Reversed PAM and pointer structures

Pierre Clairambault

September 1, 2008

Abstract

We recall the definitions for pointer structures. We then prove define a simple simply-typed lambda calculus and prove the equivalence of the normalization of its linear head reduction and the finiteness of pointer structures subject to natural conditions. To achieve this, we define a reversed version of the Pointer Abstract Machine, which browses a pointer structure, recovering dynamically two lambda terms whose set of possible interactions include the current pointer structure.

Contents

1 Pointer structures

In what follows, $\equiv$ will denote the even/odd equivalence on integers.

Definition 1. *A **pointer structure** is a function $f : \mathbb{N} \to \mathbb{N} \cup \{\bot\}$ such that :*

- $\forall i \in \mathbb{N},\ f(i) = \bot \Rightarrow \forall j > i,\ f(j) = \bot$
- $\forall i \in \mathbb{N}^*,\ f(i) \neq \bot \Rightarrow f(i) < i$
- $\forall i \in \mathbb{N}^*,\ i \not\equiv f(i)$
- $f(0) = 0$

Definition 2. *A pointer structure is **finite** if $\exists i \in \mathbb{N},\ f(i) = \bot$. Otherwise, it is **infinite**.*

Definition 3. *Let f be a pointer structure. We define its **domain** $\mathcal{D}_f$ as:*

- *$\mathbb{N}$ if f is infinite*
- *$\{i \in \mathbb{N} \mid f(i) \neq \bot\}$ otherwise.*

Definition 4. *If f is a pointer structure, we define its **P-view function** $\ulcorner f \urcorner : \mathcal{D}_f \to \mathcal{P}(\mathbb{N})$ by :*

- $\ulcorner f \urcorner(0) = \{0\}$
- *If $i \equiv 1$, $\ulcorner f \urcorner(i) = \{i\} \cup \ulcorner f \urcorner(i-1)$*
- *Otherwise $\ulcorner f \urcorner(i) = \{i\} \cup \ulcorner f \urcorner(f(i))$*

*Similarly, we define its **O-view function** by :*

- $\llcorner f \lrcorner(0) = \{0\}$
- *If $i \equiv 1$, $\llcorner f \lrcorner(i) = \{i\} \cup \llcorner f \lrcorner(f(i))$*
- *Otherwise, $\llcorner f \lrcorner(i) = \{i\} \cup \llcorner f \lrcorner(i-1)$*

*Finally, we define its **view function** by:*

- $\ulcorner f \lrcorner(x) = \ulcorner f \urcorner(x)$ *if x is odd,*
- $\ulcorner f \lrcorner(x) = \llcorner f \lrcorner(x)$ *if x is even.*

Alternatively, there is a nice mutually recursive definition of the view function:

Definition 5. *Let f be a pointer structure, we define simultaneously the **view** $\ulcorner f \lrcorner$ and the **coview** $\llcorner f \urcorner$ the following way:*

- $\ulcorner f \lrcorner(0) = \llcorner f \urcorner(0) = \{0\}$
- $\ulcorner f \lrcorner(i) = \{i\} \cup \llcorner f \urcorner(n-1)$
- $\llcorner f \urcorner(i) = \{i\} \cup \ulcorner f \lrcorner(f(i))$

Definition 6. *Let f be a pointer structure. We say that x **sees** y in f if $x \in \mathcal{D}_f$ and $y \in \ulcorner f \lrcorner(x)$.*

Definition 7. *A pointer structure f is **visible** if $\forall x \in \mathcal{D}_f$, x sees $f(x)$.*

Definition 8. *Let f be a pointer structure. A **n-fork** on f is a pair $(\psi_0, (\psi_i)_{i \in \{1 \dots n\}}) \in \mathcal{D}_f \times \mathcal{D}_f^n$ such that :*

- $\forall i, j \in \{1 \dots n\},\ f(\psi_i) = f(\psi_j) = \psi_0$
- $\forall i \neq j \in \{1 \dots n\},\ \psi_i \neq \psi_j$

Definition 9. *Let f be a pointer structure, let $(\psi_0, (\psi_i))$ be a n-fork on f. $(\psi_0, (\psi_i))$ is said to be **conscious** if $\forall i \in \mathcal{D}_f,\ \forall j \in \{0 \dots i\},\ \psi_i$ sees ψ_j.*

Definition 10. *Let f be a pointer structure. We define the **depth function** of f, denoted d_f by:*

- $d_f(0) = 0$
- $d_f(n) = 1 + d_f(f(n))$

*The **depth** of f is*

$$depth(f) = \sup_{i \in \mathcal{D}_f} d_f(i)$$

*Note that the depth may be infinite if d_f is not bounded. We say that a pointer structure is **bounded** if its depth function is bounded.*

Definition 11. *Let f be a pointer structure. We define the **memory** of f as the largest $k \in \overline{\mathbb{N}}$ such that f admits a conscious k-fork. Let $mem(f)$ denote the memory of f.*

Theorem 1. *Let f be a visible pointer structure. Then the two following proposition are equivalent:*

(i) *d_f is bounded and the memory of f is finite,*

(ii) *f is finite.*

Another formulation of the same result (in fact slightly stronger) :

Theorem 2. *Let f be a bounded visible pointer structure. Then if it has an ω-fork, it has a infinite ω-fork.*

2 The $\Lambda_{\mathcal{U}}$-calculus

2.1 Definition

Syntax.

