Computing with Böhm trees

René David

Fundamenta Informaticae XX (2000) 53–77 53
IOS Press

Computing with Böhm Trees

René David*
Laboratoire de Maths. Campus Scientifique
F-73376 Le Bourget du Lac.
Email: david@univ-savoie.fr

Abstract. This paper develops a general technique to analyze the head reduction of a term in a context. This technique is used to give a direct proof of the theorem of Hyland and Wadsworth : two λ-terms that have the same Böhm trees, up to (possibly infinite) η-equivalence, are operationally equivalent. It is also used to prove a conjecture of R. Kerth : Every unsolvable λ-term has a decoration. This syntactical result is motivated by (and gives the solution to) a semantical problem.

Keywords: λ-calculus, Böhm trees, head reduction, operational equivalence, decoration.

1. Introduction

In this paper I develop a technique to analyze the head reduction of a term in a context. This technique was first initiated in [7] (where it is called *the directed λ-calculus*) to give a syntactic proof of Krivine's theorem on storage operators. The version developed here is a bit more general than the one in [7]. The basic idea simply consists in giving names to the parts of the Böhm trees that we do not have to know for an head reduction step.

A theorem of Hyland and Wadsworth

I give a direct proof of the following well known theorem (due to Hyland and Wadsworth).

Theorem 1.1. Let t and t' be λ-terms. Assume they have the same Böhm tree up to (possibly infinite) η-equivalence (the precise definition is given in section 3). Then t and t' are operationally equivalent, i.e. for every context C, $C(t)$ is solvable iff $C(t')$ is solvable.

*Address for correspondence: Laboratoire de Maths. Campus Scientifique, F-73376 Le Bourget du Lac., Email: david@univ-savoie.fr

The original proofs of Hyland [11] and Wadsworth [13] were indirect : They show that both properties are equivalent to having the same interpretation in Scott's D_∞ model. Note that, in [11] and [13], the theorem is stated in another (equivalent) way that do not speak of Böhm trees. This notion has been introduced later by Barendregt (cf. [2]).

Due to the precise analysis of head reduction that is done, the result I get is, in fact, a bit more precise. I show that, if t and t' have the same Böhm trees (up to η-infinite equivalence) and C is some context, then the head reduction of $C(t)$ and $C(t')$ are *essentially the same* (and thus one terminates iff the other does) in the following sense. The head reduction of respectively $C(t)$ and $C(t')$ can be divided into two levels : the β-reductions which correspond to the computation of some node in the Böhm tree of respectively t and t'. These steps depend, of course, on t and t'. The other ones which, intuitively, correspond to the interaction between the Böhm trees and the context. The key point of the proof is the fact that the latter are the same for $C(t)$ and $C(t')$ except for some *administrative* β-reductions that look like $(\lambda x.(t\ x)\ u) \to_\beta (t\ u)$ and correspond to the difference between the Böhm trees in term of η.

Note that, since the theorem of Hyland and Wadsworth, other separability results have been proved. For example see [4].

Kerth's conjecture

I also use the technique to prove a conjecture stated by R. Kerth in [9]. This is a new result that already appears in [5].

Theorem 1.2. Every unsolvable λ term has a decoration.

To give the idea of the conjecture, I need the following, informal, definition: If t reduces to t' by some steps of head reduction, say that a sub-term d' of t' is a descendent (cf. definition 4.3) of a sub-term d of t if it is a "copy" of d.

Let t be unsolvable. Denote by t_k the term obtained from t after k many steps of head reduction. A sequence $(d_k)_{k\in N}$ of λ terms is a decoration for (the head reduction of) t if there is a strictly increasing function f from N to N such that for every k :

1. $t_{f(k)} = \overrightarrow{\lambda}(d_k\ \overrightarrow{u_k})$ for some finite (non empty) sequence $\overrightarrow{u_k}$ of λ terms.
2. d_k is solvable and d_{k+1} is a descendent of some element of the sequence $\overrightarrow{u_k}$ of arguments of d_k.

Some examples of decorations are give at the beginning of section 4.

The motivation (see [9]) of this conjecture is the following : A model of λ calculus is said to be sensible if all the unsolvable terms are equal in this model. It is not easy, in general, to check whether a given model of λ calculus is sensible or not. In [8] , [10] R Kerth built an uncountable number of graph models with different equational theories but he was unable to prove they were sensible, because the usual argument of reducibility did not work in his models. He was able to show that his models had no critical sequences (a semantical notion he introduced) and he showed that a graph model without critical sequences is sensible ... if his conjecture is true.

Thus, the constructions in [8] , [10] and the present paper show that there are uncountably many sensible distinct equational theories of continuous models (and similarly for the stable and strongly stable semantics).

Some general remarks. The technique developed here shows that a precise analysis of head reduction may give proofs that correspond to the intuition.

I thus hope this technique will help to prove unknown results for which there is an intuitive argument for their validity but for which we have not yet a definitive proof.

As already mentioned, the technique basically consists is using indirectly (infinite) Böhm trees by giving names to the infinite part of the tree that is not used. A notion of infinite λ-terms is introduced in [3]. Another direct proof of theorem 1.1 could be given by using such trees.

The paper is organized as follows : Except for the basic facts on λ-calculus, Böhm trees, ... that can be found in the usual textbooks on λ-calculus (for example [1], [2], [12], ...), this paper is self-contained. Section 2 gives the main definitions and properties of the directed λ-calculus. In section 3, I prove the theorem of Hyland and Wadsworth and in section 4, I prove Kerth's conjecture.

Acknowledgement. Thanks to C. Berline and R. Kerth for helpful discussions on the subject and also to the anonymous referees who suggested many improvements.

2. Computing with Böhm trees

2.1. Definitions

Definition 2.1. 1. An address is a finite list of $\mathrm{positive}$ integers.

2. Let a, a' be addresses. $a \leq a'$ means that a is an initial segment of a'.

3. Let a be an address.

   - $lg(a)$ denotes the length of a.
   - For $i \leq lg(a)$, $a \restriction i$ denotes the restriction of a to its first i elements.
   - $i + a$ (resp. $a + i$) denotes the list a with i added at the beginning (resp. at the end). Thus, $lg(i + a) = lg(a + i) = lg(a) + 1$.

4. The empty list is denoted by nil.

Definition 2.2. 1. Λ denotes the set of λ terms.

2. The set Λ' of terms is defined by the following grammar :

$$\Lambda' = V \mid \bot \mid c(a, \sigma) \mid \ \lambda x\ \Lambda' \mid (\Lambda'\ \Lambda')$$

where

   (a) V is the set of variables

   (b) A substitution is a function from a finite subset of V (called its domain) to Λ'. The empty substitution is denoted by $\emptyset$.

   (c) For every address a and every substitution, $c(a, \sigma)$ is a constant.

3. A Böhm function is a partial function f from the set of addresses into $\{\perp\} \cup \{(E, x, p) \ / \ E \subset V, E \text{ finite}, x \in V, p \in N\}$ which satisfies :

   (a) $f(nil)$ is defined.

   (b) $f(a+i)$ is defined iff $f(a) = (E, x, p)$ and $1 \leq i \leq p$.

Notations, conventions and comments

- I adopt Barendregt's convention that variables are always named in such a way that there is no undesired capture and no confusion between different names.
- $\overrightarrow{\lambda}$ denotes a sequence (possibly empty) of abstractions and $(t\ \overrightarrow{r})$ represents the term t applied to a sequence (possibly empty) of arguments.
- Recall that the Böhm tree of t (denoted by $BT(t)$) is defined as follows :
  - If the head reduction of t does not terminate, then $BT(t) = \perp$
  - If it terminates with $\lambda x_1 \ ... \ x_n \ (x \ t_1 \ ... \ t_k)$, then the root of $BT(t)$ is a node labelled with $\lambda x_1 \ ... \ x_n \ (x)$. It has k immediate successors : $BT(t_1), \ ..., \ BT(t_k)$.
- $c(a, \sigma)$ represents the term associated to the sub-tree (at the address a, in the environment given by σ) of the Böhm tree of some term u that will be substituted latter on.
- A Böhm function f codes a Böhm tree. It is an oracle that gives, on request of an address a in the tree, the following information: the variables that are abstracted, the head variable and the number of arguments. Thus $f(a) = (\{x_1, \ ..., \ x_n\}, x, p)$ means that the node at the address a in the Böhm tree coded by f is $\lambda x_1 \ ... \ x_n \ (x \ t_1 \ ... \ t_p)$ for some terms $t_1, \ ..., t_p$.
- In the following, the letters $a, b, c, \ ...$ are reserved for addresses, the letters $f, g, \ ...$ for Böhm functions and the letters $r, s, t, ...$ for terms.

Definition 2.3. Let σ be a substitution and $t \in \Lambda'$. $\sigma(t)$ is defined by the usual rules and

- $\sigma(c(a, \tau)) = c(a, \sigma \circ \tau)$ for every τ and a.
- $\sigma(\perp) = \perp$.

Lemma 2.4. Every term in Λ' can be written (in a unique way) as $\overrightarrow{\lambda} \ (R \ \overrightarrow{r})$ where R is either a variable or $\perp$ or $((\lambda x \ u) \ v)$ or $c(a, \sigma)$.

Proof: By induction on the term. □

Definition 2.5. Let $t = \overrightarrow{\lambda} \ (R \ r_1 \ ... \ r_q) \in \Lambda'$ and f be a Böhm function. One step of f -reduction of t is defined as follows :

1. If $R = x$ then t is in f-head normal form and t has no f -reduct.

2. If $R = \perp$

   (a) If $t = \perp$ then t is in f-head normal form and t has no f -reduct.

   (b) otherwise, the f-reduct of t is $\perp$.

3. If $R = ((\lambda x\ u)\ v)$ then the f-reduct of t is $\overrightarrow{\lambda}\ (\sigma(u)\ r_1\ ...\ r_q)$ where $\sigma(x) = v$.

4. If $R = c(a, \sigma)$

   (a) If $f(a) = \perp$, then the f-reduct of t is $\perp$.

   (b) If $f(a) = (\{x_1,\ ...,\ x_k\}, x, p)$ then the f-reduct of t is :
   $\overrightarrow{\lambda}\lambda x_{j+1}\ ...\ \lambda x_k\ \ \sigma'(x)\ c(a+1, \sigma')\ ...\ c(a+p, \sigma')\ r_{j+1}\ ...\ r_q$
   where $j = Min(k, q)$, $\sigma' = \tau \circ \sigma$ and τ is defined by $\tau(x_i) = r_i$ for $1 \leq i\ \leq j$.

   (c) If $f(a)$ is undefined, the f-reduct of t is not defined.

Definition 2.6. 1. $t \rightarrow_f t'$ (resp. $t \twoheadrightarrow_f t'$, resp. $t \rightarrow_f^+ t'$) means that t' is the f-reduct of t (resp. t' is obtained from t by some, possibly zero, steps of f-reductions, resp. t' is obtained from t by at least one step of f-reduction).

2. $hnf\,(f, t)$ (the f-head normal form of t) is defined by

   - If some step of the f-reduction of t is undefined, then $hnf\,(f, t)$ is not defined.
   - If $t\ \ \twoheadrightarrow_f t'$ for some term t' in f-head normal form and $t' \neq \perp$, then $hnf\,(f, t) = t'$. In this case t is said to be f-solvable.
   - If the f-reduction of t does not terminate or if $t\ \ \twoheadrightarrow_f \perp$, then $hnf\,(f, t) = \perp$. In this case t is said to be f-unsolvable.

