

A Method for Termination in a λ -calculus with References

(paper in preparation)

R. Demangeon
ENS Lyon

D. Hirschhoff
ENS Lyon

D. Sangiorgi
INRIA/University of Bologna

Abstract

We present a method for ensuring termination of λ -calculus with references, that makes it possible to combine term rewriting measure-based techniques with traditional approaches for termination in functional languages such as logical relations. The method can be made parametric on the termination technique employed on the functional parts.

1 A λ -calculus with references

1.1 Language and type system

In this section, our technique is adapted to λ_{ref} , a call-by-value λ -calculus extended with imperative operations (reference allocation, dereferencing, update) acting on a store. With respect to the π -calculus, intuitively, references correspond to imperative names, and functions to functional names. We transport the constraints for π -calculus onto the λ -calculus following this analogy. In λ_{ref} certain level and typing annotations are directly placed into the syntax (this was not mandatory, but it simplifies certain technical details).

The store is stratified into regions, which are referred to using natural numbers. Commands involving imperative operations are annotated by a natural number: a command acts on regions of a given level.

To define terms, we use two disjoint sets \mathcal{V} of variables, ranged over using x, y, z, \dots , and \mathcal{A} of addresses. Addresses are written $u_{(n,T)}$: they are explicitly associated both to a region n and a type T (types are described below). Note that values of different types can be stored in the same region. We suppose that there exists an infinite number of addresses for a given pair of a type and a region. Stores, ranged over using δ , are partial mappings from addresses to values. The (finite) support of δ is written $\text{supp}(\delta)$, \emptyset is the empty store ($\text{supp}(\emptyset) = \emptyset$), and $\delta \langle u_{(n,T)} \rightsquigarrow V \rangle$ denotes the store δ' defined by $\delta'(u_{(n,T)}) = V$ and $\delta'(v) = \delta(v)$ for every $v \in \text{supp}(\delta)$ such that $v \neq u_{(n,T)}$.

The syntax for terms, types, values, redexes and evaluation contexts is as follows:

$$\begin{aligned}
 M & ::= (M M) \mid x \mid \lambda x. M \mid \star \\
 & \quad \mid \mathbf{ref}_n M \mid \mathbf{deref}_n(M) \mid M :=_n M \mid u_{(n,T)} \\
 T & ::= \mathbf{1} \mid T \mathbf{ref}_n \mid T \rightarrow^n T \\
 V & ::= \lambda x. M \mid x \mid u_{(n,T)} \mid \star \\
 R & ::= (\lambda x. M) V \\
 & \quad \mid \mathbf{deref}_n(u_{(n,T)}) \mid \mathbf{ref}_n V \mid u_{(n,T)} :=_n V \\
 \mathbf{E} & ::= [] \mid V \mathbf{E} \mid \mathbf{E} M \\
 & \quad \mid \mathbf{deref}_n(\mathbf{E}) \mid \mathbf{ref}_n \mathbf{E} \mid \mathbf{E} :=_n M \mid V :=_n \mathbf{E}
 \end{aligned}$$

Typing rules for terms

$$\begin{array}{c}
\text{(App)} \frac{\vdash_{\Gamma} M : (T_1 \rightarrow^n T_2, m) \quad \vdash_{\Gamma} N : (T_1, k)}{\vdash_{\Gamma} M N : (T_2, \max(m, n, k))} \qquad \text{(Abs)} \frac{\vdash_{\Gamma} M : (T_2, n) \quad \Gamma(x) = T_1}{\vdash_{\Gamma} \lambda x. M : (T_1 \rightarrow^n T_2, 0)} \\
\\
\text{(Ref)} \frac{\vdash_{\Gamma} M : (T_1, m)}{\vdash_{\Gamma} \text{ref}_n M : (T_1 \text{ ref}_n, \max(n, m))} \qquad \text{(Var)} \frac{\Gamma(x) = T_1}{\vdash_{\Gamma} x : (T_1, 0)} \qquad \text{(Uni)} \frac{}{\vdash_{\Gamma} \star : (\mathbb{1}, 0)} \\
\\
\text{(Add)} \frac{}{\vdash_{\Gamma} u_{(n, T_1)} : (T_1 \text{ ref}_n, 0)} \qquad \text{(Asg)} \frac{\vdash_{\Gamma} M : (T_1 \text{ ref}_n, m) \quad \vdash_{\Gamma} N : (T_1, k)}{\vdash_{\Gamma} M :=_n N : (\mathbb{1}, \max(m, n, k))} \\
\\
\text{(Drf)} \frac{\vdash_{\Gamma} M : (T \text{ ref}_n, m)}{\vdash_{\Gamma} \text{deref}_n(M) : (T, \max(m, n))}
\end{array}$$