- $T ::= \lambda x.T \mid x \mid TT \mid T + T \mid \maltese$

Reduction. Reduction is β-reduction, with a weak head strategy, with an additionnal non-deterministic reduction rule, and a rule for the daimon.

$$(\lambda x.M)N \to M[N/x]$$

$$M + N \to M$$

$$M + N \to N$$

$$\maltese N \to \maltese$$

At each step, we only consider the unique redex which is in head position.

Typing. The types are natural numbers, with the following inference rules :

Unary Λ

$$\frac{}{\Gamma, x : k \vdash x : k}\, ax \qquad \frac{}{\Gamma \vdash \maltese : k}\, dai$$

$$\frac{\Gamma, x : k \vdash M : 0}{\Gamma \vdash \lambda x.M : k+1}\, lam \qquad \frac{\Gamma \vdash M : k+1 \quad \Gamma \vdash N : k}{\Gamma \vdash MN : 0}\, app$$

$$\frac{\Gamma \vdash M : k \quad \Gamma \vdash N : k}{\Gamma \vdash M + N : k}\, sum$$

Subtyping.[1] We define an order $\preccurlyeq$ on integer the following way :

- $\forall n, \quad 0 \preccurlyeq n$
- $\forall n, \quad n + 1 \not\preccurlyeq 0$
- $\forall n, p, \quad n + 1 \preccurlyeq p + 1 \Leftrightarrow p \preccurlyeq n$

Proposition 1. *$\preccurlyeq$ is an order on $\mathbb{N}$.*

Proof. By induction. □

Subtyping

$$\frac{\Gamma \vdash M : k \quad k \preccurlyeq p}{\Gamma \vdash M : p}\, sub$$

Proposition 2 (Subject reduction). *If $\Gamma \vdash M : k$ and $M \to M'$, then $\Gamma \vdash M' : k$*

[1] The final proof does not use subtyping anymore. However I find this definition quite nice and natural, thus I leave it here for the moment...

Proof. We extend the rewriting rules to a cut elimination procedure on the proof trees. There are three cases to consider (since there are three types of redexes). First β-reduction :

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, x : k \vdash M : 0}}{\Gamma \vdash \lambda x.M : k+1}\, lam \qquad \dfrac{\pi_2}{\Gamma \vdash N : k}}{\Gamma \vdash (\lambda x.M)N : 0}\, app \quad \rightarrow \quad \dfrac{\pi_1[\pi_2/ax_x]}{\Gamma \vdash M[N/x] : 0}$$

Where $\pi_1[\pi_2/ax_x]$ is defined on induction on π_1, replacing occurrences of x by N and uses of the axiom rule on x by π_2. This induction produces a correct type derivation for $\Gamma \vdash M[N/x] : 0$. Now non deterministic choice :

$$\dfrac{\dfrac{\pi_1}{\Gamma \vdash M : k} \qquad \dfrac{\pi_2}{\Gamma \vdash N : k}}{\Gamma \vdash M + N : k}\, sum \quad \rightarrow \quad \dfrac{\pi_1}{\Gamma \vdash M : k} \text{ or } \dfrac{\pi_2}{\Gamma \vdash N : k}$$

Finally, the daimon :

$$\dfrac{\dfrac{\pi_1}{\Gamma \vdash \maltese : k+1} \qquad \dfrac{\pi_2}{\Gamma \vdash M : k}}{\Gamma \vdash \maltese M : 0}\, app \quad \rightarrow \quad \dfrac{}{\Gamma \vdash \maltese : 0}\, dai$$

□

2.2 Weak normalization

This calculus admits an easy weak normalization proof, by means of standard *à la Kleene* realizability. In what follows, the $M \rightsquigarrow N$ will denote the fact that all reduction sequences of M converge to N. This is still weak normalization since we only reduce head redexes, however non deterministic choice lead to nondeterministic reduction sequences.

Definition 12 (Realizability)**.** *We define a relation* $\Vdash \subseteq \Lambda \times \mathbb{N}$ *the following way :*

- $M \Vdash 0 \Leftrightarrow M \rightsquigarrow \maltese$
- $M \Vdash k+1 \Leftrightarrow \forall N \Vdash k,\ MN \rightsquigarrow \maltese$.

Then we can prove immediately the

Lemma 1 (Adequation)**.** *Suppose* $x_1 : n_1, \ldots, x_p : n_p \vdash M : k$*. Then for all* $N_1 \Vdash n_1, \ldots, N_p \Vdash n_p$*,* $M[N_1/x_1, \ldots, N_p/x_p] \Vdash k$*.*

Proof. By induction on the proof tree of M. Look at the root rule of the tree :

- *ax, dai.* trivial.
- *lam.* The last rule is
$$\dfrac{\Gamma, x : k \vdash M : 0}{\Gamma \vdash \lambda x.M : k+1}\, lam$$
Let $N_1 \Vdash n_1 \ldots N_p \Vdash n_p$. We need to show that $(\lambda x.M)[N_i/x_i] \Vdash k+1$. Suppose $N \Vdash k$, then :
$$((\lambda x.M)[N_i/x_i])N = (\lambda x.(M[N_i/x_i]))N \rightarrow M[N_i/x_i][N/x] \rightsquigarrow \maltese$$
The first step of the reduction is the only one to consider because we use a head reduction strategy. The other steps are justified by the induction hypothesis.
- *app.* The last rule is
$$\dfrac{\Gamma \vdash M : k+1 \qquad \Gamma \vdash N : k}{\Gamma \vdash MN : 0}\, app$$
Let $N_1 \Vdash n_1 \ldots N_p \Vdash n_p$. We need to show that $(MN)[N_i/x_i] \rightsquigarrow \maltese$. This is just by definition of realizability since by induction hypothesis, $M[N_i/x_i] \Vdash k+1$ and $N[N_i/x_i] \Vdash k$.