Comments and examples

- Let $F = \lambda x \lambda y\ (y\ x)$ and $t' = (c(nil, \emptyset\ )\ F\ (c(nil, \emptyset)\ F))$. Define f by : $f(nil) = (\{x\}, x, 1)$ and, for $n \geq 1$, $f(1^n) = (\emptyset, x, 1)$ where $1^n = [1, 1, 1..., 1]$. The f-reduction of t' is given by (where $\sigma(x) = F$) :

$$\begin{array}{lll} t' & \rightarrow_f & (F\ c([1], \sigma)\ (c(nil, \emptyset)\ F)) \\ & \rightarrow & (c(nil, \emptyset)\ F\ c([1], \sigma)) \\ & \rightarrow_f & (F\ c([1], \sigma)\ c([1], \sigma)) \\ & \rightarrow & (c([1], \sigma)\ c([1], \sigma)) \\ & \rightarrow_f & (F\ c([1,\ 1], \sigma)\ c([1], \sigma)) \\ & \twoheadrightarrow_f & ... \end{array}$$

- If $t \in \Lambda'$ and f "represents" the term $u \in \Lambda$, the f-reduction "corresponds" to the (ordinary) head reduction of t' (denoted $t\langle u\rangle$ in definition 2.10 below) where

  - t' is the term t where each constant $c(a, \sigma)$ is replaced (iteratively) by the term (in the environment σ) whose Böhm tree is the sub-tree of $BT(u)$ at the address a.
  - "corresponds" means that the reduction is the same except that the part of the computation of t' that "comes from" the computation of the node at the address a in the Böhm tree of u has been forgotten and is given by the "oracle" f.

- In the example above, f corresponds to a fixed-point operator.
- Note that the f-reduction is a *big-step* head reduction. Assume f corresponds to the term u. In definition 2.5, for $R = c(a, \sigma)$, one step of reduction consists in : first replace $c(a, \sigma)$ by the corresponding node of $BT(u)$ and next *reduce all the redexes* that are immediately introduced. This is done in this way for the following reason. For example, assume $u \twoheadrightarrow \lambda x\ u_1 \twoheadrightarrow \lambda x\ (x\ v)$. The head reduction of $(u\ r)$ is : $(u\ r) \twoheadrightarrow (\lambda x\ u_1\ r) \to u_1[x := r] \twoheadrightarrow (r\ v[x := r])$. This is not exactly the same order as : first compute the node of the Böhm tree and then reduce since it would be : $(u\ r) \twoheadrightarrow (\lambda x\ u_1\ r) \twoheadrightarrow (\lambda x\ (x\ v)\ r) \twoheadrightarrow (r\ v[x := r])$. If the f-reduction was defined in another way, proposition 2.15 would be more difficult to state.
- I allow $f(a)$ to be undefined in the definition of the f-reduction of t because I made no restrictions in the definition of Λ'. However the typical situation where the f-reduction is used is the following: let $t = C(u) \in \Lambda$ where C is some context (i.e. a term with a hole), $t' = C(c(nil, \emptyset))$ and f "represents" u. In this case the f-reduction will clearly always be defined.

Convention. If $t \in \Lambda$ the f-reduction is the ordinary head reduction (f is never used and thus can be anything). In this case, I will omit the symbol f and write $t \twoheadrightarrow t'$ instead of $t \twoheadrightarrow_f t'$ and $hnf(t)$ instead of $hnf(f, t)$. More generally, in the rest of the paper, I will omit the parameter f when it is useless. The convention on the type of the letters will avoid possible confusions : for example, between $hnf(f, t)$ and $hnf(t, a)$. The latter is defined below and is thus the abbreviated form of $hnf(f, t, a)$ for a useless f.

Definition 2.7. Let a be an address, $t \in \Lambda'$ and f be a Böhm function.

1. "a is f-accessible in t" is defined by
   - nil is f-accessible in t.
   - $i + a$ is f-accessible in t iff $hnf(f, t) = \overrightarrow{\lambda}(x\ t_1 ...\ t_n), 1 \leq i \leq n$ and a is f -accessible in t_i.
2. Let a be f-accessible in t. Then, $hnf(f, t, a)$ is defined by
   - $hnf(f, t, nil) = hnf(f, t)$.
   - $hnf(f, t, i + a) = hnf(f, t_i, a)$ where $hnf(f, t) = \overrightarrow{\lambda}(x\ t_1 ...\ t_n)$.
3. Let a be f-accessible in t. Then, $adr(f, t, a)$ is defined by
   - $adr(f, t, nil) = t$.
   - $adr(f, t, i + a) = adr(f, t_i, a)$ where $hnf(f, t) = \overrightarrow{\lambda}(x\ t_1 ...\ t_n)$.

Definition 2.8. Let $u \in \Lambda$. $\psi(u)$ is the Böhm function f defined as follows

- $f(a)$ is defined iff a is accessible in u.
- $f(a) = (\{x_1,\ ...\ ,\ x_k\}, x, p)$ iff $hnf(u,a) = \lambda x_1\ ...\ \lambda x_k\ (x\ t_1\ ...\ t_p)$ for some terms $t_1,\ ..., t_p$
- $f(a) = \bot$ iff $hnf(u,a) = \bot$.

Comments and examples. In the following, assume $t \in \Lambda$ (and thus, I omit the parameter f).

- a is accessible in t iff $BT(t)$ has a node at the address a. For example, if t is unsolvable nil is the only accessible address in t .
- $hnf(t,a)$ is the λ-term we get at the address a when the computation of the node at this address in $BT(t)$ has *ended*.
- $adr(t,a)$ is the λ term we get at the *beginning* of the computation of the node at this address in $BT(t)$.
- Let $t = (I\ \ \lambda x\ (x\ (\delta\ \delta)))$. Then $adr(t,[1]) = (\delta\ \delta)$ and $hnf(t,[1]) = \bot$.
- Let Y be the Türing fixed-point operator, i.e. $Y = (B\ \ B)$ where $B = \lambda b \lambda x\ (x\ (b\ \ b\ \ x))$. Since $BT(Y) = \lambda x\ (x\ (x\ (x\ ...$ the addresses 1^n are the only accessible addresses in Y. For $n > 0$, $hnf(Y, 1^n) = (x\ (B\ \ B\ \ x))$ and $adr(Y, 1^n) = (B\ \ B\ \ x)$. Let $f = \psi(Y)$, then : $f(nil) = (\{x\}, x, 1)$ and, for $n \geq 1$, $f(1^n) = (\emptyset, x, 1)$.

Definition 2.9. Let f be a Böhm function. A term $t \in \Lambda'$ is f-correct if, for some context $C \in \Lambda$, $C(c(nil, \emptyset\ )) \twoheadrightarrow_f t$.

Definition 2.10. Let $u \in \Lambda$ and $t \in \Lambda'$. Assume t is $\psi(u)$-correct. $t\langle u\rangle$ is the λ-term defined by :

- If $t = x$ or $t = \bot$, then $t\langle u\rangle = t$
- If $t = \lambda x\ t_1$, then $t\langle u\rangle = \lambda x\ t_1\langle u\rangle$
- If $t = (t_1\ t_2)$, then $t\langle u\rangle = (t_1\langle u\rangle\ t_2\langle u\rangle)$
- If $t = c(a,\sigma)$, then $t\langle u\rangle = \tau(adr(u,a))$ where τ is defined by : $dom(\tau) = dom(\sigma)$ and $\tau(x) = \sigma(x)\langle u\rangle$.

$t\langle u\rangle$ is thus the term obtained by replacing, iteratively, in t the occurrences of $c(a,\sigma)$ by $\sigma(adr(u,a))$ for every a and σ. In particular, if $t = C(c(nil, \emptyset\ ))$ then $t\langle u\rangle = C(u)$.

Note that if t is any term in Λ', $t\langle u\rangle$ may be undefined because, either some address a, such that $c(a,\sigma)$ appears in t, is not accessible in u (and thus, $adr(u,a)$ is undefined), or because the process of replacing $c(a,\sigma)$ by $\sigma(adr(u,a))$ does not terminate. However, the following lemma shows that, if t is $\psi(u)$-correct, this does not occur.

Lemma 2.11. If t is $\psi(u)$-correct, then $t\langle u\rangle$ is defined.

Proof: By induction on the length of the reduction $C(c(nil, \emptyset\ )) \twoheadrightarrow_{\psi(u)} t$. Look at the various cases in definition 2.5 and remark that :

- $t\langle u\rangle$ is defined iff $t'\langle u\rangle$ is defined for every sub-term t' of t.
- $c(a,\sigma)\langle u\rangle$ is defined iff a is $\psi(u)$ -accessible and $\sigma(x)\langle u\rangle$ is defined for every variable x. □

Remark. $\psi(u)$ and $t\langle u\rangle$ are defined only for $u \in \Lambda$. Actually, there is one (and only one) result whose proof needs the definition of $\psi(u)$ and $t\langle u\rangle$ for $u \in \Lambda'$. It is proposition 4.12 that shows that usefulness is transitive. The definition is, of course, the same but this means that all the results concerning either $\psi(u)$ or $t\langle u\rangle$ should be given for this general case and then should be parametrized by a Böhm function, i.e. I should define $\psi(f,u)$ and $t\langle f,u\rangle$. Since these general results are proved exactly in the same way as the particular case, in order to simplify notations, I restricted myself to the case $u \in \Lambda$. These general results are stated in [5].

2.2. Some basic results on the f-reduction

In this section I prove some basic facts on the f-reduction. The main one (proposition 2.15) means that there is some *modularity* in the head reduction of a term t : Let C be some context and u be a λ-term. Let $t = C(u)$ and $t' = C(c(nil, \emptyset\ ))$. The head reduction of t is the same as the $\psi(u)$-head reduction of t' where, when $c(a,\sigma)$ appears in head position, the computation of the node at the address a in the Böhm tree of u "inserted".

Lemma 2.12. Let $v, v' \in \Lambda'$ and f be a Böhm function. Assume that $v \twoheadrightarrow_f v'$.

1. Let σ be a substitution. Then $\sigma(v) \twoheadrightarrow_f \sigma(v')$.
2. Let $\overrightarrow{r}$ be a sequence of terms and assume v' does not begin with λ. Then $(v\ \overrightarrow{r}) \twoheadrightarrow_f (v'\ \overrightarrow{r})$

   Moreover in both cases the length of the f-reduction remains the same.

Proof: Note that the more general case, where v' begins with λ, will be given in lemma 2.14. The proof is by induction on the length of the reduction and case analysis. Use the fact that, in general, $\sigma(u[x := v]) = \sigma(u)[x := \sigma(v)]$. □

Lemma 2.13. Let $t \in \Lambda'$ and f be a Böhm function such that t is f-unsolvable.

1. Let σ be a substitution. Then $\sigma(t)$ is f -unsolvable.
2. Let $\overrightarrow{r}$ be a sequence of terms. Then $(t\ \overrightarrow{r})$ is f-unsolvable.

Proof:

1. This follows immediately from lemma 2.12.
2. If t does not reduce to a term beginning with λ this follows immediately from lemma 2.12. Otherwise let $\overrightarrow{r}$ = $(r_1\ ...\ r_n)$ and t' be the least step where λ appears. Then (by lemma 2.12) $(t\ \overrightarrow{r})$ $\twoheadrightarrow_f$ $(t'\ \overrightarrow{r}) = (\lambda x\ t_1\ \overrightarrow{r}) \twoheadrightarrow_f (\sigma(t_1)\ r_2\ ...\ r_n)$ where $\sigma(x)$ = r_1. The result follows by lemma 2.12 and by repeating, if necessary, the same argument.

□

Lemma 2.14. Let v, r_1, ..., $r_p \in \Lambda'$, σ be a substitution and f be a Böhm function. Assume that $v \twoheadrightarrow_f \lambda x_1 ... \lambda x_k\ (w\ \overrightarrow{t'})$. Then $(\sigma(v)\ r_1\ ...\ r_p) \twoheadrightarrow_f \lambda x_{j+1} ... \lambda x_k\ (\sigma'(w)\ \overrightarrow{\sigma'(t)}\ r_{j+1}\ ...\ r_p)$ where $j = Min(k,p)$, $\sigma' = \tau \circ \sigma$ and τ is given by $\tau(x_i) = r_i$ for $1 \leq i \leq\ j$.