Typing rules for stores

$$\begin{array}{c}
\text{(Emp)} \frac{}{\vdash_{\Gamma} \emptyset} \qquad \text{(Sto)} \frac{\vdash_{\Gamma} \delta \quad \vdash_{\Gamma} V : (T, 0)}{\vdash_{\Gamma} \delta \langle u_{(n, T)} \rightsquigarrow V \rangle}
\end{array}$$

Figure 1: λ_{ref} : Type and Effect System

We impose a well-formedness condition on types, that intuitively reflects the stratification of regions: a term acting at regions less than or equal to n cannot be stored in a region smaller than $n + 1$. For this, we define $\text{reg}(T)$, the set of regions accessed by T , by:

$$\begin{aligned}
\text{reg}(\mathbb{1}) &= 0 & \text{reg}(T \text{ ref}_n) &= \max(n, \text{reg}(T)) \\
\text{reg}(T_1 \rightarrow^n T_2) &= \max(n, \text{reg}(T_2))
\end{aligned}$$

Definition 1 (Well-formedness of types) *A type T is well-formed, written $\text{wf}(T)$, if for all its subtypes of the form $T' \text{ ref}_n$, we have $\text{reg}(T') < n$.*

In the following, we shall implicitly assume that all types we manipulate are well-formed.

Figure 1 defines two typing judgements, of the form $\vdash_{\Gamma} M : (T, n)$ for terms and $\vdash_{\Gamma} \delta$ for stores. As above, our type system is presented *à la Church*, and we write $\Gamma(x) = T$ whenever variable x has type T according to Γ .

In the type and effect system of λ_{ref} , a typing judgement has the form $\vdash_{\Gamma} M : (T, n)$, where n defines a bound on the *effect* of the evaluation of M , intuitively the maximum level of a region accessed when evaluating M . effects can be thought of as sets of regions, and are given by a natural number, intuitively corresponding to the maximum level of a region in the effect. Values (functions, variables, constants and addresses) have effect 0.

Contrarily to the π -calculus case, the typing rules do not feature inequality constraints on levels, as the stratification of the store is guaranteed by the well-formedness of types. Their main purpose is to record the effect of computations. In contrast with the π -calculus, the measure stratification for λ -calculus is purely on the syntax of types. The type system is similar to Boudol [Bou07] and Amadio [Ama09]. The termination proof, however, in Section 1.2, is quite different.

We extend typing to evaluation contexts by treating the hole as a term variable which can be given any type.

The execution of programs is specified by a reduction relation written \mapsto , relating pairs consisting of a term and a store, and which is defined on Figure 2 ($M\{V/x\}$ stands for the capture-avoiding substitution of x with V in M).

$$\begin{array}{c}
(\beta) \frac{}{(\lambda x. M \ V, \delta) \mapsto (M\{V/x\}, \delta)} \qquad (\mathbf{ref}) \frac{u_{(n,T)} \notin \text{supp}(\delta) \quad \vdash_{\Gamma} V : (T, -)}{(\mathbf{ref}_n \ V, \delta) \mapsto (u_{(n,T)}, (\delta \langle u_{(n,T)} \rightsquigarrow V \rangle))} \\
(\mathbf{deref}) \frac{\delta(u_{(n,T)}) = V}{(\mathbf{deref}_n(u_{(n,T)}), \delta) \mapsto (V, \delta)} \qquad (\mathbf{store}) \frac{\vdash_{\Gamma} V : (T, -)}{(u_{(n,T)} :=_n V, (\delta)) \mapsto (\star, (\delta \langle u_{(n,T)} \rightsquigarrow V \rangle))} \\
(\mathbf{context}) \frac{(M, \delta) \mapsto (M', \delta')}{(\mathbf{E}[M], \delta) \mapsto (\mathbf{E}[M'], \delta')}
\end{array}$$

Figure 2: $\lambda_{\mathbf{ref}}$: Reduction Rules

As above, we write \mapsto_{Γ}^n for a *functional* reduction, obtained using rule (β) ; n refers to the effect of the β -redex, that is, in this call-by-value setting, the level that decorates the type of the function being triggered (that is, we suppose in rule (β) that $\vdash_{\Gamma} \lambda x. M : (T_V \rightarrow^n T, -)$). We introduce similarly *imperative* reductions, noted \mapsto_{Γ}^n , for reductions obtained using rules (\mathbf{ref}) , (\mathbf{deref}) or (\mathbf{store}) (in these cases, the level n appears explicitly in the rules of Figure 2).