- *sum.* The last rule is
$$\frac{\Gamma \vdash M : k \qquad \Gamma \vdash N : k}{\Gamma \vdash M + N : k}\, sum$$
Let $N_1 \Vdash n_1 \dots N_p \Vdash n_p$. If $k = 0$, $(M + N)[N_i/x_i]$ reduces in one step to $M[N_i/x_i]$ or $N[N_i/x_i]$ which both realize 0 by induction hypothesis, thus $(M + N)[N_i/x_i] \rightsquigarrow \maltese$. Otherwise, we need to show that $(M+N)[N_i/x_i] \Vdash k+1$. For all $T \Vdash k$, $((M+N)[N_i/x_i])T$ reduces in once step either to $(M[N_i/x_i])T$ or to $(N[N_i/x_i])T$, which both normalize to $\maltese$ by induction hypothesis.

- *sub.* The last rule is
$$\frac{\Gamma \vdash M : k \qquad k \leqslant p}{\Gamma \vdash M : p}\, sub$$
We first prove that without context, if $M \Vdash k$ and $k \leqslant p$ then $M \Vdash p$. We do this by induction on $min(k, p)$.

  - If $min(k, p) = 0$, necessarily $k = 0$. If $p = 0$ it is evident, otherwise we write it $p + 1$. Suppose $N \Vdash p$. Now since $M \Vdash k = 0$, we have $M \rightsquigarrow \maltese$, therefore $MN \rightsquigarrow^* \maltese N \to \maltese$, thus $M \Vdash p + 1$.
  - If $min(k, p) > 0$, we write $k + 1$ and $p + 1$. Let $M \Vdash k + 1$, and $N \Vdash p$. Since $k + 1 \leqslant p + 1$ we have $p \leqslant k$, therefore $N \Vdash k$ by induction, thus $MN \rightsquigarrow \maltese$.

  Now the general case. Let $N_1 \Vdash n_1 \dots N_p \Vdash n_p$. We distinguish the same cases :

  - If $k = 0$, the case $p = 0$ is trivial. Otherwise we write it $p + 1$, and take $N \Vdash p$. By induction hypothesis, $M[N_i/x_i] \Vdash \maltese$, thus $(M[N_i/x_i])N \rightsquigarrow \maltese N \to \maltese$.
  - Finally, none of p and k are zero, we write $p + 1$ and $k + 1$. Let $N \Vdash p$, then by the above lemma $N \Vdash k$ ($p \leqslant k$ since $k+1 \leqslant p+1$). Thus by induction hypothesis $(M[N_i/x_i])N \rightsquigarrow \maltese$ which proves that $M[N_i/x_i] \Vdash p + 1$.

□

An immediate corollary of the adequation lemma is the weak normalization theorem.

Theorem 3 (Weak normalization)**.** *Let M be a typable closed term ($\vdash M : k$ is provable). Then any head reduction sequence of M is finite.*

Proof. Suppose $\vdash M : k$. By adequation lemma, $M \Vdash k$. If $k = 0$, $M \rightsquigarrow \maltese$ by definition of realizability. Otherwise we write $M \Vdash k + 1$. Now note that $\maltese \Vdash k$. Indeed, if $k = 0$ this is the definition and if $k > 0$, for all $N \Vdash k - 1$, $\maltese N \to \maltese$. Thus necessarily $M\maltese \rightsquigarrow \maltese$. But if there was an infinite reduction sequence of M, it would be as well an infinite reduction sequence of $M\maltese$, thus M is normalizable. □

2.3 Linear Head Reduction, definition and normalization

We know that the weak head reduction terminates, but what we are interested in is in fact the normalization of linear head reduction. These two normalization properties are in fact equivalent, but we will only prove here that normalization of weak head reduction implies normalization for head linear reduction. We follow Barendregt's convention, and we consider terms up to α-equivalence.

If x is a variable of T, we distinguish each occurrence $x_0, x_1, \dots, x_n$ of x in T. If x is a bound variable in T, we call *abstraction subterm* of λx the subterm of T beginning at λx.

Definition 13 (Head occurrences)**.** *Let T be a term of type k in Λ_u. We define its set of* ***head occurrences*** *$hoc(T)$ as follows :*

- $hoc(\lambda x.M) = hoc(M)$
- $hoc(x) = \{x\}$
- $hoc(MN) = hoc(M)$

- $hoc(M+N) = hoc(M) \cup hoc(N)$

Only an occurrence of a variable can be a head occurrence. We say sometimes that a variable x or ✠ is in *head position*. This means the same as above, but extended to ✠.

