$ such that $M \twoheadrightarrow G \lceil M \rceil$.

Thus we obtain our characterization theorem:

Theorem 3.49. For all $\mathcal{B} \subseteq EBT$, there exists $M \in \Lambda$ such that $BT(M) = \mathcal{B}$ if and only if $\mathcal{B}$ is a $\sqsubseteq$-ideal, $FV(\mathcal{B})$ is finite, and $\mathcal{B}$ is R.E.

The left-to-right implication is a trivial consequence of the lemmas 3.18, 3.24, and 3.36. We now show the right-to-left implication.

By theorem 3.47, we have $E \in \Lambda^0$ so that for all $M \in \Lambda^0$, $E \lceil M \rceil \twoheadrightarrow M$, and we will use this E.

Let $\mathcal{B}$ be a $\sqsubseteq$-ideal, such that $FV(\mathcal{B})$ is finite, and $\mathcal{B}$ is R.E.

If $\sigma \in (\mathbb{N})^*$ and $\mathcal{B}(\sigma) \neq \emptyset$, or $\{\Omega\}$, then $\mathcal{B}(\sigma)$ and $\varrho_{\mathcal{B}}(\sigma)$ are defined: let $\mathcal{B}\{\sigma\} = \lambda x_0 \dots x_{n-1}.(y)$, we set:

$$\begin{aligned} C_\sigma =& \lambda m \lambda FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) \lambda x_0 \dots x_{n-1}.(y) \\ & (Em \lceil \sigma \cdot < 0 > \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma \cdot < 0 >) \dots (Em \lceil \sigma \cdot < \varrho_{\mathcal{B}}(\sigma) - 1 > \rceil \\ & FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma \cdot < \varrho_{\mathcal{B}}(\sigma) - 1 >) \end{aligned}$$

We remark that $C_\sigma \in \Lambda^0$. Furthermore, as $\mathcal{B}$ is R.E., by remark 3.27, there is an effective procedure that computes C_σ.

Let Ψ be a partial recursive function $\Psi(\sigma) = \begin{cases} \# C_\sigma, & \text{if } \mathcal{B}(\sigma) \neq \emptyset \text{ and } \neq \{\Omega\} \\ \uparrow & \text{elsewhere} \end{cases}$

The function F λ-defines Ψ: $F \lceil \sigma \rceil = \begin{cases} \lceil \Psi(\sigma) \rceil = \lceil C_\sigma \rceil & \text{if } \mathcal{B}(\sigma) \neq \emptyset \text{ and } \neq \{\Omega\} \\ \uparrow & \text{elsewhere} \end{cases}$

We now construct the λ-term N such that $BT(N) = \mathcal{B}$. Let $F' = \lambda m \lambda s.((E)(F)s)m$. By theorem 3.48, there exists $M \in \Lambda^0$, such that $M \twoheadrightarrow F' \lceil M \rceil \to \lambda s.((E)(F)s) \lceil M \rceil$.

If $\Psi(\sigma) \downarrow$, i.e. if $\mathcal{B}(\sigma) \neq \emptyset$ and $\neq \{\Omega\}$:

$$\begin{aligned} M \lceil \sigma \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) & \twoheadrightarrow E(F \lceil \sigma \rceil) \lceil M \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) && \text{by definition of } F \\ & \twoheadrightarrow E(\lceil C_\sigma \rceil) \lceil M \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) \\ & \twoheadrightarrow C_\sigma \lceil M \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma) && \text{by definition of } E \\ & \twoheadrightarrow \lambda x_0 \dots x_{n-1}.(y)(E \lceil M \rceil \lceil \sigma \cdot < 0 > \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma \cdot < 0 >) \dots \\ & (E \lceil M \rceil \lceil \sigma \cdot < \varrho_{\mathcal{B}}(\sigma) - 1 > \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma \cdot < \varrho_{\mathcal{B}}(\sigma) - 1 >) \\ & \twoheadrightarrow \lambda x_0 \dots x_{n-1}.(y)(M \lceil \sigma \cdot < 0 > \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma \cdot < 0 >) \dots && \text{by definition of } E \end{aligned}$$

If $\mathcal{B}(\sigma) \uparrow$ (i.e. $\mathcal{B}(\sigma) = \emptyset$ or $= \{\Omega\}$) , then $M \lceil \sigma \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma)$ is unsolvable.