Proof: By induction on k. The case $k = 0$ is given by lemma 2.12. Assume $k \geq 1$. Look at the least step in the reduction $v \twoheadrightarrow_f v'$ where v' begins with λ, say $v' = \lambda x_1\ v_1$. Note that $x_1 \notin dom(\sigma)$ since x_1 is bounded in v. By lemma 2.12, $(\sigma(v)\ r_1\ ...\ r_p) \twoheadrightarrow_f (\sigma_1(v_1)\ r_2\ ...\ r_p)$ where $\sigma_1 = \tau \circ \sigma$ and τ is given by : $\tau(x_1) = r_1$. By the induction hypothesis, $(\sigma_1(v_1)\ r_2\ ...\ r_p) \twoheadrightarrow_f \lambda x_{j+1} ... \lambda x_k\ (\sigma'(w)\ \overrightarrow{\sigma'(t)}\ r_{j+1} ...\ r_p)$. □

Proposition 2.15. Let $t \in \Lambda'$ and $u \in \Lambda$. Assume $t = \overrightarrow{\lambda}\ (R\ r_1\ ...\ r_p)$ is $\psi(u)$-correct and t' is the $\psi(u)$-reduct of t. Then

1. if $R = x$, then $t\langle u\rangle$ is in head normal form.
2. if $R = (\lambda x\ v\ w)$ or $\perp$, then the head-reduct of $t\langle u\rangle$ is $t'\langle u\rangle$.
3. if $R = c(a,\sigma)$
   - If $f(a) = \perp$, then $t\langle u\rangle$ is unsolvable.
   - If $f(a) = (\{x_1,\ ...,\ x_k\}, x, q)$ then $t\langle u\rangle \twoheadrightarrow t'\langle u\rangle$.

Proof: (1) and (2) are clear. (3.1) follows from lemma 2.13 and (3.2) follows from lemma 2.14. □

3. The theorem of Hyland and Wadsworth

Definition 3.1. A binary relation $\sim$ on λ-terms is a bisimulation iff $t \sim t'$ implies :

- either t and t' both are unsolvable
- or $hnf(t) = \lambda x_1 ... \lambda x_n\ (x\ t_1 ...\ t_m)$, $hnf(t') = \lambda x_1 ... \lambda x_{n+p}\ (x\ t'_1 ...\ t'_{m+p})$ and,
  - for $1 \leq i \leq m$, $t_i \sim t'_i$
  - for $1 \leq j \leq p$, $x_{n+j} \sim t'_{m+j}$ and x_{n+j} is not free in t
- or $hnf(t') = \lambda x_1 ... \lambda x_n\ (x\ t'_1 ...\ t'_m)$, $hnf(t) = \lambda x_1 ... \lambda x_{n+p}\ (x\ t_1 ...\ t_{m+p})$ and,
  - for $1 \leq i \leq m$, $t_i \sim t'_i$
  - for $1 \leq j \leq p$, $x_{n+j} \sim t_{m+j}$ and x_{n+j} is not free in t'

Definition 3.2. Let f be a Böhm function. f_i is the Böhm function defined by : $f_i(a) = f(i+a)$.

Definition 3.3. A binary relation $\leq$ on Böhm functions is a simulation iff $f \leq f'$ implies :

- either $f(nil) = f'(nil) = \bot$
- or $f(nil) = (\{x_1,\ ...,\ x_n\},\ x,\ m)$, $f'(nil) = (\{x_1,\ ...,\ x_{n+p}\},\ x,\ m+p))$ and
  - for $1 \leq i \leq m$, $f_i \leq f'_i$
  - for $1 \leq j \leq p$, $\psi(x_{n+j}) \leq f'_{m+j}$ and $x_{n+j} \notin \{x_1,\ ...,\ x_n,\ x\}$

Proposition 3.4. (and definition)
There is a greatest bisimulation on λ-terms (resp. simulation on Böhm functions). It is denoted by $\sim$ (resp. by $\leq$).

Proof: Bisimulations (resp. simulations) are closed by union. □

Proposition 3.5. Let $u, u' \in \Lambda$ be such that $u \sim u'$. Then, there is a Böhm function f such that $\psi(u) \leq f$ and $\psi(u') \leq f$.

Proof: Immediate. □

Proposition 3.6. Let $t \in \Lambda'$ and $u \in \Lambda$. Assume t is $\psi(u)$-correct. Then, t is $\psi(u)$-solvable iff $t\langle u\rangle$ is solvable. Moreover $hnf(t\langle u\rangle) = hnf(\psi(u), t)\langle u\rangle$.

Proof: Immediate, by proposition 2.15. □

Theorem 3.7. Let u and u' be λ-terms such that $u \sim u'$. Then, for every context C, $C(u)$ is solvable iff $C(u')$ is solvable.

The idea of the proof is the following. Assume $u \sim u'$ and C is a context. Let $t = C(c(nil, \emptyset))$. Since $t\langle u\rangle = C(u)$, by proposition 3.6, $C(u)$ (resp. $C(u')$) is solvable iff t is $\psi(u)$-solvable (resp. $\psi(u')$ -solvable). By proposition 3.5, it is then enough to show : Let f, f' be Böhm functions such that $f \leq f'$. Then t is f-solvable iff t is f'-solvable.

This is proved by showing that the f-reduction and the f' -reduction of t are *essentially* the same. The intuitive meaning is the following. I introduce the calculus Λ" by adding to Λ' new constants denoted by $d(a, \sigma)$. The only difference between $d(a, \sigma)$ and $c(a, \sigma)$ is that the change of name marks the difference at the address a (due to η-expansions) between f and f'. The f'-reduction of t will be re-defined in such a way that, in definition 2.5, when there is a difference between f and f' at the address a, $c(a + i, \sigma)$ is replaced by $d(a + i, \sigma)$.

The precise relation between both reductions is given by proposition 3.15 : It intuitively says that, if $t \twoheadrightarrow_f t_1$, then $t \twoheadrightarrow_{f'} t'_1$ for some t'_1 such that t_1 is obtained from t'_1 by doing some η -reductions at the root of t'_1 and by erasing the differences between f and f'.

For the rest of this section, fix the Böhm functions f and f' such that $f \leq f'$.

Definition 3.8. Λ" is defined by the grammar :

$$\Lambda" = V \mid \bot \mid c(a, \sigma) \mid\ d(a, \sigma) \mid \lambda x\ \Lambda" \mid (\Lambda"\ \Lambda")$$

where $c(a, \sigma)$ and $d(a, \sigma)$ are constants as in definition 2.2.

Definition 3.9. Let $t \in \Lambda$". The f' reduction of t is defined as in definition 2.5 with the following changes:

1. the case 4.(b) (when $R = c(a,\sigma)$) is replaced by 4'.(b) :

   If $f'(a) = (\{x_1,\ ...,\ x_{k+l}\}, x, p+l)$ and $f(a) = (\{x_1,\ ...,\ x_k\}, x, p)$ then the f'-reduct of t is $\overrightarrow{\lambda}\lambda x_{j+1}\ ...\ \lambda x_{k+l}\ \ \sigma'(x)\ \overrightarrow{c}\ \overrightarrow{d}\ \ r_{j+1}\ ...\ r_q$ where $\overrightarrow{c} = c(a+1,\sigma')\ ...\ c(a+p,\sigma')$, $\overrightarrow{d} = d(a+(p+1),\sigma')\ ...\ d(a+(p+l),\sigma')$, $j = Min(k+l, q)$, $\sigma' = \tau \circ \sigma$ and τ is defined by $\tau(x_i) = r_i$ for $1 \leq i \leq j$.

2. the case 5. (when $R = d(a,\sigma)$) is added :

   (a) If $f'(a) = \perp$, then the f'-reduct of t is $\perp$.

   (b) If $f'(a) = (\{x_1,\ ...,\ x_k\}, x, k)$ then the f'-reduct of t is :
   $\overrightarrow{\lambda}\lambda x_{j+1}\ ...\ \lambda x_k\ \ \sigma'(x)\ d(a+1,\sigma')\ ...\ d(a+k,\sigma')\ r_{j+1}\ ...\ r_q$
   where $j = Min(k,q)$, $\sigma' = \tau \circ \sigma$ and τ is defined by $\tau(x_i) = r_i$ for $1 \leq i \leq j$.

   (c) If $f'(a)$ is undefined, the f'-reduct of t is undefined.

Remark. It is easily seen that this new definition consists only in changing some $c(a,\sigma)$ by $d(a,\sigma)$. In particular this does not affect the f'-solvability of a term.

Definition 3.10. Let $t \in \Lambda$".

1. t is correct if, for some context $C \in \Lambda$, $C(c(nil, \emptyset)) \twoheadrightarrow_{f'} t$.

2. Assume t is correct. $D(t)$ is the term in Λ' defined as follows :

   - $D(x) = x.\ \ D(\lambda x\ u) = \lambda x\ D(u).\ \ D((u\ v)) = (D(u)\ D(v))$.
   - $D(c(a,\sigma)) = c(a, D(\sigma))$ where $D(\sigma) = \tau$ is defined by $\tau(x) = D(\sigma(x))$.
   - $D(d(a,\sigma)) = D(\sigma(x))$ where x is the head variable of $f'(a)$, i.e. $f'(a) = (E, x, p)$.

The notion of correctness defined here is the (revised version) of f'-correctness of definition 2.9.

$D(t)$ is the term obtained by replacing, iteratively, in t the occurrences of $d(a,\sigma)$ by $\sigma(x)$ where x is the head variable of $f'(a)$. The following example gives the intuition :

Let $I = \lambda x\ x$ and $J = (Y\ G)$ where $G = \lambda j \lambda x \lambda x_1\ (x\ (j\ x_1))$. It is easily seen that $I \sim J$ and that, letting $f = \psi(I)$ and $f' = \psi(J)$, $f \leq f'$. $f(nil) = (\{x\}, x, 0)$, $f'(nil) = (\{x, x_1\}, x, 1)$ and $f'([1]) = (\{x_2\}, x_1, 1)$. Let $t = (c(nil, \emptyset)\ u\ v)$ where u, v are some λ-terms. Then $t \rightarrow_f (u\ v) = t_1$ and $t \rightarrow_{f'} (u\ d([1],\sigma)) = t_2$ where $\sigma(x_1) = v$ and thus $D(t_2) = t_1$. Note that $t\langle I\rangle \rightarrow (u\ v)$ and $t\langle J\rangle \rightarrow (u\ (J\ v))$

Note that if t is any term in Λ", $D(t)$ may be undefined because, either for some address a such that $d(a,\sigma)$ appears in t , $f'(a)$ is undefined, or because the process of replacing $d(a,\sigma)$ by $\sigma(x)$ does not terminate. However, the following lemma shows that, if t is correct, this does not occur.

Lemma 3.11. If t is correct, then $D(t)$ is well defined.

Proof: By induction on the length of the reduction $C(c(nil, \emptyset\ )) \twoheadrightarrow_{f'} t$. Look at the various cases in definition 3.9 and remark that :

- $D(t)$ is defined iff for every sub-term t' of t, $D(t')$ is defined,
- $D(c(a,\sigma))$ is defined iff $D(\sigma(x))$ is defined for $x \in dom(\sigma)$,
- $D(d(a,\sigma))$ is defined iff a is f'-accessible and $D(\sigma(x))$ is defined for the head variable of $f'(a)$.

□

Definition 3.12. Let $t \in \Lambda$" be correct and let $t' \in \Lambda'$. $t \lhd t'$ iff

- $t = \lambda x_1 ... x_n\ (R\ r_1\ ... r_p)$ and for some $k \geq 0$,
- $t' = \lambda x_1\ ... x_{n-k}\ (D(R)\ D(r_1) ...\ D(r_{p-k}))$ and, for $p-k < i \leq p$, $D(r_i) = x_i$.