Definition 14 (head linear redexes)**.** *A head linear redex in a Λ_u term T is a β-redex where the variable to be substituted appears as a head occurrence. We denote a head linear redex by (x_i, A) where x_i is an occurrence of x in head position, and A is the argument subterm of the abstraction subterm of λx.*

Remark. For general head linear reduction, redexes are not necessarily β-redexes (only up to a behaviour preserving equivalence relation on terms). Howevers, one can show easily that for Λ_u, any such redex is also a β-redex, since only one variable at a time is abstracted. Therefore this definition suffices.

Definition 15 (Head Linear reduction)**.** *Let (x_i, A) be a head linear redex in T. The **head linear reduction** substitutes x_i by A (and only x_i, not the possible other occurrences of x). A itself is not affected.*

We build another realizability equivalence on Λ_u, based this time on linear head reduction. In what follows, $\to$ will denote linear head reduction. From this point, we will never talk again on the realizability relation $\Vdash$ defined above, therefore we reuse the same symbol for the new realizability relation.

Definition 16 (LHR-realizability)**.** *We define $\Vdash \subseteq \Lambda_u \times \mathbb{N}$ the following way :*

- $M \Vdash 0 \Leftrightarrow$ *All reductions of M terminates and lead to terms M' where only ✠ appears in head position.*
- $M \Vdash k+1 \Leftrightarrow \forall N \Vdash k, MN \Vdash 0$

The following lemma allows to link linear head reduction with β-reduction.

Lemma 2 (Consistency)**.** *Let $M, N \in \Lambda_u$, then*

$$(\lambda x.M)N \Vdash 0 \Leftrightarrow M[N/x] \Vdash 0$$

Proof. Let A denote the set of linear head reduction sequences of $(\lambda x.M)N$, and B the set of linear head reduction sequences of $M[N/x]$. Consider the application $\Phi : A \to B$ defined as follows. Take a reduction chain $u_1 \to u_2 \to \dots$ of A. Necessarily each u_i is of the form $u_i = (\lambda x.M_i)N$ by property of linear head reduction. We remove of $u_1 \to \dots$ each reduction step where an occurrence of x is substituted, and we set $u'_i := M'_i[N/x]$. It has an inverse which remplaces all copies of N by x, and inserts a substitution by N each time the first abstraction of a copy of N was reduced in the original reduction sequence.

The spaces of linear head reductions of $(\lambda x.M)N$ and $M[N/x]$ are isomorphic, therefore $(\lambda x.M)N \Vdash 0 \Leftrightarrow M[N/x] \Vdash 0$. □

Theorem 4 (Convergence of linear head reduction)**.** *Let $\vdash M : k$. Then any reduction sequence of M by linear head reduction is finite.*

Proof. First, notice that we have shown in 3 normalization of head β-reduction in presence of the rules for + and ✠. This extends to β and + by remarking that ✠ must appear in head position before applying its rule, and a term with a ✠ in head position is normal for β plus +. Finally this extends to β alone by an application of König's lemma.

If $k = 0$, the lemma above states that β-reduction preserves convergence. Therefore we normalize M using weak head beta reduction, stopping as soon as ✠ comes in head position. This happens by the remark above. By 2, the obtained normal form has all its linear head reductions finite if and only if the original term has all its linear head reductions finite.

If $k > 0$, we apply the reasoning above to M✠. □

2.4 Unary Bœhm Trees

We give here a variant of the calculus above, terms of which will arise as recomposition of agents on pointer structures. Hence they contain markers which point on parts of the terms which can be expanded, when more information about them is recovered from the pointer structures. Syntactically, these markers will behave as ✠, but will be annotated by an integer to denote the expanding sites. It is clear that these distinctions do not change anything to the dynamical behaviour of the terms, hence to normalization.

□-UBT

$$\frac{}{\Gamma \vdash \lambda x.\Box_i : k} \qquad \frac{\Gamma, x : k \vdash M_1 : p \quad \dots \quad \Gamma, x : k \vdash M_{|I|} : p}{\Gamma \vdash \lambda x.\, \oplus_{i\in I}\, x_i M_i : k+1}\ \forall i \in I,\ (x_i : p+1) \in \Gamma \cup \{x : k\}$$

Proposition 3 (Unique argument)**.** *Let $T \in \Box$-UBT. Then any occurrence of a variable in T has an unique argument (which may be a $\Box_i$). If h is an occurrence of a variable, let $arg(h)$ denote this unique argument.*

Proof. By induction on the type derivation of T. □

Definition 17. *Let $t \in \Box$-UBT. Let x be an occurrence of a variable in t. We define the* ***occurrence chain*** *of x by induction on t:*

- $occ_x(\lambda y.\, \oplus_{i\in I}\, z_i M_i) = [x]$ *if x is one of the z_i*
- $occ_x(\lambda y.\, \oplus_{i\in I}\, z_i M_i) = Cons(z_k, occ_x(M_k))$ *if x is in M_k.*

This definition is sane, because any occurrence defines an unique position in the term. Note that the occurrence chain is a list rather than a set, because the order is relevant : it is a path from the root of the term to x.