Fact 2 (Determinism)

Given M , either M is a value or there exist a unique evaluation context \mathbf{E} and redex R such that $M = \mathbf{E}[R]$.

Proof. By structural induction on M . □

First, we notice that values have an effect 0.

Fact 3 (Value effect)

If V is a value and $\vdash_{\Gamma} V : (T, m)$, then $m = 0$.

Proof. By examining the last rule used to derive $\vdash_{\Gamma} V : (T, m)$. Indeed, rules (\mathbf{Abs}) , (\mathbf{Var}) , (\mathbf{Uni}) and (\mathbf{Add}) have an effect 0 in their conclusion. □

The following result will be used in the proof of Lemma 5:

Lemma 4 (Subtyping)

If $\vdash_{\Gamma} \mathbf{E}[M_1] : (T, n)$, $\vdash_{\Gamma} M_1 : (T_1, m_1)$, $\vdash_{\Gamma} M_{(1)} : (T_1, m_{(1)})$ and $m_{(1)} \leq m_1$, then $\vdash_{\Gamma} \mathbf{E}[M_{(1)}] : (T, n')$ with $n' \leq n$.

Proof. By structural induction on \mathbf{E} :

- Case \square . Then $m_1 = n$ and $T = T_1$. We set $n' = m_{(1)}$. As $m_{(1)} \leq m_1$ we conclude.
- Case $(\mathbf{E}_2 \ M_2)$. We derive $\vdash_{\Gamma} M_2 : (T_2, n^{(2)})$ and $\vdash_{\Gamma} \mathbf{E}_2[M_1] : (T_2 \rightarrow^{n^2} T, n_2)$ with $n = \max(n_2, n^2, n^{(2)})$. By the induction hypothesis we get $\vdash_{\Gamma} \mathbf{E}_2[M_{(1)}] : (T_2 \rightarrow^{n^2} T, n'_2)$ with $n'_2 \leq n_2$. We derive $\vdash_{\Gamma} (\mathbf{E}_2[M_{(1)}] \ V_2) : (T, \max(n'_2, n^2, n^{(2)}))$ using rule (\mathbf{App}) . As $n'_2 \leq n_2$, we conclude.
- Case $\mathbf{ref}_{n^2} \ \mathbf{E}_2$. We derive $\vdash_{\Gamma} \mathbf{E}_2[M_1] : (T_2, n_2)$ with $n = \max(n_2, n^2)$, $T = T_2 \ \mathbf{ref}_{n^2}$ and T is well-formed. By the induction hypothesis we get $\vdash_{\Gamma} \mathbf{E}_2[M_{(1)}] : (T_2, n'_2)$ with $n'_2 \leq n_2$. We derive $\vdash_{\Gamma} \mathbf{ref}_{n^2} \ \mathbf{E}_2[M_{(1)}] : (T, \max(n'_2, n^2))$ using rule (\mathbf{App}) . As $n'_2 \leq n_2$, we conclude.
- The other cases are similar.

□

Our type and effect system enjoys the two following standard properties:

Lemma 5 (Subject substitution)

If $\vdash_{\Gamma} M : (T, n)$, $\Gamma(x) = T'$ and $\vdash_{\Gamma} V : (T', m)$ then $\vdash_{\Gamma} M\{V/x\} : (T, n)$.

Proof. First notice that Fact 3 implies $m = 0$.

From $\vdash_{\Gamma} M : (T, n)$ and $\vdash_{\Gamma} V : (T', m)$, we derive $\vdash_{\Gamma} M\{V/x\} : (T, n)$. We proceed by induction on the typing derivation:

- Case **(Var)**. Rule **(Var)** gives $\vdash_{\Gamma} y : (T, 0)$.
 - Either $x = y$, $T = T'$ and $M\{V/x\} = V$. As $n = m = 0$, we conclude.
 - Or $x \neq y$ and $M\{V/x\} = y$. We use **(Var)** to derive $\vdash_{\Gamma} y\{V/x\} : (T, 0)$.
- Case **(App)**. We have $M = (M_1 M_2)$. We derive $\vdash_{\Gamma} M_1 : (T_2 \rightarrow^{n_3} T, n_1)$ and $\vdash_{\Gamma} M_2 : (T_2, n_2)$ with $n = \max(n_1, n_2, n_3)$. We use the induction hypothesis to get $\vdash_{\Gamma} M_1\{V/x\} : (T_2 \rightarrow^{n_3} T, n_1)$ and $\vdash_{\Gamma} M_2\{V/x\} : (T_2, n_2)$ and we conclude using rule **(App)**.
- Case **(Abs)**. We have $M = \lambda y.M_1$, $\Gamma(y) = T_2$, and $T = T_2 \rightarrow^{n_1} T_1$. We derive $\vdash_{\Gamma} M_1 : (T_1, n_1)$. We use the induction hypothesis to get $\vdash_{\Gamma} M_1\{V/x\} : (T_1, n_1)$ and we conclude using rule **(Abs)**.
- Case **(Add)**. We have $M = M' = u_{(n, T')}$ and we use **(Add)** to derive $\vdash_{\Gamma} u_{(n, T')}\{V/x\} : (T, 0)$.
 item Case **(Ref)**. We have $M = \mathbf{ref}_m M_1$. We derive $\vdash_{\Gamma} M_1 : (T_1, n_1)$ with $T = T_1 \mathbf{ref}_m$ and $n = \max(n_1, m)$. We use the induction hypothesis to get $\vdash_{\Gamma} M_1\{V/x\} : (T_1, n_1)$ and we conclude using rule **(Ref)**.
- Cases **(Sto)**, **(Drf)** are similar. Case **(Uni)** is easy.

□

Proposition 6 (Subject reduction)

$\vdash_{\Gamma} M : (T, n)$, $\vdash_{\Gamma} \delta$ and $(M, \delta) \mapsto (M', \delta')$ entail that $\vdash_{\Gamma} \delta'$ and $\vdash_{\Gamma} M' : (T, n')$ for some $n' \leq n$.

Proof. By induction on the derivation of $(M, \delta) \mapsto (M', \delta')$,

- Case **(context)**. Then $M = \mathbf{E}[M_1]$, $M' = \mathbf{E}[M'_1]$ and $(M_1, \delta) \mapsto (M'_1, \delta')$. We derive $\vdash_{\Gamma} M_1 : (T_1, n_1)$. The induction hypothesis gives us $\vdash_{\Gamma} M'_1 : (T_1, n'_1)$ with $n'_1 \leq n_1$ and a typing judgment for δ' . As $n'_1 \leq n_1$, we use Lemma 4 to conclude.
- Case **(β)**. We have $M = \lambda x : T_V.M_1 V$. We derive $\vdash_{\Gamma} M_1 : (T, n)$, $\vdash_{\Gamma} V : (T_V, 0)$ (this holds as $\lambda x : T_V.M_1$ has type $T_V \rightarrow^n T$). We use Lemma 5 and get $\vdash_{\Gamma} M_1\{V/x\} : (T, n)$. We conclude.
- Case **(deref)**. We have $M = \mathbf{deref}_n(u_{(n, T)})$, $\delta(u_{(n, T)}) = V$, and $M' = V$. From $\vdash_{\Gamma} \delta$ we derive $\vdash_{\Gamma} V : (T, 0)$. We conclude.
- Case **(ref)**. We have $M = \mathbf{ref}_n V$, $M' = u_{(n, T')}$ with $T = T' \mathbf{ref}_n$, $\delta' = \delta\langle V \rightsquigarrow u_{(n, T')} \rangle$. We derive $\vdash_{\Gamma} V : (T', 0)$. We use rule **(Add)** to build $\vdash_{\Gamma} u_{(n, T')} : (T, 0)$. We use rules **(Sto)** to get $\vdash_{\Gamma} \delta\langle V \rightsquigarrow u_{(n, T')} \rangle$.
- Case **(store)** is treated similarly.

□

1.2 Termination of λ_{ref} programs

An important difference w.r.t. the π -calculus case in defining pruning is that we cannot completely remove a subterm the way we do in the case of the π -calculus (using $\mathbf{0}$). Moreover, the pruning function is simpler in the context of the π -calculus, because it is correct to get rid of process $\text{def } f^n = (x).P \text{ in}$ when $n \leq p$: by typing, the subprocess P cannot perform any interaction at level p .

On the contrary, in the present setting, when pruning the term $\text{deref}_n(M)$ with $n \leq p$, we cannot simply ignore M , as M could perform reductions at level p . Indeed, computing an address of level n may involve dereferencing at levels above n .