The following claim results from the definition of the label at node β:

Claim 3.50. For all $\sigma, \beta \in (\mathbb{N})^*$, we have $BT(M \lceil \sigma \rceil \, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma))\{\beta\} = \mathcal{B}(\sigma)\{\beta\}$.

Proof. We show this claim by induction on β. If $\beta = \varepsilon$ we already have the equality, $BT(M\lceil\sigma\rceil\, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma))\{\varepsilon\} = \lambda x_0 \ldots x_{n-1}.(y) = \mathcal{B}(\sigma)\{\varepsilon\}$. If $\beta =< i > \cdot\beta'$,

$$\begin{aligned} BT(M\lceil\sigma\rceil\, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma))\{< i > \cdot\beta'\} &= BT(M\lceil\sigma\cdot < i >\rceil\, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\sigma\cdot < i >))\{\beta'\} \\ &= \mathcal{B}(\sigma\cdot < i >)\{\beta'\} = \mathcal{B}(\sigma)\{< i > \cdot\beta'\} \end{aligned}$$

□

The result follows: for all $\beta \in (\mathbb{N})^*$, $\mathcal{B}\{\beta\} = \mathcal{B}(\varepsilon)\{\beta\} = BT(M\lceil\varepsilon\rceil\, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\varepsilon))\{\beta\}$. Therefore, by lemma 3.46, we have $\mathcal{B} = BT(M\lceil\varepsilon\rceil\, FV_{\mathcal{B}}^{\ell_{\mathcal{B}}}(\varepsilon))$, concluding the proof of the theorem.

4 Coherent spaces resulting from resource terms

4.1 Taylor Expansion

4.1.1 Introduction to resource λ-calculus

First, we introduce the resource λ-calculus ($\Delta^{(!)}$ for short), which is a resource sensitive variant of the λ-calculus.

Definition 4.1. We define the set Δ of *simple terms*, and the set $\Delta^!$ of *simple poly-terms* by mutual recursion:

- $\Delta = s,\, t ::= x \mid \lambda x.s \mid s[t_1, \ldots, t_k]$ where x is a variable
- $\Delta^! = S,\, T ::= 1 \mid [s] \mid TS = [t_1, \ldots, t_k][s_1, \ldots, s_l] = [t_1, \ldots, t_k, s_1, \ldots, s_l]$

In fact, $\Delta^!$ is the set of finite multisets of simple terms, in which we use the multiplication operator, representing disjoint union, 1 represents the empty multiset. Finally, $\Delta^{(!)} = \Delta \sqcup \Delta^!$.

Example 4.2. An example of simple term is $\lambda x.x[y, y, z] = \lambda x.x[y^2, z]$, and $[s, s, t, s] = [s^3, t]$ is a simple poly-term.

Remark 4.3. We extend in a trivial way the notions of free and bound variables coming from the λ-calculus.

4.1.2 Linear combinations and reduction

We now work on the free module $\{0, 1\}[\Delta]$ over the commutative semi-ring $(\{0, 1\}, \max, \min, 0, 1)$, generated by Δ. In [EhrReg2], the calculus is studied for any commutative semi-ring; here we restrict our study to the case of the semi-ring $\{0, 1\}$, so the free module generated by Δ is simply the powerset of Δ with the union as addition, and 0 as the empty set $\emptyset$.

We define a reduction in the resource calculus, similar to the reduction of λ-terms.

Definition 4.4. A redex is a simple term of the shape $r = (\lambda x.s)\,[s_1, \dots, s_k]$.

Definition 4.5. [Resource Reduction] The single-step reduction in the resource calculus, denoted by $\longrightarrow_r \subseteq \Delta^{(!)} \times \{0,1\}[\Delta]$, is the smallest relation that satisfies the following rules:

$$\frac{t \longrightarrow_r \alpha}{\lambda x.t \longrightarrow_r \lambda x.\alpha = \{\lambda x.s \mid s \in \alpha\}} \qquad \frac{t \longrightarrow_r \alpha}{tT \longrightarrow_r \alpha T = \{sT \mid s \in \alpha\}}$$

$$\frac{t \longrightarrow_r \alpha}{[t] \longrightarrow_r \{[s]; s \in \alpha\}} \qquad \frac{T \longrightarrow_r \Pi}{(S)T \longrightarrow_r S\Pi = \{ST' \mid T' \in \Pi\}}$$

Finally, if $r = \lambda x.s\,[s_1, \dots, s_k]$ and k is distinct from the number of free occurrences of x in s, then $r \longrightarrow_r \emptyset$, r reduces to the empty set. Otherwise, r reduces to $\{s \ll s_1/x_{f(1)}, \dots, s_k/x_{f(k)} \gg \mid f \in \sigma_k\} \in \{0,1\}[\Delta]$ where $x_1, \dots, x_k$ are the free occurrences of x in s, and where σ_k stands for the group of all permutations on the set $\{1, \dots, k\}$.