Definition 18. *Let $t \in \Box$-UBT. Let x be an occurrence of a variable in t. We define the* ***subterm chain*** *of x by induction on t:*

- $sub_x(\lambda y.\, \oplus_{i\in I}\, z_i M_i) = [\lambda y.\, \oplus_{i\in I}\, z_i M_i)]$ *if x is one of the z_i*
- $sub_x(\lambda y.\, \oplus_{i\in I}\, z_i M_i) = Cons(\lambda y.\, \oplus_{i\in I}\, z_i M_i, sub_x(M_k))$ *if x is in M_k*

Proposition 4. *Let $t \in \Box$-UBT, and a proof π of $\vdash t : k$. Then, just by incrementing by 2 all the types in π, we get a proof π' of $\vdash t : k+2$.*

Proof. By induction on π. □

Definition 19 (□-expansion)**.** *Take a proof π in $\Box$-UBT and a subtree T of π. Let $Var(T)$ be the set of variable in the context at the root of T, including the possible variable abstracted at the root of T. Take $y \in Var(T)$. We define the* ***□-expansion of π at T along y****, denoted $exp_\Box(\pi, T, y)$, the following way. We look at the form of T and the type of y, and rewrite π the following way :*

- *If T is of the form $\lambda x.\Box_i$ and $y : 1 \in \Gamma \cup \{x : k\}$, we replace every type k in π by $k+2$ while performing the following substitution on T :*

$$\frac{}{\Gamma \vdash \lambda x.\Box_i : k+1} \quad \rightsquigarrow \quad \frac{\dfrac{}{\Gamma, x : k+2 \vdash \lambda z.\Box_i : 2}}{\Gamma \vdash \lambda x.y(\lambda z.\Box_i) : k+3}$$

- *If T is of the form $\lambda x.\Box_i$ and $y : p+2 \in \Gamma \cup \{x : k\}$, we perform the following substitution on T, leaving the rest of the proof unchanged.*

$$\frac{}{\Gamma \vdash \lambda x.\Box_i : k+1} \quad \rightsquigarrow \quad \frac{\dfrac{}{\Gamma, x : k \vdash \lambda z.\Box_i : p+1}}{\Gamma \vdash \lambda x.y(\lambda z.\Box_i) : k+1}$$

- *If T is of the form $\lambda x.(\oplus_{i\in I} x_i M_i)$ and $y \in \{x_i \mid i \in I\}$, the expansion is the identity. In the following cases, we suppose $y \notin \{x_i \mid i \in I\}$.*
- *If T is of the form $\lambda x.(\oplus_{i\in I} x_i M_i)$, and $y : 1 \in \Gamma \cup \{x : k\}$, we replace every type k in π by $k+2$ while performing the following substitution on T :*

$$\frac{\pi_1 \dots \pi_{|I|}}{\Gamma \vdash \lambda x.\, \oplus_{i\in I}\, x_i M_i : k+1} \quad \rightsquigarrow \quad \frac{\pi_1 \dots \pi_{|I|} \qquad \overline{\Gamma, x : k+2 \vdash \lambda z.\square_j : 2}}{\Gamma \vdash \lambda x.(\oplus_{i\in I} x_i M_i) + y(\lambda z.\square_j) : k+3}\ (\square_j \textit{ fresh})$$

- *If T is of the form $\lambda x.(\oplus_{i\in I} x_i M_i)$ and $y : p+2 \in \Gamma \cup \{x : k\}$ we perform the following substitution on T, leaving the rest of π unchanged :*

$$\frac{\pi_1 \dots \pi_{|I|}}{\Gamma \vdash \lambda x.\, \oplus_{i\in I}\, x_i M_i : k+1} \quad \rightsquigarrow \quad \frac{\pi_1 \dots \pi_{|I|} \qquad \overline{\Gamma, x : k \vdash \lambda z.\square_j : p+1}}{\Gamma \vdash \lambda x.(\oplus_{i\in I} x_i M_i) + y(\lambda z.\square_j) : k+1}\ (\square_j \textit{ fresh})$$

Clearly, any expansion of a $\square$-UBT is a $\square$-UBT. This expansion can be seen of the addition of new information about the behaviour of a strategy in its P-view tree.

3 Normalization proof for pointer structures.

We give here a proof of the normalization theorem for pointer structures. The proof is a bit technical, it inlines a part of the definition and correction of the PAM. Consider a pointer structure f.

Induction invariant. We build by induction the following data:

I Two terms $\vdash t_1 : k_1 \in \square$-UBT and $\Gamma \vdash t_2 : k_2 \in \triangle$-UBT along with their type derivations π_1 and π_2.

II Let $d = \max_{i \leq N} d_f(i)$ be the pointer depth of f until N, then :

- If d is even, then $k_1 = d + 2$ and $k_2 = d + 1$
- If d is odd, then $k_1 = d + 3$ and $k_2 = d + 2$

In particular, $k_2 = k_1 - 1$.

III A sequence of linear head reductions $u_1 \to \cdots \to u_N$ such that $u_1 = t_1 t_2$

IV A sequence of pairs $(h_i, A_i)_{i \in \{0 \ldots N\}}$ where h_i are occurrences of variables (or $*$ at the initilization of the machine) in u_0 and A_i are subterms of u_0 such that :

(1) If $N > 0$, $\forall i \in \{0 \ldots N-1\}$, h_{i+1} is a head occurrence of A_i.

(2) $\forall i \in \{1 \ldots N\}$, h_i is bounded by the first abstraction of $A_{f(i)}$.

(3) $\forall i \in \{1 \ldots N-1\}$, (h_i, A_i) is an linear redex (reduced in $u_i \to u_{i+1}$).