$t \lhd t'$ means that t' is obtained from t by some η-reductions in head position and the D-operation. The following example gives the intuition :

Let $H = \lambda x \lambda y \lambda y'\ (x\ (J\ y)\ (J\ y'))$ and $h = \psi(H)$ where J is the term given in the example above. Clearly, $I \sim H$ and $f \leq h$. Let $t = (c(nil, \emptyset)\ u\ v)$ where u, v are some λ-terms. Then $t \rightarrow_f (u\ v) = t_1$ and $t \rightarrow_h \lambda y'(u\ d([1],\sigma)\ d([2],\sigma)) = t_2$ for some σ such that $D(t_2) = \lambda y'\ (u\ v\ y')$. Thus $t_2 \lhd t_1$.

Definition 3.13. Let $t \in \Lambda$" be correct. $n(t)$ is defined as follows. Assume $t = \overrightarrow{\lambda}\ (R\ \overrightarrow{r})$ as in definition 3.9. If $R = d(a,\sigma)$ then $n(t) = 1 + n(\sigma(x))$ where x is the head variable of $f'(a)$ and otherwise $n(t) = 0$.

Intuitively, $n(t)$ is the number of steps needed to get (by applying the function D) a head redex which is not $d(a,\sigma)$ (cf. definition 3.9 case 5).

Lemma 3.14. If t is correct, then $n(t)$ is well defined.

Proof: By induction on the length of the reduction $C(c(nil, \emptyset\ )) \twoheadrightarrow_{f'} t$. Use the fact that, if t is correct and $d(a,\sigma)$ appears in t, then $f'(a) = (\{x_1, ..., x_k\}, x, k)$ and $x \neq x_i$. □

Proposition 3.15.
1. Let $t = C(c(nil, \emptyset\ ))$ for some context $C \in \Lambda$. Assume $t \twoheadrightarrow_f t'$. Then $t \twoheadrightarrow_{f'} u'$ for some $u' \lhd t'$.
2. Let $u \in \Lambda$" be correct. Assume $u \lhd t$ and $t \rightarrow_f t_1$. Then $u \rightarrow^+_{f'} u_1$ for some $u_1 \lhd t_1$

Proof: (1) follows from (2) by an immediate induction on the length of the reduction $t \twoheadrightarrow_f t'$: use (2) with $t \lhd t$. Note that the converse (i.e. if $t \twoheadrightarrow_{f'} t'$, then $t \twoheadrightarrow_f u'$ for some $t' \lhd u'$) could be proved in a similar way but I don't need it.

(2) is proved by induction on $n(u)$. The proof, which is straightforward but rather tedious, is given in the appendix. □

The theorem 3.7 follows from proposition 3.15 in the following way :

- Assume t is f-solvable. Then $t \twoheadrightarrow_f t_1 = \overrightarrow{\lambda}(x\ \overrightarrow{r})$. By proposition 3.15 (1), $t \twoheadrightarrow_{f'} t'_1$ for some $t'_1 \lhd t_1$. It is clear that t'_1 is in head normal form.
- Assume t is not f-solvable. Then, the f-reduction of t is infinite. By proposition 3.15 (2), the f'-reduction of t also is infinite.

4. Proof of Kerth's conjecture

I first give some examples of decorations. Let $\delta = \lambda x\ (x\ x)$, $I = \lambda x\ x, B = \lambda b\ \lambda f\ (f\ (b\ b\ f))$ and $Y = (B\ B)$. Recall that Y is the Turing fixed point operator.

1. Let $t = (\delta\ \delta)$. Then the constant sequence (δ) is a decoration for t since t reduces by head reduction to $t' = (\delta\ \delta)$ and the first δ in t' is a descendent of the second δ in t.

2. Let $t = (B\ B\ I)$. Then the constant sequence (B) is a decoration for t since t reduces to itself (in 3 steps) and the first occurrence of B in this reduct is a descendent of the second occurrence of B in t.

3. Let $w_1 = \lambda xyz\ (z\ x\ y)$, $w_2 = \lambda xyz\ (y\ (x\ (z\ x)))$, $R = (w_1\ I\ w_2)$ and $w_3 = (w_2\ R)$. Then,

   - $t = (w_2\ R\ I\ w_2) \twoheadrightarrow (R\ w_3)$ (in 4 steps)
   - $(R\ w_3) \twoheadrightarrow (w_3\ I\ w_2) = t'$ (in 3 steps)
   - $(w_3\ I\ w_2) \twoheadrightarrow (w_2\ R\ I\ w_2) = t$ (in 7 steps)

   It is easy to check that w_2, w_3 and R are solvable and that the descendent condition is satisfied. Thus the sequence $[w_2, R,\ w_3,\ w_2,\ R,\ w_3,\ w_2, ...]$ is a decoration for t. Note that t' is equal to t but t is written as w_2 applied to 3 arguments whereas t' is written as w_3 applied to 2 arguments and thus the R in t' is not seen as an argument of the head term.

4. Other examples can be found in [8].

4.1. The idea of the proof

R. Kerth defines a decoration only for the head reduction of unsolvable terms, i.e. terms whose Böhm tree is $\perp$. I define below a decoration for the computation (by left reduction) of *any* branch of a term t. A branch in t is either an infinite branch of its Böhm tree or a finite one finishing with $\perp$, i.e. a branch in t which corresponds to an infinite computation. I prove a more general result (The computation of any branch in any λ term admits a decoration. cf. Theorem 4.6) but this general notion of decoration is necessary for the proof of even the restricted case. The idea of the proof is the following.

1) Let a be a branch of t and b be a branch of a sub-term u of t. I say that b is (t, a) useful if, intuitively (cf. definition 4.4) the computation of the branch a of t "uses" all the nodes of addresses $b \restriction i$ $(i < lg(b))$ of the Böhm tree of u. I first show that (cf. proposition 4.19) if a branch b of u is (t, a) useful and there is a decoration for (u, b), then there is a decoration for (t, a) . This is the reason for which it is necessary to extend the notion of decoration to solvable terms. The decoration of an unsolvable term t may "come from" a decoration of a solvable sub-term u of t.

2) Let $t = (u\ r_1\ ...\ r_n)$ and a be a branch in t. Say that a is created by the application of u to $r_1\ ...\ r_n$ if neither in u nor in any r_i there is a branch that is (t, a) useful. I also show (this is the key point of the proof. cf. proposition 4.20) that if the branch a in $t = (u\ r_1\ ...\ r_n)$ is created by the application of u to $r_1\ ...\ r_n$, then t reduces to some $t' = \overrightarrow{\lambda}(r_i\ s_1\ ...\ s_m)$ for some $s_1\ ...\ s_m$ and

- the occurrence of r_i in t' is a descendent of the one in t.

- the branch a in t' still is created by the application of r_i to $s_1 \ ... \ s_m$.

Actually, proposition 4.20 is a bit more complicated because we have to deal with possible substitutions of the free variables.

3) Theorem 4.6 is then proved by induction on the complexity of t. If t is in head normal form the result follows immediately from the induction hypothesis. Otherwise $t = \overrightarrow{\lambda}\ (u\ r_1\ ...\ r_p)$ for some $p \geq 1$. If the branch a is not created by the application of u to $r_1\ ...\ r_n$, i.e. either in u or in some r_i there is a branch that is (t, a) useful, the result follows from the induction hypothesis and the first point above. Otherwise, we get a decoration by using repeatedly the second point above.

4.2. Definitions

Definition 4.1. Let A be the set of finite or infinite list of positive integers.

Definition 4.2. Let $a \in A$, $t \in \Lambda'$ and f be a Böhm function.

1. a is an f-branch in t iff
   - $\forall i < lg(a)\ a \upharpoonright i$ is f-accessible in t.
   - if a is finite, then *hnf* $(f, t, a) = \ \perp$
2. Assume a is an f- branch in t and $k \in N$. $Res(f, t, a, k)$ and $Br(f, t, a, k)$ are defined by :
   - $Res(f, t, a, 0) = t$ and $Br(f, t, a, 0) = a$
   - If $Res(f, t, a, k)$ is not an f-head normal form then $Res(f, t, a, k + 1)$ = the f- reduct of $Res(f, t, a, k)$ and $Br(f, t, a, k + 1) = Br(f, t, a, k)$
   - If $Res(f, t, a, k) = \overrightarrow{\lambda}(x\ t_1 ...\ t_n)$ and $a = i\ +\ b$ then $Res(f, t, a, k + 1) = t_i$ and $Br(f, t, a, k + 1) = b$
   - Otherwise $Res(f, t, a, k)$ and $Br(f, t, a, k)$ are undefined.
3. $t \twoheadrightarrow_{f,a} t'$ means that $t' = Res(f, t, a, k)$ for some k.

Comments and examples. In the following, assume $t \in \Lambda$. Recall that, in this case, the parameter f is omitted since the f-reduction is the head-reduction.

1. $Res(t, a, k)$ is the term we get after k many steps in the computation of the branch a of $BT(t)$.
2. If $t' = Res(t, a, k)$ then $a' = Br(t, a, k)$ is the branch of t' that has to be computed to finish the computation of the branch a of t. Thus, if $t \twoheadrightarrow_a t'$ and $t' \twoheadrightarrow_{a'} t"$ then $t \twoheadrightarrow_a t"$.
3. If t is unsolvable, then nil is the only branch in t.
4. The only branch of Y is $1^\infty = [1,\ 1,\ ...]$.

5. Let $w = \lambda xyz\ (z\ (y\ (x\ x\ y))\ z)$ and $t = (w\ w)$.
   - $hnf\ (t, nil) = \lambda yz\ (z\ (y\ (w\ w\ y))\ z)$,
   - $hnf\ (t, [1]) = (y\ (w\ w\ y))$,
   - $hnf\ (t, [2]) = z$,
   - $hnf\ (t, [1, 1]) = \lambda z_1(\ z_1\ (y\ (w\ w\ y))\ z_1)$
   - a is accessible in t iff $a = 1^n$ or $a = 1^n + 2$. The only branch of t is 1^∞.

Definition 4.3. Let $t \in \Lambda'$.

1. The notion of sub-term of t is defined as usual, with the following additional rule : u is a (strict) sub-term of $c(a, \sigma)$ if u is a sub-term of $\sigma(x)$ for some x.

2. Let f be a Böhm function, b be f-accessible in t and $t \twoheadrightarrow_{f,b} t'$.

   - A sub-term u' of t' is a residue of a sub-term u of t if it is a "copy by β-reduction" of u where, possibly, the free variables have been substituted. u' is a descendent of u if it is a residue of u and the free variables have not been substituted.
   - The sub-term $u' = c(a', \sigma')$ of t' is an immediate successor of the sub-term $u = c(a, \sigma)$ of t if
     - $t \twoheadrightarrow_{f,b} t_1 = \overrightarrow{\lambda}(c(a, \tau)\ \overrightarrow{r}) \rightarrow_f t_2 = \overrightarrow{\lambda'}(\tau'(x)\ c(a + 1, \tau')\ ...\ c(a + p, \tau')\ \overrightarrow{r'}) \twoheadrightarrow_{f,b} t'$
     - u' is a residue of some element of the sequence $c(a + 1, \tau')\ ...\ c(a + p, \tau')$ in t_2
     - the occurrence of $c(a, \tau)$ in t_1 is a residue of u.

3. The successor relation (between terms as $c(a, \sigma)$) is the transitive closure of the immediate successor relation.

Remark. A more "formal" definition of these notions (that are intuitively very clear) is rather tedious. For more details see [9]. It is clear that the notion of descendent given above is exactly the one in [9] . In particular, if $t = (d\ \overrightarrow{u}) \twoheadrightarrow_a (d'\ \overrightarrow{u'})$ and d' is a residue of some element of the sequence $\overrightarrow{u}$ then it is also a descendent of this element.