We also want $\longrightarrow_r$ to satisfy the following rules:

$$\frac{t \longrightarrow_r \alpha}{\{t\} \cup \beta \longrightarrow_r \alpha \cup \beta} \qquad \frac{T \longrightarrow_r \Pi}{\{T\} \cup \rho \longrightarrow_r \Pi \cup \rho}$$

The reflexive transitive closure is denoted by the same convention as in λ-calculus: $\twoheadrightarrow_r \subseteq \{0,1\}[\Delta^{(!)}] \times \{0,1\}[\Delta^{(!)}]$. Constructions of this syntax are linear.

Example 4.6. We give a few examples in order to understand the $\longrightarrow_r$- reduction:
The simple resource term $(\lambda x.x[x])1$ reduces to the emptyset $\emptyset$.
$(\lambda x.x[x])[y] \longrightarrow_r \emptyset$.
$(\lambda x.x[x])[y^3] \longrightarrow_r \emptyset$.
$(\lambda x.x[x])[y, z] \longrightarrow_r \{y[z], z[y]\}$.
$(\lambda x.x[x])[y, y] \longrightarrow_r \{y[y]\}$.
Notice that we are not counting the number of times we have $y[y]$, as we are in $\{0,1\}[\Delta]$.

Example 4.7. We have that $(\lambda x.x[x])[\lambda x.x[x]] \longrightarrow_r \emptyset$, and $(\lambda x.x[x^2])[(\lambda x.x[x])^3] \longrightarrow_r \{(\lambda x.x[x])\,[\lambda x.x[x], \lambda x.x[x]]\}$ which reduces to $\emptyset$.

Example 4.8. Let $r = (\lambda fx.f[f[x], f[x]])[\lambda gy.g[g[y]], \lambda z.z, \lambda z.z]$. Then r reduces to $\{\lambda x.(\lambda gy.g[g[y]])[(\lambda z.z)[x], (\lambda z.z)[x]]\}$. Finally, r reduces to $\{\lambda x.\lambda y.x[x[y]]\}$.

We denote by Δ_n (resp. Δ^0) the set of all normal simple terms (resp. of all closed simple terms).

Theorem 4.9. [EhrReg2] The resource reduction relation is confluent, and strongly normalizing.

To prove that the reduction relation is strongly normalizing, remark that if $t \longrightarrow_r s_i$, then the number of symbols of its s_i is strictly less than the number of symbols in t. Then, confluence follows by Newman's lemma.

For every $\rho \in \{0,1\}[\Delta^{(!)}]$ we define $NF(\rho)$:

Definition 4.10. Let $\rho \in \{0,1\}[\Delta^{(!)}]$. The function $NF : \{0,1\}[\Delta^{(!)}] \longrightarrow \{0,1\}[\Delta_n^{(!)}]$ associates ρ to its normal form.

Definition 4.11. Let r be a simple term. Then r is an *elementary resource term* (ERT) if r is in normal form.

4.1.3 Taylor Expansion

We now describe the Taylor expansion of a λ-term, where the term is expressed as a possibly infinite set of simple terms.

Definition 4.12. Let $M \in \Lambda$. The *Taylor expansion* of M, $\tau(M)$. $\tau : \Lambda \longrightarrow \mathcal{P}(\Delta)$ is:

- If $M = x$, then $\tau(M) = \{x\}$;
- If $M = \lambda x.N$, then $\tau(M) = \{\lambda x.s \mid s \in \tau(N)\}$;
- If $M = (P)Q$, then $\tau(M) = \{pQ = p[q_1, \ldots, q_k] \mid p \in \tau(P); k \in \mathbb{N}; q_1, \ldots, q_k \in \tau(Q)\}$.

Example 4.13. The Taylor expansion of $\lambda x.(x)y$ is $\{\lambda x.x[y^n] \mid n \in \mathbb{N}\}$.