(4) $\forall i \leq N$, $A_i = arg(h_{f(i)})$.

(5) $\forall i \in \{1 \ldots N\}$, consider the sequence $k_1, \ldots, k_p$ defined as $\{j \in \ulcorner f \urcorner(i) \mid j \not\equiv i\}$, sorted by $<$. Then $A_{k_1}, \ldots, A_{k_p}$ is the subterm chain of h_i.

(6) $\forall i \in \{1 \ldots N\}$, consider the sequence $k_1, \ldots, k_p$ defined as $\{j \in \ulcorner f \urcorner(i) \mid j \equiv i\}$, sorted by $<$. Then $h_{k_1}, \ldots, h_{k_p}$ is the occurrence chain of h_i.

V For $i \geq 1$, let $type(A_i)$ (resp. $type(h_i)$) denote the type annotation of the subterm (resp. the occurrence) in π_1 or π_2. Then it satisfies :

$$type(A_i) = type(h_i) = d + d[2] - d_f(i) + 2$$

Initialization. $N = 0$ is just the bootstrap of the machine and the first reduction occurs at $N = 1$. Thus we assume that f is defined at least up to 1, and we set, for $N = 1$:

I We choose $t_1 = \lambda x_0.x_0(\lambda x_1.\square_0) : 4$ and $t_2 = \lambda y_0.\triangle_0 : 3$, with the following type derivations :

$$\pi_1 = \dfrac{\dfrac{}{x_0 : 3 \vdash \lambda x_1.\square_0 : 2}}{\vdash \lambda x_0.x_0(\lambda x_1.\square_0) : 4} \qquad \pi_2 = \dfrac{}{\vdash \lambda y_0.\triangle_0 : 3}$$

II The depth of f until $N = 1$ is 1 ($f(1) = 0$ is forced by the axioms of pointer structures), thus it satisfies the requirements.

III The derivation sequence is the trivial sequence $t_1 t_2$.

IV The state of the machine is $(*, t_1), (x_0, t_2)$. It satisfies the requirements.

V We just have to check for $i = 1$.

$$type(A_1) = 1 + 1 - 1 + 2 = 3 = type(t_2)$$

$$type(h_1) = 1 + 1 - 1 + 2 = 3 = type(x_0)$$

Run of the machine. Suppose the machine has been run until step N. If $f(N+1) = \bot$, the machine halts. Otherwise, we consider $A_{f(N+1)}$. We need to show that $A_{f(N+1)}$ begins with an abstraction of a variable which is in the typing context of A_N so that it has a sense to have this variable as a head occurrence of A_N. There are two cases : either $f(N+1) = N$ and it is evident, or by visibility, $f(N+1) \in \ulcorner f \urcorner(N+1) = \ulcorner f \urcorner(f(N))$ and $f(N+1) \not\equiv f(N)$. Thus, by IV.5, $A_{f(N+1)}$ is in the subterm chain of $h_{f(N)}$. But by IV.4, A_N is the argument subterm of $h_{f(N)}$, thus the variable abstracted at $A_{f(N+1)}$ is necessarily in the typing context of A_N. Now we distinguish several cases :

- x is already a head occurrence of A_N. This means that we do not need to expand the terms. This means as well that $f(N+1)$ has already been pointed thus the depth (or equivalently, the types or t_1 and t_2) need to change. We review point by point the definition of the invariant :

  I $t_1, t_2, k_1, k_2, \pi_1, \pi_2$ are unchanged.

  II d is unchanged, nothing to check.

  III We extand $u_1 \to \cdots \to u_{N+1}$. By hypothesis, this can be done simply by reducing the linear redex (h_N, A_N).

  IV We set h_{N+1} to x (which is by hypothesis a head occurrence of A_N), and $A_{N+1} = arg(h_{f(N+1)})$. As needed :

  (1) By definition of h_{N+1}.

  (2) By construction, h_{N+1} is an occurrence of the variable beeing abstracted at $A_{f(N+1)}$, therefore with the induction hypothesis this is true for $i \leq N+1$.

  (3) (h_{N+1}, A_{N+1}) is a linear redex in u_{N+1} : (h_N, A_N) was a linear redex in u_N (by IV.3), and h_N has been substituted by A_N which has h_{N+1} as head occurrence. Moreover, h_{N+1} was abstracted at $A_{f(N+1)}$ (by IV.2), and since $(h_{f(N+1)}, A_{f(N+1)})$ was a linear redex by IV.3, $h_{f(N+1)}$ was substitued by $A_{f(N+1)}$ at $u_{f(N+1)} \to u_{f(N+1)+1}$, thus the argument of $h_{f(N+1)}$ is the subterm corresponding to h_{N+1}, thus (h_{N+1}, A_{N+1}) is a linear redex.

  (4) $A_{N+1} = arg(h_{f(N+1)})$ by construction.

  (5) $\{j \in \ulcorner f \urcorner(f(N)) \mid j \not\equiv f(N)\}$ sorted by $<$ corresponds to the subterm chain of $h_{f(N)}$ by IV.5. But $h_{N+1} \in hoc(A_N) = hoc(arg(h_{f(N)}))$ (by IV.1 and IV.4), thus the subterm chain of h_{N+1} is just the subterm chain of $h_{f(N)}$ appended with A_N. But $\{j \in \ulcorner f \urcorner(N+1) \mid j \not\equiv N+1\} = \{N\} \cup \{j \in \ulcorner f \urcorner(f(N)) \mid j \not\equiv f(N)\}$, which concludes.