Definition 4.4. Let $t, u \in \Lambda$ and assume that $t = D(\sigma(u))$ for some context D and some substitution σ. Let $t' = D(c(nil, \sigma))$ and $f = \psi(u)$. Let a be a branch in t.

1. Let b be an address accessible in u. b is (t, a) useful if, for some k, $\overrightarrow{v}$ and σ, $Res(f, t', a, k) = \overrightarrow{\lambda}\ (c(b, \sigma)\ \overrightarrow{v})$.

2. Let b be a branch in u. b is (t, a) useful if there is a sequence $< k_i, \sigma_i, \overrightarrow{v_i} >_{i<lg(b)}$ such that, for every i, $Res(f, t', a, k_i) = \overrightarrow{\lambda}\ (c(b \restriction i, \sigma_i)\ \overrightarrow{v_i})$. Moreover, the occurrence of $c(b \restriction i + 1, \sigma_{i+1})$ in $Res(f, t', a, k_{i+1})$ is an immediate successor of the occurrence of $c(b \restriction i, \sigma_i)$ in $Res(f, t', a, k_i)$.

Remarks and examples.

- A context is a λ term (not a λ' term !) with some holes. As usual, variables may be captured by a substitution in a context.
- It will be shown (see proposition 4.10) that, with the notations of the previous definition, a is an f-branch in t' and thus the definition makes sense.
- Most often, either σ is empty (i.e. u is a sub-term of t) or D is an applicative context (i.e. $t = (\sigma(u)\ \overrightarrow{r})$) but it is not always the case and I thus need this general definition. In fact, both cases are essentially the same since it is not difficult to prove the following fact.

  Let $t = D(u)$ for some context D and a be a branch in t. Assume that the address nil in u is (t, a) useful, then $t \twoheadrightarrow_a \overrightarrow{\lambda}(\sigma(u)\ \overrightarrow{r})$ for some σ whose domain consists in the free variables of u that are captured by the context D.
- Let $t = (Y\ I)$. t is unsolvable and thus nil is a branch in t. 1^∞ is a branch in Y. It is easy to check that 1^∞ is (t, nil) useful.
- Note that a term t may have many sub-terms each of them has a branch that is (t, a) useful. For example, let $t = (Y_1\ F)\ (Y_2\ F)$ where $Y_1 = Y_2 = Y$ and $F = \lambda f \lambda g\ (g\ f)$. The reduction given in the example after definition 2.6 shows that the branch 1^∞ in Y_1 is (t, nil) useful. Similarly, the branch 1^∞ in Y_2 is (t, nil) useful.
- Also note that, for an infinite branch b, being (t, a) useful is stronger that simply asking that for every $i, b \restriction i$ is (t, a) useful. Let $t = (Y_1\ H\ Y_2\ 0)$ where $Y_1 = Y_2 = Y$, $H = \lambda f n p\ (u\ n\ p\ (f\ n\ (s\ p)))$, $u = \lambda n p a\ (n\ F\ (p\ F\ \lambda x\ a))$, $F = \lambda x y\ (y\ x)$, $0 = \lambda x y\ y$ and $s = \lambda n f x\ (f\ (n\ f\ x))$. For every k, the address 1^k is (t, nil) useful both in Y_2 and Y_1. The branch 1^∞ of Y_1 is (t, nil) useful but the branch 1^∞ of Y_2 is not. The reason is the following. u is a term (given by Maurey) such that $(u\ n\ p\ a) \to a$ for every Church integers $n \geq p$. Since Y acts here as an infinite Church integer, $(u\ Y\ k\ a) \to a$ for every k and this computation "uses" the address 1^k of Y. It follows that, letting $G = (Y_1\ H)$, $t = (G\ Y_2\ 0) \to (G\ Y_2\ 1) \to (G\ Y_2\ 2) \to \ldots$. It is easy to see that, in this computation, the node at the address 1^{k+1} of Y_1 that is used for the reduction $(G\ Y_2\ k) \to (G\ Y_2\ k+1)$ satisfies the descendent condition whereas, since the occurrence of Y_2 in $(G\ Y_2\ k+1)$ is a "new" one, the node at the address 1^{k+1} of Y_2 that is used in this reduction does not satisfy the condition.

Definition 4.5. Let $t \in \Lambda$, a be a branch of t and (d_n) a sequence of λ terms. (d_n) is a decoration for (t, a) if there is a strictly increasing sequence (k_n) of integers and a sequence $(\overrightarrow{r_n})$ such that for every $n \geq 0$

1. $Res(t, a, k_n) = \overrightarrow{\lambda}(d_n\ \overrightarrow{r_n})$
2. d_{n+1} is the descendent of an element of $\overrightarrow{r_n}$

In [9], it is also asked that d_n is solvable. This is useless since it is a consequence of the other hypotheses : this follows immediately from lemma 4.9 below. Also note that d_n remains solvable under any substitutions : this follows immediately from proposition 4.18.

Theorem 4.6. Let $t \in \Lambda$ and a be a branch in t. Then (t, a) has a decoration.

Corollary 4.7. Every unsolvable λ term has a decoration in the sense of [9].

4.3. Some useful results

In this section, I prove some basic results on usefulness and, in particular, the fact that this notion is "transitive" (see proposition 4.12). I also prove some results dealing with the descendent relation. Some of them, are stronger versions of lemmas already proved in section 2.2.

Lemma 4.8. Let $t, t' \in \Lambda'$, f be a Böhm function and a be f-accessible in t. Assume $t \twoheadrightarrow_{f,a} t'$. Then, for some $a' \leq a, t \twoheadrightarrow_{f,a} adr(f, t, a') \twoheadrightarrow_f t'$.

Proof: Immediate from the definition. □

Lemma 4.9. Let $t \in \Lambda'$ and f be a Böhm function such that t is f-unsolvable. Let $\overrightarrow{r}$ be a sequence of terms. Then $(t\ \overrightarrow{r})$ is f-unsolvable. Moreover $(t\ \overrightarrow{r})$ has no reduct of the form $\overrightarrow{\lambda}\ (r_i\ \overrightarrow{v})$ where r_i is a descendent of an element of $\overrightarrow{r}$.

Proof: This is a stronger version of lemma 2.13. The new result is clear from the proof of the previous version. □

Proposition 4.10. Let $t \in \Lambda'$, $u \in \Lambda$, and $a \in A$. Assume that t is $\psi(u)$-correct. Then a is a $\psi(u)$-branch in t iff a is a branch in $t\langle u\rangle$.

Proof: It follows immediately from proposition 2.15 that t has an $\psi(u)$ -head normal form iff $t\langle u\rangle$ has an head normal form. Moreover if $hnf\ (f, t, nil) = \lambda x_1 ...\ \lambda x_k\ (x\ t_1 ...\ t_p)$ then $hnf\ (t\langle u\rangle, nil) = \lambda x_1 ...\ \lambda x_k\ (x\ t_1\langle u\rangle\ ...\ t_p\langle u\rangle)$. The result follows easily. □

Proposition 4.11. Let $t \in \Lambda'$, $u \in \Lambda$, $f = \psi(u)$ and a be f-accessible in t. Assume t is f-correct and $t \twoheadrightarrow_{f,a} t' = \overrightarrow{\lambda}(R\ \overrightarrow{s})$ and R is either x or $((\lambda x\ v)\ w)$ or $c(b, \sigma)$ and $f(b) \neq \perp$. Then, $t\langle u\rangle \twoheadrightarrow_a t'\langle u\rangle$. Moreover, let d' be a sub-term of t' that is a residue (resp. a descendent) of a sub-term d of t. Then $d'\langle u\rangle$ is a residue (resp. a descendent) of the corresponding sub-term $d\langle u\rangle$.

Proof: The proof is by induction on a. For $a = nil$, this is proposition 2.15 . If $a = i + b$, then $t \twoheadrightarrow_f \overrightarrow{\lambda}(x\ t_1\ ...\ t_n)$. By proposition 2.15, $t\langle u\rangle \twoheadrightarrow \overrightarrow{\lambda}(x\ t_1\langle u\rangle\ ...\ t_n\langle u\rangle)$ and the result follows from the induction hypothesis. The extra property (on the descendence relation) is easily checked from the proof of proposition 2.15. □

Proposition 4.12. Let t, u, v be in Λ, a (resp. b, c) be a branch in t (resp. in u, v). Assume that b is (t, a) useful and c is (u, b) useful. Then c is (t, a) useful.

Proof: Let $t = D(\sigma(u))$, $u = E(\tau(v))$. Let $t' = D(c(nil, \sigma)), u' = E(c(nil, \tau))$. Let $F = D(\sigma(E))$. Then $t = F(\sigma \circ \tau(v))$. Let $t" = F(c(nil, \sigma \circ \tau))$. I only prove $t" \twoheadrightarrow_{g,a} \overrightarrow{\lambda}(c(c \restriction j, \tau_j)\ \overrightarrow{r_j})$ for every $j < lg(c)$, where $g = \psi(v)$. I should prove a bit more, namely that the corresponding

$c(c \upharpoonright j, \tau_j)$ are in the immediate successor relation (see definition 4.4). This is rather tedious to write but this follows immediately from the proof.

Let $f = \psi(u)$ and $d = c \upharpoonright j$. Since c is (u, b) useful, $u' \twoheadrightarrow_{g,b} \overrightarrow{\lambda}(c(d, \tau')\ \overrightarrow{r'})$. Thus, by lemma 4.8, $u' \twoheadrightarrow_{g,b} adr(g, u', b') \twoheadrightarrow_g \overrightarrow{\lambda}(c(d, \tau')\ \overrightarrow{r'})$ for some $b' \leq b$. Since b is (t, a) useful, $t' \twoheadrightarrow_{f,a} \overrightarrow{\lambda}(c(b', \sigma')\ \overrightarrow{s'})$. Clearly $t" = t'\langle u' \rangle$: here is the point (cf. the remark at the end of section 2.1), where I use $t'\langle u' \rangle$ for $u' \in \Lambda'$). Thus, by proposition 4.11 and lemma 2.14 , $t" \twoheadrightarrow_{g,b} \overrightarrow{\lambda}(\sigma'(adr(u', b')\ \overrightarrow{s'}) \twoheadrightarrow_{g,b} \overrightarrow{\lambda}(c(d, \tau")\ \overrightarrow{r'})$. □

Definition 4.13. Let σ, σ' be substitutions. $\tau = \sigma \oplus \sigma'$ if

- $dom(\sigma) \cap dom(\sigma') = \emptyset$ and $dom(\tau) = dom(\sigma) \cup dom(\sigma')$
- $\tau(x) = \sigma(x)$ (resp. $\sigma'(x)$) if $x \in dom(\sigma)$ (resp. $x \in dom(\sigma')$)

Definition 4.14. Let u be in Λ. Define, for a accessible in u, $FV(u, a)$ by :

- $FV(u, nil) = \emptyset$
- $FV(u, a + i) = FV(u, a) \cup \{x_1 \ldots x_k\}$ where $hnf(u, a) = \lambda x_1 \ldots x_k\ (x\ \overrightarrow{r'})$

Lemma 4.15.

1. Let $t = (\sigma(u)\ \overrightarrow{r'}) \in \Lambda$, $t' = (c(nil, \sigma)\ \overrightarrow{r'})$, b be accessible in t, $f = \psi(u)$, $t' \twoheadrightarrow_{f,b} t"$ and $c(a, \tau)$ be a sub-term of $t"$. Then $\tau = \sigma \oplus \sigma'$ for some σ' such that $dom(\sigma') = FV(u, a)$. Moreover, for every $y \in dom(\tau)$, for every $a' > a$ and every $x \in FV(u, a') - FV(u, a)$, x is not free in $\tau(y)$.
2. Similarly for $t = D(\sigma(u))$ with $\tau = \sigma \oplus \sigma" \oplus \sigma'$ where $dom(\sigma")$ is the set of variables captured by the context D.
3. Moreover if $c(a', \tau')$ is a descendent of $c(a, \tau)$ then $\tau' = \tau \oplus \mu$ for some μ whose domain is $FV(u, a') - FV(u, a)$.