Example 4.14. The Taylor expansion of $\underline{\mathbf{2}} = \lambda fx.(f)(f)x$ is

$$\{\lambda fx.f[f[x^{n_1}], \ldots, f[x^{n_k}]] \mid k \in \mathbb{N}; n_1, \ldots, n_k \in \mathbb{N}\}$$

We extend τ from $\mathcal{P}(EBT)$ to $\mathcal{P}(\Delta)$: $\tau(\mathcal{B}) = \bigcup_{b \in \mathcal{B}} \tau(b)$, and, as a convention, $\tau(\Omega) = \emptyset$. We finally recall the main theorem from [EhrReg2], which makes a link between Böhm trees and Taylor expansion:

Theorem 4.15. [Ehrhard Régnier] Let $M \in \Lambda$, then $\tau(BT(M)) = NF(\tau(M))$.

In order to illustrate this crucial theorem, we give examples:

Example 4.16. Consider the λ-term $M = (\lambda x.(x)x)\lambda x.(x)x$. Then $\tau(BT(M)) = \tau(\Omega) = \emptyset$. We have $\tau(M) = \{\lambda x.x[x^n][\lambda x.x[x^{n_1}], \ldots, \lambda x.x[x^{n_k}]] \mid n, k, n_1, \ldots, n_k \in \mathbb{N}\}$. It can be proved by induction on $n, k, n_1, \ldots, n_k$, that $\tau(M)$ reduces to $\emptyset$. Therefore, $NF(\tau(M)) = NF(\emptyset) = \emptyset$.

Example 4.17. On the one hand, we have

$$\begin{aligned}\tau(BT(\lambda fx.\,(f)(f)(\lambda y.y)x)) &= \tau(BT(\underline{\mathbf{2}})) = \tau(\{\Omega, \lambda fx.\,(f)\Omega, \lambda fx.\,(f)(f)x\}) \\ &= \{\lambda fx.f[f[x^{n_1}], \ldots, f[x^{n_k}]] \mid k \in \mathbb{N}; n_1, \ldots, n_k \in \mathbb{N}\}\end{aligned}$$

On the other hand, we have $NF(\tau(\lambda fx.\,(f)(f)(\lambda y.y)x))$
$= NF(\{\lambda fx.f[f[((\lambda y.y)x)^{n_1}],\dots,f[((\lambda y.y)x)^{n_k}]]] \mid k \in \mathbb{N}; n_1,\dots,n_k \in \mathbb{N}\})$
$= \{\lambda fx.f[f[x^{n_1}],\dots,f[x^{n_k}]] \mid k \in \mathbb{N}; n_1,\dots,n_k \in \mathbb{N}\}$
$= \tau(BT(\lambda fx.\,(f)(f)(\lambda y.y)x))$

4.2 Towards the characterization theorem for resource calculus

Recall that we can see λ-terms as computer programs, where the Böhm tree of a term/program describes the interaction between the program and the external environment, and where Taylor expansion is a quantitative refinement of Böhm trees. To every λ-term M, we can associate its Taylor expansion $\tau(M)$, and we would like to characterize the sets of simple terms that come from the Taylor expansion of a λ-term.

We easily adapt the definition of free variables to resource calculus, as well as the notion of recursive enumerability.

We now define the binary *coherence* relation (see [EhrReg1] section 3):

Definition 4.18. The coherence relation on resource terms, denoted by $\frown$, is defined by induction on simple terms:

- $x \frown t'$ if $t' = x$,
- $\lambda x.s \frown t'$ if $t' = \lambda x.s'$, with $s \frown s'$,
- $sT \frown t'$ if $t' = s'T'$ with $s \frown s'$ and $T \frown T'$,
- $[s_1,\dots,s_n] \frown [s_{n+1},\dots,s_m]$ if for all $i,j \in \{1,\dots,m\}$, $s_i \frown s_j$.

When two terms are not coherent, we say that they are *incoherent*. Incoherence is denoted by the symbol $\asymp$.

Remark 4.19. Coherence is not an equivalence relation. It is not transitive, since if $y \neq z$ then $x[y] \frown x1$ and $x1 \frown x[z]$, but $x[y] \asymp x[z]$. It is not reflexive, since if $x \asymp y$ then $[x,y] \asymp [x,y]$.

Definition 4.20. A term $\sigma \in \Delta^{(!)}$ is *uniform* if $\sigma \frown \sigma$.

Definition 4.21. We call *clique* a subset of resource terms U such that, for every $\tau, \tau' \in U$, $\tau \frown \tau'$.

Theorem 4.22. For every $M \in \Lambda$, $\tau(M)$ is a maximal clique in $(\Delta, \frown)$.