  (6) $\{j \in \ulcorner f \urcorner(f(N)) \mid j \equiv f(N)\}$ sorted by $<$ corresponds to the occurrence chain of $h_{f(N)}$ by IV.6. But $h_{N+1} \in hoc(A_N) = hoc(arg(h_{f(N)}))$ (by IV.1 and IV.4), thus the occurrence chain of h_{N+1} is just the occurrence chain of $h_{f(N)}$ appended with h_{N+1}. But $\{j \in \ulcorner f \urcorner(N+1) \mid j \equiv N+1\} = \{N+1\} \cup \{j \in \ulcorner f \urcorner(f(N)) \mid j \equiv f(N)\}$, which concludes.

  V The depth has not changed, therefore we only need to check the equalities for A_{N+1} and h_{N+1}.

$$\begin{aligned} type(A_{N+1}) &= type(h_{f(N+1)}) - 1 \quad (A_{N+1} \text{ is the argument of } h_{f(N+1)}) \\ &= d + d[2] - d_f(f(N+1)) + 2 - 1 \\ &= d + d[2] - (d_{f(N+1)} - 1) + 2 - 1 \\ &= d + d[2] - d_f(N+1) + 2 \end{aligned}$$

$$\begin{aligned} type(h_{N+1}) &= type(A_{f(N+1)}) - 1 \quad (h_{N+1} \text{ is abstracted at } A_{f(N+1)}) \\ &= d + d[2] - d_f(N+1) + 2 \end{aligned}$$

- x is not a head occurrence of A_N, but we know it should have type $type(A_{f(N+1)}) - 1$. Necessarily, $type(A_{f(N+1)}) \geq 2$ (this is an easy consequence of the formula). Suppose the inequality is strict, we can safely expand π_1 at A_N along x (or π_2, depending on which one A_N is a subterm) without changing the types. Same checks as above :

I Depending on the equivalence class modulo 2 of N, either t_1, π_1 or t_2, π_2 are expanded. k_1, k_2 are unchanged, and the definition of expansion ensures that π_1 and π_2 are still proofs of $\vdash t_1 : k_1$ and $\vdash t_2 : k_2$.

II Two cases arise. If $d = \max_{i \leq N} d_f(i)$, we may have $d_f(N+1) \leq d$, or $d_f(N+1) = d+1$.

* If $d_f(N+1) \leq d$, there is nothing to check since both d and k_1, k_2 are unchanged.
* Otherwise, d is odd. Indeed,

$$\begin{aligned} type(A_{f(N+1)}) > 2 &\Leftrightarrow d + d[2] - d_f(f(N+1)) + 2 > 2 \\ &\Leftrightarrow d + d[2] > d_f(f(N+1)) \\ &\Leftrightarrow d + d[2] > d \\ &\Leftrightarrow d[2] = 1 \end{aligned}$$

Therefore, the new depth $d+1$ is even, thus the equalities still hold.

III,IV The dynamical concerns are exactly the same, thus the proof is the same.

V We use the remark in **II** that if the depth changes, it goes from odd to even, therefore $d + d[2]$ does not change, and neither do we change the type derivations. Therefore the same argument as in the first case applies.

- x is not a head occurrence of A_N, and $type(A_{f(N+1)}) = 2$. This can only happen in a very particular situation :

$$\begin{aligned} type(A_{f(N+1)}) = 2 &\Leftrightarrow d + d[2] - d_f(f(N+1)) + 2 = 2 \\ &\Leftrightarrow d + d[2] = d_f(f(N+1)) \end{aligned}$$

But since $d_f(f(N+1)) \leq d$, this can only happen if d is even and $d_f(f(N+1)) = d$. Note that this implies that $f(N+1)$ is even (because $\forall i \in \mathcal{D}_f,\ f(i) \not\equiv i$), thus $N+1$ is odd, which means that A_N is in t_1. Thus we'll have to expand t_1 at A_N along x. As above, we update the invariant :

I We define the following :

$$\begin{aligned} t'_1 &= exp_{\square}(t_1, A_N, x) \\ t'_2 &= t_2 \\ k'_1 &= k_1 + 2 \\ k'_2 &= k_2 + 2 \end{aligned}$$

II d is even, thus $k_1 = d+2$ and $k_2 = d+1$ by induction hypothesis. Thus $k'_1 = d' + 3$ and $k'_2 = d' + 2$, which is perfectly correct since d' is odd.

III,IV As above, the dynamical part of the invariant of the invariant are as in the first case.

V $d' + d'[2] = d + d[2] + 2$, thus the formula is preserved for all $i \leq N$. For $N+1$, the proof is the same as in the first case, but with d replaced by d'.

Termination. We are going to prove that if d_f is bounded but f is infinite, then the memory of f is infinite. Suppose d_f bounded, we make the following observations :

(1) By the induction invariant, the types of π_1 and π_2 are bounded as well.

(2) We apply the following construction to π_1 and π_2. Take each non-leaf node $\lambda x. \bigoplus_{i \in I} x_i M_i$. Separate it into $|I|+1$ nodes : $\lambda x, x_1 M_1, \ldots, x_{|I|} M_{|I|}$. Each occurrence node will point to its corresponding λ node, each λ node will point to the occurrence node for which it is an argument (if it exists). This way we get two trees π'_1 and π'_2. They both have finite depth by remark (1).