Proof: This comes immediately from the fact that we are doing head reduction (and of course the renaming rule to avoid capture). More precisely, this is proved by induction on the length of the reduction $t' \twoheadrightarrow_{f,b} t"$ by a simple case analysis. □

Lemma 4.16. Let $t = (\sigma(u)\ \overrightarrow{r'})$ be in Λ, b be a branch in t and $f = \psi(u)$. Let $t' = (c(nil, \sigma)\ \overrightarrow{r'})$.

1. Assume $t' \twoheadrightarrow_{f,b} \overrightarrow{\lambda}\ (c(a, \tau)\ \overrightarrow{s'})$ and $u \twoheadrightarrow_a adr(u, a) \twoheadrightarrow \lambda x_1 \ldots \lambda x_k\ (d\ \overrightarrow{v'}) \twoheadrightarrow \lambda x_1 \ldots \lambda x_k \ldots \lambda x_{k+k'}\ (d'\ \overrightarrow{v'})$ and d' is the descendent of an element of $\overrightarrow{v'}$.

   Then, there exist substitutions μ and μ' such that $t \twoheadrightarrow_b \overrightarrow{\lambda}(\mu(d)\ \mu(\overrightarrow{v'})\ \overrightarrow{w'}) \twoheadrightarrow_b \overrightarrow{\lambda'}\ (\mu'(d')\ \mu'(\overrightarrow{v'})\ \overrightarrow{w'})$ and $\mu'(d') = \mu(d')$ is a descendent of the corresponding element of $\mu(\overrightarrow{v'})$.
2. Similarly assume that :
   - $t' \twoheadrightarrow_{f,b} \overrightarrow{\lambda}\ (c(a, \tau)\ \overrightarrow{s'}) \twoheadrightarrow_{f,b} \overrightarrow{\lambda'}\ (c(a', \tau')\ \overrightarrow{s'})$ for some $a < a'$ and $c(a', \tau')$ is a successor of $c(a, \tau)$.

- $u \twoheadrightarrow_{a'} adr(u,a) \twoheadrightarrow \overrightarrow{\lambda}(d\ \overrightarrow{v}) \twoheadrightarrow_{a'} adr(u,a') \twoheadrightarrow \overrightarrow{\lambda}(d'\ \overrightarrow{v'})$ and d' is the descendent of an element of $\overrightarrow{v}$.

Then, there exist substitutions μ and μ' such that t $\twoheadrightarrow_b \overrightarrow{\lambda}(\mu(d)\mu(\overrightarrow{v})\overrightarrow{w}) \twoheadrightarrow_b \overrightarrow{\lambda}(\mu'(d')\mu'(\overrightarrow{v'})\ \overrightarrow{w'})$ and $\mu'(d') = \mu(d')$ is a descendent of the corresponding element of $\mu(\overrightarrow{v})$.

Proof:

1. By lemma 4.15, $\tau = \sigma \oplus \sigma_1$. By proposition 4.11, $t \twoheadrightarrow_b \overrightarrow{\lambda}(\tau(adr(u,a))\ \overrightarrow{s\langle u\rangle})$ and, by lemma 2.14, $\overrightarrow{\lambda}(\tau(adr(u,a))\ \overrightarrow{s\langle u\rangle}) \twoheadrightarrow \overrightarrow{\lambda}(\mu(d)\ \mu(\overrightarrow{v})\ \overrightarrow{w}) \twoheadrightarrow \overrightarrow{\lambda'}(\mu'(d')$
$\mu'(\overrightarrow{v'})\ \overrightarrow{w'})$ where $\mu = \sigma' \circ \tau$ (resp. $\mu' = \sigma" \circ \tau$) and the domain of σ' (resp. $\sigma"$) is $\{x_1 \ldots x_k\}$ (resp. $\{x_1 \ldots x_{k+k'}\}$). By lemma 4.15, $\mu = \tau \oplus \sigma'$ and $\mu' = \tau \oplus \sigma"$. Since d' is the descendent of an element of $\overrightarrow{v}$ the variables $x_{k+1} \ldots x_{k+k'}$ do not appear in d' and $\mu(d') = \mu'(d')$.

2. Similarly $t \twoheadrightarrow_b \overrightarrow{\lambda}(\mu(d)\ \mu(\overrightarrow{v})\ \overrightarrow{w}) \twoheadrightarrow_b \overrightarrow{\lambda}(\mu'(d')\ \mu'(\overrightarrow{v'})\ \overrightarrow{w'})$ where $\mu = \tau \oplus \sigma'$, $\mu' = \mu \oplus \sigma"$ and the domain of $\sigma"$ is $FV(u,a') - FV(u,a)$. Since d' is the descendent of an element of $\overrightarrow{v}$, d' has no free variables in $FV(u,a') - FV(u,a)$ and thus $\mu'(d') = \mu(d')$.

□

Proposition 4.17. Let $t = (\sigma(u)\ \overrightarrow{r})$ be in Λ and b be a branch in t. Let a be a branch in u that is (t,b) useful. Assume that $Res(u,a,k) = \overrightarrow{\lambda}(u_1\ \overrightarrow{v_1})$. Then,

- For some j and some τ, $Res(t,b,j) = \overrightarrow{\lambda}(\tau(u_1)\ \tau(\overrightarrow{v_1})\ \overrightarrow{w})$.
- Let c be a branch in u_1 that is $(Res(u,a,k), Br(u,a,k))$ useful. Then c is $(Res(t,b,j), Br(t,b,j))$ useful.

Proof: By lemma 4.8, $u \twoheadrightarrow_a adr(u,a_1) \twoheadrightarrow \overrightarrow{\lambda}(u_1\ \overrightarrow{v_1}) = u'$. Let $t' = (c(nil,\sigma),\ \overrightarrow{r'})$ and $f = \psi(u)$. Since a is (t,b) useful $t' \twoheadrightarrow_{f,b} \overrightarrow{\lambda}(c(a_1,\sigma_1)\ \overrightarrow{s'})$. Thus $t \twoheadrightarrow_b \overrightarrow{\lambda}(\sigma_1(adr(u,a_1))\ \overrightarrow{s'})$
$\twoheadrightarrow_b \overrightarrow{\lambda}(\tau(u_1)\ \tau(\overrightarrow{v_1})$
$\overrightarrow{w}) = Res(t,b,j) = t"$. Let $a' = Br(u,a,k)$ and $b" = Br(t,b,j)$. Since a is (t,b) useful, it is clear that a' is $(t",b")$ useful and since c is (u',a') useful, by proposition 4.12, c is $(t",b")$ useful. □

4.4. The key results

Propositions 4.19 and 4.20 give the key points mentioned in section 4.1. Intuitively proposition 4.20 gives the next step of the decoration and proposition 4.21 is the technical result that allows to iterate the construction.

Proposition 4.18. Let u be in Λ. Assume that u is unsolvable and (d_k) is a decoration for (u, nil).

1. Let σ be a substitution. Then, $(\sigma(d_k))$ is a decoration for $(\sigma(u), nil)$.
2. Let $t = (u\overrightarrow{r'})$. Then, there is a sequence (σ_k) of substitutions such that $(\sigma_k(d_k))$ is a decoration for (t, nil).

Proof:

1. This is trivial since, by lemma 2.12, if $u \twoheadrightarrow u'$ then $\sigma(u) \twoheadrightarrow \sigma(u')$.

2. Let p be the length of $\overrightarrow{r}$. If $p = 0$, this is trivial. Assume $p \geq 1$. If, for every k, $Res(u, nil, k)$ does not begin with λ the result follows from lemma 2.12. Otherwise, let k be the least integer such that $Res(u, nil, k) = \lambda x\ u'$. Since (d_n) is a decoration for (u, nil), let (k_n) be the sequence such that $Res(u, nil, k_n) = \overrightarrow{\lambda}(d_n\ \overrightarrow{v_n})$.

   Assume first that $k_0 > k$. Then (by lemma 2.12) $(u\overrightarrow{r}) \twoheadrightarrow (\lambda x\ u'\ \overrightarrow{r}) \twoheadrightarrow (\sigma(u')\ r_2\ ..\ r_p)$ where $\sigma(x) = r_1$. Repeating the same argument with $(\sigma(u')\ r_2\ r_p)$ yields the result.

   Assume that $k_0 \leq k$. Let n_0 be the largest integer such that $k_{n_0} \leq k$. Then (by lemma 2.12) for $n \leq n_0$: $Res(t, nil, k_n) = (d_n\ \overrightarrow{v}_n\ \overrightarrow{r})$. $Res(t, nil, k_{n_0}) \twoheadrightarrow (\lambda x\ u'\ \overrightarrow{r}) \twoheadrightarrow (\sigma(u')\ r_2\ ...\ r_p)$ where $\sigma(x) = r_1$. Since $(d_n)_{n>n_0}$ is a decoration for (u', nil), $(\sigma(d_n))_{n>n_0}$ is a decoration for $(\sigma(u'), nil)$. Since d_{n_0+1} is a descendent of an element of v_{n_0}, x is not free in d_{n_0+1}. Repeating the same argument with $((\sigma(u')\ r_2\ ...\ r_p)\ , nil)$ yields the result.

□

Proposition 4.19. Let t, u be in Λ and b (resp. a) be a branch in t (resp. u). Assume a is (t, b) useful and let (d_k) be a decoration for (u, a). Then there is a sequence (σ_k) of substitutions such that $(\sigma_k(d_k))$ is a decoration for (t, b).

Proof: - If a is infinite, the sequence (σ_k) is easily constructed by using lemma 4.16.

- If a is finite the sequence (σ_k) is easily constructed by using lemma 4.16 for the finite part of the branch and proposition 4.18 for its last node.

Proposition 4.20. Let $t = (u\ r_1 ...\ r_n)$ be a λ term and a be a branch in t. Assume there is no branch neither in u nor in any r_i that is (t, a) useful. Then there is $< i, k, u_1, \nu >$ such that, letting $t' = Res(t, a, k)$ and $a' = Br(t, a, k)$:

- $t' = \overrightarrow{\lambda}\ (\nu(u_1)\ \overrightarrow{v})$ for some $\overrightarrow{v}$,
- $u_1 = (r_i\ s_1\ ...\ s_m)$ and $\nu(r_i) = r_i$ is a descendent of its occurrence in t.
- For $1 \leq j \leq m$, s_j has no branch that is (t', a') useful
- u_1 has a branch that is (t', a') useful.

Comments. The intuition of the proof is the following : Since there is no useful branch in u the set of useful nodes in $BT(u)$ is (by König's lemma) finite. Assume, for example, that $t = (\lambda x \lambda y\ (x\ s_1\ s_2)\ r_1\ r_2)$. Then $t \twoheadrightarrow (r_1\ s'_1\ s'_2)$. If there is no useful branch neither in s'_1 nor in s'_2 we are done. Otherwise there is such a useful branch in, say, s'_1. Thus $t \twoheadrightarrow \overrightarrow{\lambda}(s'_1\ \overrightarrow{w})$ for some $\overrightarrow{w}$. By the previous lemmas, it is mainly enough to prove the result for s'_1. But $t' = (\lambda x \lambda y\ s_1\ r_1\ r_2) \twoheadrightarrow s'_1$ and the cardinality of the set of useful nodes of t' is smaller than the one of t. We get the result by repeating the previous argument.