Proof. We prove the theorem by induction on M:

- If $M = x$, then $\tau(M)$ is a clique. It is maximal, as any element different from x cannot be coherent with x.

- If $M = \lambda x.N$, then $\tau(M) = \{\lambda x.s \mid s \in \tau(N)\}$. As $\tau(N)$ is a clique, by induction hypothesis, $\tau(M)$ is a clique. Let U be a maximal clique that includes $\tau(M)$. Then, for any $r \in U$, $r = \lambda x.s$, where $s \frown s'$ for any $s' \in \tau(N)$. As $\tau(N)$ is a maximal clique, by induction hypothesis, $s' \in \tau(N)$, therefore $r \in \tau(M)$.
- If $M = (P)Q$, then $\tau(M) = \{p[q_1, \dots, q_k] \mid p \in \tau(P); k \in \mathbb{N}; q_1, \dots, q_k \in \tau(Q)\}$. By induction hypothesis on $\tau(P)$ and $\tau(Q)$, we deduce that $\tau(M)$ is a maximal clique.

□

We obtain three necessary conditions for a subset U of Δ to be the Taylor expansion of a λ-term:

- U has a finite number of free variables,
- U is a clique,
- U is recursively enumerable.

For instance, the set $\{x, y\}$ is not a clique, and does not come from a λ-term. The set $\{x_1 1, x_1[x_2 1], \dots\}$ is a clique, recursively enumerable, but has an infinite number of free variables, and does not come from a λ-term. Finally, we define the sequence:

$$r_n = \lambda xy.u_0[\dots[u_n 1]\dots] = \begin{cases} \lambda xy.u_0[u_1[\dots[u_{n-1}[x1]\dots] & \text{if program } n \text{ halts on input } n \\ \lambda xy.u_0[u_1[\dots[u_{n-1}[y1]\dots] & \text{otherwise} \end{cases}$$

The set $\{r_n \mid n \in \mathbb{N}\}$ has a finite number of free variables, is a clique, but not recursively enumerable, and does not come from a λ-term, as the Taylor expansion is an effective procedure.

We have shown by examples that these three conditions are necessary for a subset U of Δ to be the imageset of the normal form of a Taylor expansion of a λ-term. The next step is to show that these conditions are sufficient.

Conclusions

This work suggests a number of interesting directions to explore in further developments. The role of the scalar coefficients in formal series of resource terms deserves a deep investigation, to uncover a possible connection between the scalar value and the time and space complexity. Another direction is to study formal series of resource terms *not* coming from a λ-term, where there is a superposition of inconsistent information. Take the example of a program that tests twice a boolean variable b, and returns true if the two values are the same and false if the two values are different. In a pure functional programming language, the value of a variable cannot be changed once set, so this example would always return true. In a nondeterministic setting, the value of b might change between the two tests, making necessary to model inconsistent information.

References

[Bar] Henk P. Barendregt. The Lambda Calculus: Its Syntax and Semantics. *North-Holland, Revised edition*, 1985.

[EhrReg1] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 2006.

[EhrReg2] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. *Computability in Europe*, 2006.

[Ehr] Thomas Ehrhard. Notes de cours de Master LMFI et MPRI. `http://www.pps.univ-paris-diderot.fr/~ehrhard/cours-MPRI-LMFI.pdf`

[Sel] Peter Selinger. Lecture Notes on the Lambda Calculus. `http://www.mathstat.dal.ca/~selinger/papers/lambdanotes.pdf`

[Plot] Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science 5: 223–255*, 1977.

[EhrReg3] Thomas Ehrhard and Laurent Regnier.. The differential lambda-calculus. *Theoretical Computer Science 309, 1-3*, 2003.

[Rey] John C. Reynolds. Idealized ALGOL and its specification logic. *ALGOL-like Languages, Volume 1, pages 125 - 156* , 1997.

[PagTas] Michele Pagani and Christine Tasson. The Inverse Taylor Expansion Problem in Linear Logic. *LICS 2009: 222-231* , 2009.

[PagMan] Michele Pagani and Giulio Manzonetto. Böhm's theorem for resource lambda calculus through Taylor expansion. *TLCA'11 Proceedings of the 10th international conference on Typed lambda calculi and applications, pages 153-168* , 2011.

[Guha] Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi. The essence of javascript. *ECOOP'10 Proceedings of the 24th European conference on Object-oriented programming, pages 126-150* , 2010.