(3) If π'_1 and π'_2 were both finite, this would mean that there is $N \in \mathbb{N}$ after which no expansion is made on t_1 and t_2. Therefore $u_1 \to u_2 \to \dots$ would be an infinite head linear reduction sequence of the finite term $t_1 t_2$, which has type 0 by construction. This is impossible, since by 4, Λ_u has only finite linear head reduction sequences. Therefore one of those trees is infinite, let π'_i denote this infinite tree.

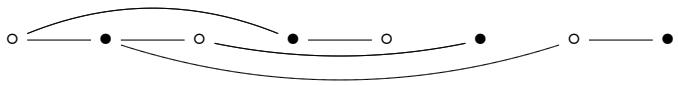
(4) Now, by König's lemma, since this π'_i tree has its depth bounded, it must have a node with an infinite degree. Since each occurrence node has by construction an unique son in the tree, it must be an abstraction node.

(5) Let call x the repeated variable. Consider again this π'_i where x is infinitely repeated. We focus back on its original Böhm tree π_i. Since all terms are unary, its only branchings are the sums. We remove all branches where x does not appear : the obtained tree is necessarily still infinite. However it is finitely branching, because all sums are finite. Therefore it has an infinite branch, where each occurrence of x is in the argument subterm of the previous.

(6) Consider this infinite branch $x_1, x_2, \dots$. Let $\psi_1, \psi_2, \dots$ be the indices when each of these occurrences of x have been introduced. Then $\forall i \in \mathbb{N}^*$, $\forall j \in \{1 \dots i\}$, ψ_i sees ψ_j. Indeed, each of these occurrences is in the argument subterm of the previous, therefore for a fixed occurrence x_i of this infinite chain, every x_j with $j \leq i$ is in the occurrence chain of x_i. Thus, by IV.6, any $j \leq i$ is such that $\psi_j \in \ulcorner f \urcorner(\psi_i)$. Remains to prove that $\forall i, j \geq 1$, $f(\psi_i) = f(\psi_j)$. Take $1 < i < j$, then by IV.5 the subterm chain of h_{ψ_i} is a prefix of the subterm chain of h_{ψ_j} (since they correspond to the views). Now it is trivial from the definition that a given subterm appears at most once in a subterm chain, thus the abstraction subterm of h_{ψ_i} appears at most once in the subterm chain of h_{ψ_i}. Let us summarize : ψ_i and ψ_j must both point to $f(\psi_i)$ and $f(\psi_j)$ such that $A_{f(\psi_i)} = A_{f(\psi_j)}$ is their corresponding abstraction subterm. But by visibility they point into their views which are prefix of one another and which are their subterm chains by IV.5, which contain no more that one occurrence of the corresponding abstraction subterm. Conclusion : they must point to the same integer.

Finally, we have successfully built a conscious ω-fork from an infinite but bounded pointer structure, which concludes the proof.

Example. To help the reader getting an idea of how the proof works, we include an example of the dynamics of the machine. Consider the following finite visible pointer structure :



The diagram is intended to be read from left to right. Each ∘ or • represents an integer, and points to its image by f. We describe the computation of the machine when run on this input. At each step i, h_i will be in blue and A_i will be in red. There is an inaccuracy in the diagram below : when the machine updates the terms t_4 and t_2, it updates them retroactively in the part of the computation that has been already done, in order to have at any moment a valid reduction sequence for the last version of t_1 and t_2. Here, we will show only at each step the part of the terms which is already known at this moment.

$*\lambda x_0.x_0(\lambda x_1.\square_0$) : 4 $\lambda y_0.\triangle_0$: 3

$\lambda x_0.x_0(\lambda x_1.\square_0$) : 4 $\lambda y_0.\triangle_0$: 3

$\lambda x_0.x_0(\lambda x_1.\square_0$) : 4 $\lambda y_0.y_0(\lambda y_1.\triangle_0$) : 3

$\lambda x_0.x_0(\lambda x_1.\ x_0(\lambda x_2.\square_0$)) : 4 $\lambda y_0.y_0(\lambda y_1.\triangle_0$) : 3

$\lambda x_0.x_0(\lambda x_1.\ x_0(\lambda x_2.\square_0$)) : 4 $\lambda y_0.y_0(\lambda y_1.\triangle_0$) : 3

$\lambda x_0.x_0(\lambda x_1.\ x_0(\lambda x_2.x_1(\lambda x_3.\square_0))$) : 6 $\lambda y_0.y_0(\lambda y_1.\triangle_0$) : 5

$\lambda x_0.x_0(\lambda x_1.\ x_0(\lambda x_2.x_1(\lambda x_3.\square_0))$) : 6 $\lambda y_0.y_0(\lambda y_1.y_0(\lambda y_2.\triangle_0))$: 5

$\lambda x_0.x_0(\lambda x_1.[x_0(\lambda x_2.x_1(\lambda x_3.\square_0))] + [x_1(\lambda x_2.\square_1)])$: 6 $\lambda y_0.y_0(\lambda y_1.y_0(\lambda y_2.\triangle_0))$: 5

And the machine stops since f is no longer defined.