Before giving the proof, I give an example of the difficult case (the case 2.b in the proof). This is the example 4.3.6 in [8]. Let $w = \lambda xyz\ (y\ (x\ (z\ x)))$, $R = \lambda z\ (z\ I\ w)$ and $t = (w\ R\ I\ w)$. t is unsolvable. w, R, I are normal and so they do not have a branch that is (t, nil) useful. $t \twoheadrightarrow (I\ (R\ (w\ R))) \twoheadrightarrow$

$(R\ (w\ R))$. We cannot choose the step $(I\ (R\ (w\ R)))$ and the argument I as the first element of the decoration for t since the unsolvability is already created (and "used") in $(R\ (w\ R))$. We will choose the next step $(R\ (w\ R))$ and the argument R because, at this step, the unsolvability is not yet created since R and $(w\ R)$ are solvable. Thus, here, the solution is : $k = 4, u_1 = (R\ (w\ R)), i = 1, \nu = \emptyset$ and $\overrightarrow{v}$ is empty.

Proof: Let E = {b / b is an address accessible in u, that is (t, a) useful}. Note that for b in E,$hnf(u, b) \neq \perp$ because otherwise b would be a branch in u that is (t, a) useful.

I define a procedure to construct the desired $< i, k, u_1, \nu >$ and a branch in u. This procedure halts (and I thus get the result) because otherwise this means we always are in the case (1) below and this procedure has constructed an infinite branch in u that is (t, a) useful and this is a contradiction. Note that I cannot use the fact that E is finite (and prove the result by induction on the cardinality of E). Intuitively this is actually the argument used but we cannot formalize it in this way. If E is infinite, by König's lemma, there is an infinite branch b such that for every i, $b \restriction i \in E$ but (see the example after definition 4.4) this does not imply that b is (t, a) useful.

nil clearly is in E. Let $hnf(u, nil) = \lambda x_1\ ...\ x_k\ (x\ w_1\ ...\ w_p)$, $j_0 = Min(k, n)$ and σ is given by $\sigma(x_j) = r_j$ for $j \leq j_0$. It is clear that $j_0 \geq 1$ because otherwise t reduces to $\overrightarrow{\lambda}(x\ \overrightarrow{w}\ \overrightarrow{r})$ and then u or some r_i would have a branch that is (t, a) useful.

1) Assume first that $x \notin \{x_1\ ...\ x_k\}$. Then $t \twoheadrightarrow \lambda x_{j_0+1}\ ...\ x_k\ (x\ \ \sigma(w_1)\ ...\ \sigma(w_p)\ r_{j_0+1}\ ...\ r_n)$ and thus $a \neq nil$. Let $a = i + l$. If $i > p$, there is a branch in r_i that is (t, a) useful and this contradicts the hypothesis. Thus $i \leq p$. Let $u' = \lambda x_1\ ...\ x_{j_0}\ w_i$. Then $t \twoheadrightarrow_a \sigma(w_i)$ and $(u'\ \ r_1\ ...\ r_n) \twoheadrightarrow \sigma(w_i)$. The first node of the branch constructed by the procedure is i. Repeat the procedure (to get the other nodes) with $(u'\ \ r_1\ ...\ r_n)$.

2) Assume that $x = x_i$. Then $t \twoheadrightarrow \lambda x_{j_0+1}\ ..\ x_k\ (r_i\ \ \sigma(w_1)\ ..\ \sigma(w_p)\ r_{j_0+1}..\ r_n)$.

a) Assume first that for $1 \leq q \leq p$, $\sigma(w_q)$ has no branch that is (t, a) useful. Then $< i, j_0, u_1, \emptyset >$ where $u_1 = (r_i\ \ \sigma(w_1)\ ...\ \sigma(w_p)\ r_{j_0}\ ...\ r_n)$ clearly satisfies the conclusion of the proposition.

b) Assume that, for some $1 \leq q \leq p$, $\sigma(w_q)$ has a branch that is (t, a) useful.

Claim : There is $b \in E$ and $j \leq j_0$ such that $hnf\ (u, b) = \overrightarrow{\lambda}(x_j\ s_1\ ...\ s_l)$ and $\sigma(hnf\ (u, b))$ has a branch that is (t, a) useful but no $\sigma(s_m)$ has such a branch.

Proof : Note that $adr(u, [q]) = w_q$. By the hypothesis, $[q]$ is in E. Let $hnf\ (u, [q]) = \overrightarrow{\lambda}(y\ s_1\ ...\ s_l)$. If $y = x_j$ and no $\sigma(s_m)$ has a branch that is (t, a) useful, $b = [q]$ satisfies the conclusion of the claim. Otherwise some $\sigma(s_m)$ has a branch that is (t, a) useful. (*Proof :* If $y = x_j$ this is clear. If $y \notin \{x_1\ ...\ x_k\}$, $\sigma(\ hnf\ (u, [q])) = \overrightarrow{\lambda}(y\ \sigma(s_1)\ ...\ \sigma(s_l))$ and this is again clear since a branch in $\sigma(hnf\ (u, [q]))$ is a branch in some $\sigma(s_m)$). We may repeat the argument with $b = q + m$. If the claim fails we get in this way an infinite branch in u that is (t, a) useful. (Q.E.D. of the claim)

Let (b, j) be given by the claim. Let $t' = (c(nil, \emptyset\ )\ r_1\ ...\ r_n)$ and $f = \psi(u)$. $t' \twoheadrightarrow_{f,a} \overrightarrow{\lambda}(c(b, \tau)\ \overrightarrow{w})$ for some $\tau = \sigma \oplus \sigma'$ and thus $t \twoheadrightarrow_a \overrightarrow{\lambda}(\tau(adr(u, b))\ \ \overrightarrow{w})$. By lemmas 2.14 and 4.15, there is a substitution τ' such that $\overrightarrow{\lambda}(\tau(adr(u, b))\ \ \overrightarrow{w}) \twoheadrightarrow \overrightarrow{\lambda}(\mu(x_j)\ \mu(\overrightarrow{s'})\ \ \overrightarrow{v'}) = Res(t, a, k)$ where $\mu = \tau \oplus \tau' = \sigma \oplus \sigma' \oplus \tau'$. Then, $< j, k, u_1, \sigma' \oplus \tau' >$ satisfies the conclusion of the proposition, where $u_1 = (r_j\ \overrightarrow{\sigma(s)}) = \sigma((x_j\ s_1\ ...\ \ s_l))$. □

Proposition 4.21. Let $(d_n)_{n\geq 0}$ (resp. $(\overrightarrow{u_n})_{n\geq 0}$, $(\overrightarrow{v_n})_{n\geq 1}$, resp. $(a_n)_{n\geq 0}$, resp.

$(\sigma_n)_{n\geq 1}$) be a sequence of λ terms (resp. be sequences of finite sequences of λ terms, resp. be a sequence of elements of A, resp. be a sequence of substitution). Assume that for every $n \geq 0$

- $t_n = (d_n\ \overrightarrow{u_n})$ and a_n *is* a branch in t_n.
- For some k_n, $Res(t_n, a_n, k_n) = \overrightarrow{\lambda_n}(\sigma_{n+1}(t_{n+1})\ \ \overrightarrow{v}_{n+1})$ and a_{n+1} is $(Res(t_n, a_n, k_n), Br(t_n, a_n, k_n))$ useful.
- d_{n+1} is the descendent of an element of the sequence $\overrightarrow{u_n}$
- $\sigma_{n+1}(d_{n+1}) = d_{n+1}$.

  Then, there is an increasing sequence (τ_n) of substitutions such that the sequence $(\tau_n(d_n))$ is a decoration for $(t_0,\ a_0)$.

Proof: I construct (by induction on n) a sequence $< j_n, r_n, b_n, \tau_n >$ such that: $r_0 = t_0, j_0 = 0, \tau_0 = \emptyset$, $b_0 = a_0$ and, for $n \geq 1$, $r_n = Res(r_0, b_0, j_n) = \overrightarrow{\lambda}(\tau_n(t_n)\ \overrightarrow{w_n}), b_n = Br(r_0, b_0, j_n), \tau_n(d_n) = \tau_{n-1}(d_n)$ and a_n is (r_n, b_n) useful. It is clear that the sequence (τ_n) satisfies the conclusion.
$t_n \twoheadrightarrow_{a_n} \overrightarrow{\lambda_n}(\sigma_{n+1}(t_{n+1})\ \overrightarrow{v_{n+1}})$. Since a_n is (r_n, b_n) useful and by proposition 4.17, $r_n \twoheadrightarrow_{b_n} r'_n = \overrightarrow{\lambda}(\overrightarrow{\lambda}_n\ (\tau_n(\sigma_{n+1}(t_{n+1}))\ \tau_n(\overrightarrow{v_{n+1}}))\ \overrightarrow{w_n})$ for some τ_n and $\overrightarrow{w_n}$.

Clearly $r'_n \twoheadrightarrow \overrightarrow{\lambda}\ (\tau_{n+1}(t_{n+1})\ \overrightarrow{w_{n+1}}) = Res(r_0, a_0, j_{n+1})$ for some $\overrightarrow{w_{n+1}}$ where $\tau_{n+1} = \tau_n \circ \sigma_{n+1} \oplus \mu_n$ and the domain of μ_n is included in the variables in $\overrightarrow{\lambda}_n$. Since d_{n+1} is the descendent of an element of $\overrightarrow{u_n}$, d_{n+1} is not affected by μ_n . Since, by the hypothesis, $\sigma_{n+1}(d_{n+1}) = d_{n+1,}$ we have $\tau_{n+1}(d_{n+1}) = \tau_n(d_{n+1})$. Finally, again by proposition 4.17, a_{n+1} is $(r_{n+1},\ b_{n+1})$ useful.

4.5. End of the proof of the theorem

Let t be a λ term and a be branch in t. The existence of a decoration is proved by induction on the complexity of t.

- If $t = \lambda x\ u$ or $t = (x\overrightarrow{r})$ the result follows immediately from the induction hypothesis.
- If $t = (u\ r_1 ... \ r_n)$ and there is, either in u or in some r_i, a branch that is (t, a) useful. For example, say b is such a branch in u. By the induction hypothesis there is a decoration of (u, b) and by proposition 4.19 there is a decoration for (t, a).
- Otherwise $t = (u\ r_1 ... \ r_n)$ and there is no branch neither in u nor in any r_i that is (t, a) useful. Let $a_0 = a, d_0 = u$, $\overrightarrow{u_0} = r_1\ ...\ r_n, t_0 = (d_0\ \overrightarrow{u_0})$ and $\overrightarrow{v_0}$ be the empty sequence. By proposition 4.20 there is $< i, k_0, t_1, \sigma >$ such that, letting $t' = Res(t_0, a_0, k_0)$ and $a' = Br(t_0, a_0, k_0)$:
  - $t' = \overrightarrow{\lambda}\ (\sigma(t_1)\ \ \overrightarrow{v_1}), t_1 = (r_i\ s_1\ ...\ s_m), \sigma(r_i) = r_i$ for some terms $s_1\ ...\ s_m\ \overrightarrow{v_1}$ and some substitution σ.
  - For $1 \leq j \leq m$, s_j has no branch that is (t', a') useful
  - t_1 has a branch a_1 that is (t', a') useful.

Let $d_1 = r_i$ and $\overrightarrow{u_1} = s_1\ ...\ s_m$. No s_j has a branch that is (t_1, a_1) useful since, otherwise, by proposition 4.12 such a branch would be (t', a') useful. We may again use proposition 4.20 with t_1 and the branch a_1. By repeating the same argument we get sequences satisfying the hypothesis of proposition 4.21 and thus a decoration for t. □

References

[1] R. Amadio & P.L. Curien, *Domains and Lambda-Calculi*, Cambridge University Press, 1998.

[2] H.P. Barendregt, *The Lambda-Calculus, its syntax and semantics*, North-Holland (1984).

[3] A. Berarducci, *Infinite λ-calculus and non-sensible models*, in Logic and Algebra, ed. A. Ursini & P. Agliano, Lecture Notes in Pure and Applied Mathematics 180, Marcel Dekker Inc., 1996.

[4] M. Coppo & M. Dezani-Ciancaglini & S. Ronchi Della Rocca, *Semi-separability of finite sets of terms in Scott's D_∞-models of λ-calculus, in* Lecture Notes in Computer Science 62, p 142-164 (1978).

[5] R. David, *Every unsolvable λ term has a decoration*, Typed Lambda Calculi and Applications (TLCA'99), Lecture Notes in Computer Science 1581, p 98-113.

[6] R. David & K. Nour, *A syntactical proof of the operational equivalence of two λ-terms*, Theoritical Computer Science 180 (1997) p 371-375.

[7] R. David & K. Nour, *Storage operators and directed lambda-calculus*, Journal of Symbolic Logic 60 (4) p 1054-1086 (1995).

[8] R. Kerth *Isomorphisme et équivalence équationnelle entre modèles du λ calcul* Ph.D. thesis Université Paris 7, 1995.

[9] R. Kerth *The interpretation of Unsolvable λ Terms in Models of Untyped λ Calculus* . To appear in the JSL

[10] R. Kerth *On the Construction of Stable Models of Untyped λ Calculus* . To appear in TCS

[11] M. Hyland, *A syntactic characterization of the equality in some models for the Lambda-Calculus*, Journal of the London Mathematical Society 12 (2), p 361-370, 1976.

[12] J.L. Krivine, *Lambda-Calcul, Types et modèles*, Masson, Paris (1990).

[13] C.P. Wadsworth, *The relation between computational and denotational properties for Scott's D_∞-models of λ-calculus*, SIAM Journal of Computing 5 (3), p 488-521, 1976.

5. Appendix : Proof of proposition 3.15

Lemma 5.1. $n((u\ \overrightarrow{v})) \leq n(u)$.

Proof: Assume $\overrightarrow{v}$ has at least one element (otherwise the result is trivial). If u begins with λ or if $u = (R\ \overrightarrow{r})$ and $R \neq d(a,\sigma)$, then $n((u\ \overrightarrow{v})) = 0$ and the result is clear. Otherwise let $u = (d(a,\sigma)\ \overrightarrow{r'})$ and $f'(a) = (E,x,k)$. Then $n((u\ \overrightarrow{v})) = 1 + n(\sigma(x)) = n(u)$. □

The proof of proposition 3.15 (2) is by induction on $n(u)$. Let $u = \overrightarrow{\lambda'}\ (R'\ \overrightarrow{r'})$.

- If $R' = \bot$. Then $t = \overrightarrow{\lambda}\ (\bot\ \overrightarrow{r'})$ where $\overrightarrow{\lambda}$ is an initial segment of $\overrightarrow{\lambda'}$ and $\overrightarrow{r'}$ is an initial segment of $D(\overrightarrow{r'})$. The result is thus clear.
- If $R' = x$. Similarly $t = \overrightarrow{\lambda}\ (x\ \overrightarrow{r'})$ and again the result is clear.
- If $R' = (\lambda x\ w'\ v')$. Similarly $t = \overrightarrow{\lambda}\ ((\lambda x\ w\ v)\ \overrightarrow{r'})$ where $D(w') = w$ and $D(v') = v$. $u \to_{f'} \overrightarrow{\lambda}\ (w'[x := v']\ \overrightarrow{r'})$ and $t \to_f \overrightarrow{\lambda}\ (w[x := v]\ \overrightarrow{r'})$. The result follows from the (immediate) fact that $D(w'[x := v']) = D(w')[x := D(v')]$.

- If $R' = c(a,\tau)$. Then $t = \overrightarrow{\lambda}\ (c(a,\sigma)\ \overrightarrow{r})$ where $D(\tau) = \sigma$. Let k (resp. k') be the number of abstractions in $f(a)$ (resp. $f'(a)$). Let p (resp. p') be the length of $\overrightarrow{r}$ (resp. $\overrightarrow{r'}$). The proof depends on the relations on these numbers :

  1. $k \geq p$ and $k' \geq p'$.
  2. $k \geq p$ and $k' \leq p'$.
  3. $k \leq p$ and $k' \geq p'$.
  4. $k \leq p$, $k' \geq p$ and $k' \leq p'$.
  5. $k \leq p$ and $k' \leq p$.

  Note that, since $k \leq k'$ and $p \leq p'$, these are the only possible cases. In order to simplify notations I will only give examples. The reader will easily be convinced that the given examples are generic.

  In the examples, α will denote $\sigma' \circ \sigma$ and ν will denote $\tau' \circ \tau$ (σ' and τ' are defined in each example). In the examples 1, 2, 4, 5 : $f(a) = (\{x_1, x_2\}, z, 1)$ and $f'(a) = (\{x_1, x_2, x_3\}, z, 2)$. In the example 3 : $f(a) = (\{x_1, x_2\}, z, 1)$ and $f'(a) = (\{x_1, x_2, x_3, x_4, x_5\}, z, 4)$. When the context is clear, I will omit the letter a and write $c(\sigma)$ instead of $c(a,\sigma)$, $c(i,\alpha)$ instead of $c(a+i,\alpha)$, etc.

  1. $t = (c(\sigma)\ r)$ and $u = \lambda y_1\ (c(\tau)\ s\ v)$. Then $t \to_f \lambda x_2\ (\alpha(z)\ c(1,\alpha))$ and $u \to_{f'} \lambda y_1 x_3 (\nu(z)\ c(1,\nu)\ d(2,\nu))$ where $\sigma'(x_1) = r$ and $\tau'(x_1) = s$, $\tau'(x_2) = v$.
  2. $t = (c(\sigma)\ r)$ and $u = \lambda y_1 y_2\ y_3\ (c(\tau)\ s\ u_1\ u_2\ u_3)$. Then $t \to_f \lambda x_2\ (\alpha(z)\ c(1,\alpha))$ and $u \to_{f'} \lambda y_1 y_2 y_3\ (\nu(z)\ c(1,\nu)\ d(2,\nu)\ u_3)$ where $\sigma'(x_1) = r$ and $\tau'(x_1) = s$, $\tau'(x_2) = u_1, \tau'(x_3) = u_2$.
  3. $t = (c(\sigma)\ r_1\ r_2\ r_3)$ and $u = \lambda y_1\ (c(\tau)\ s_1\ s_2\ s_3\ v)$. Then $t \to_f\ (\alpha(z)\ c(1,\alpha)\ r_3)$ and $u \to_{f'} \lambda y_1 x_5\ (\ (\nu(z)\ c(1,\nu)\ d(2,\nu)\ d(3,\nu)\ d(4,\nu))$ where $\sigma'(x_1) = r_1, \sigma'(x_2) = r_2$ and $\tau'(x_1) = s_1$, $\tau'(x_2) = s_2, \tau'(x_3) = s_3, \tau'(x_4) = v$.
  4. $t = (c(\sigma)\ r_1\ r_2\ r_3)$ and $u = \lambda y_1\ (c(\tau)\ s_1\ s_2\ s_3\ v)$. Then $t \to_f\ (\alpha(z)\ c(1,\alpha)\ r_3)$ and $u \to_{f'} \lambda y_1\ (\nu(z)\ c(1,\nu)\ d(2,\nu)\ v)$ where $\sigma'(x_1) = r_1, \sigma'(x_2) = r_2$ and $\tau'(x_1) = s_1$, $\tau'(x_2) = s_2, \tau'(x_3) = s_3$.
  5. $t = (c(\sigma)\ r_1\ r_2\ r_3\ r_4)$ and $u = \lambda y_1\ (c(\tau)\ s_1\ s_2\ s_3\ s_4\ v)$. Then $t \to_f\ (\alpha(z)\ c(1,\alpha)\ r_3\ r_4)$ and $u \to_{f'} \lambda y_1\ (\nu(z)\ c(1,\nu)\ d(2,\nu)\ s_4\ v)$ where $\sigma'(x_1) = r_1, \sigma'(x_2) = r_2$ and $\tau'(x_1) = s_1$, $\tau'(x_2) = s_2, \tau'(x_3) = s_3$.

  I must check that the $\lhd$ relation is satisfied. I will only give the proof for the case 1. and 3. The other are similar.

  Example 1.

  - $D(\nu) = \alpha$ (with the α-conversion between x_2 and y_1) : Let x be a variable. If $x \neq x_i$ (for $i = 1, 2$) then $\alpha(x) = \sigma(x)$, $\nu(x) = \tau(x)$ and the result follows from the fact that $D(\tau) = \sigma$. Note that x_2 (resp. y_1) is not free in $\sigma(x)$ (resp. $\tau(x)$). If $x = x_1$, then $\alpha(x) = r$ and $\nu(x) = s$ and the result follows from the fact that $D(s) = r$. If $x = x_2$ then $\alpha(x) = x_2$ and $\nu(x) = v$ and the result follows from the fact that $D(v) = y_1$.

– $D(d(2,\nu)) = x_3$: Immediate.

Example 3.

– $D(\nu) = \alpha$: As in example 1.
– $D(d(2,\nu)) = r_3$: The head variable of $f'(a+2)$ is x_3, $\nu(x_3) = s_3$ and $D(s_3) = r_3$.
– $D(d(3,\nu)) = y_1$. The head variable of $f'(a+3)$ is x_4, $\nu(x_4) = v$ and $D(v) = y_1$.
– $D(d(4,\nu)) = x_5$. The head variable of $f'(a+4)$ is x_5, $\nu(x_5) = x_5$.

- If $R' = d(a,\tau)$. Let $f'(a) = (\{x_1,\ ...,\ x_n\}, x, k)$. Then $t = \overrightarrow{\lambda}\ (w\ \overrightarrow{r'})$ where $w = D(\tau(x))$. Let p (resp. p') be the length of $\overrightarrow{r}$ (resp. $\overrightarrow{r'}$). The proof again depends on the relations on these numbers :

1. $k \leq p$
2. $k \geq p$ and $k \leq p'$
3. $k \geq p'$

In order to simplify the notations, I will again give only generic examples and omit the address a. Let $\alpha = \tau' \circ \tau$ (where τ' is defined in each example)

1. $f'(a) = (\{x_1\}, x, 1), t = (w\ r_1\ r_2)$ and $u = \lambda y\ (d(\tau)\ s_1\ s_2\ v)$. Then $u \rightarrow_{f'} u' = \lambda y\ (\tau(x)\ d(1,\alpha)\ s_2\ v)$ where $\tau'(x_1) = s_1$.
2. $f'(a) = (\{x_1, x_2\}, x, 2), t = (w\ r)$ and $u = \lambda y_1 y_2\ (d(\tau)\ s\ u_1\ u_2)$. Then $u \rightarrow_{f'} u' = \lambda y_1 y_2\ (\tau(x)\ d(1,\alpha)\ d(2,\alpha)\ u_2)$ where $\tau'(x_1) = s$ and $\tau'(x_2) = u_1$.
3. $f'(a) = (\{x_1, x_2, x_3\}, x, 3), t = (w\ r)$ and $u = \lambda y_1\ (d(\tau)\ s\ v)$. Then $u \rightarrow_{f'} u' = \lambda y_1 \lambda x_3 (\tau(x)\ d(1,\alpha)\ d(2,\alpha)\ d(3,\alpha))$ where $\tau'(x_1) = s$ and $\tau'(x_2) = v$.

It is enough to show that $u' \lhd t$ and $n(u') < n(u)$ because the result follows then from the induction hypothesis. The second fact follows immediately from the definition of $n(t)$ and lemma 5.1. I will check the $\lhd$ relation only for the example 3. The other are similar.

– $D(\tau(x)) = w : w = D(d(\tau) = D(\tau(x))$.
– $D(d(1,\alpha)) = r$: The head variable of $f'(a+1)$ is x_1, $\alpha(x_1) = s$ and $D(s) = r$.
– $D(d(2,\alpha)) = y_1$: The head variable of $f'(a+2)$ is x_2, $\alpha(x_2) = v$ and $D(v) = y_1$.
– $D(d(3,\alpha)) = x_3$: The head variable of $f'(a+3)$ is x_3, $\alpha(x_3) = x_3$.

□