Let us explain the intuition in the definition of pruning for $\text{deref}_n(M)$ (similar ideas are at work when pruning $\text{ref}_n M$ and $M :=_n M'$). Because, as explained, we cannot just throw away M , the pruning function enters recursively M , in such a way as to remove imperative constructions from M . Pruning therefore transforms $\text{deref}_n(M)$ into a term that first runs the pruned version of M , and then returns a *generic value* of the appropriate type. Generic values are canonical terms that are used to replace a given subterm *once we know that no divergence can arise due to the evaluation of the subterm* (this would correspond either to a divergence of the subterm, or to a contribution to a more general divergence). They are defined as follows:

Definition 7 (Generic value)

The generic value V_T of type T is defined by: $V_{T \text{ ref}_n} = V_{\mathbf{1}} = \star$, and $V_{T_1 \rightarrow^n T_2} = \lambda x.V_{T_2}$ (x being of type T_1).

In order to program the evaluation of a pruned subterm and its replacement with a generic value, the definition of pruning makes use of the following projectors:

$$\Pi^{(1,2)} = \lambda x.\lambda y.x \qquad \Pi^{(1,3)} = \lambda x.\lambda y.\lambda z.x$$

(in the following, we suppose that these terms are always used in a well-typed fashion).

Finally, in order to define the pruning function, we need a last notion, that conveys the intuition that a given term M can be involved in a reduction at level p . This can happen for two reasons. Either M is able to perform (maybe after some preliminary reduction steps) a reduction at level p , in which case, by the typing rules, the effect of M is greater than p , or M is a function that can receive some arguments and eventually perform a reduction at level p , in which case the type system ensures that its type T satisfies $\text{reg}(T) \geq p$.

Definition 8 (Related to p)

Suppose $\vdash_{\Gamma} M : (T, n)$. We say that M is related to p if either $n \geq p$ or $\text{reg}(T) \geq p$. In the former case, we can also say that M is related to p via its effect. In the latter case, via its type.

We extend this notion to evaluation contexts by treating the hole like a term variable, given a typing derivation for a context (this is relevant in particular in the statement of Lemma 18).

Notice that a term containing a subterm whose effect is p is not necessarily related to p : for instance $\vdash_{\Gamma} (\lambda x.\star) \lambda y.\text{deref}_3(u_{(3,\mathbf{1})}) : (\mathbf{1}, 0)$ is not related to 3 (one can easily notice that this term cannot be used to trigger a reduction at level 3).

Definition 9 (Pruning) Given a typable M of type T , we define the pruning at level p of M , written $\text{pr}_{\Gamma}^p(M)$, as follows:

$$\begin{aligned} &\text{If } M \text{ is not related to } p: \\ &\quad \text{pr}_{\Gamma}^p(M) = V_T \\ &\text{Otherwise:} \\ &\quad \text{pr}_{\Gamma}^p(M_1 M_2) = \text{pr}_{\Gamma}^p(M_1) \text{pr}_{\Gamma}^p(M_2) \\ &\quad \text{pr}_{\Gamma}^p(x) = x \\ &\quad \text{pr}_{\Gamma}^p(\lambda x.M_1) = \lambda x.\text{pr}_{\Gamma}^p(M_1) \\ &\quad \text{pr}_{\Gamma}^p(\text{ref}_n M_1) = (\Pi^{(1,2)} \star \text{pr}_{\Gamma}^p(M_1)) \\ &\quad \text{pr}_{\Gamma}^p(\text{deref}_n(M_1)) = (\Pi^{(1,2)} V_T \text{pr}_{\Gamma}^p(M_1)) \\ &\quad \text{pr}_{\Gamma}^p(M_1 :=_n M_2) = (\Pi^{(1,3)} \star \text{pr}_{\Gamma}^p(M_1) \text{pr}_{\Gamma}^p(M_2)) \\ &\quad \text{pr}_{\Gamma}^p(u_{(n,T_1)}) = \star \end{aligned}$$

The definition above is extended to contexts as follows (notice that this pruning is not the one obtained by considering \mathbf{E} as a standard term):

$$\begin{aligned} \text{pr}_\Gamma^p([\]) &= [\] & \text{pr}_\Gamma^p(\mathbf{E} \ M) &= \text{pr}_\Gamma^p(\mathbf{E}) \ \text{pr}_\Gamma^p(M) & \text{pr}_\Gamma^p(V \ \mathbf{E}) &= \text{pr}_\Gamma^p(V) \ \text{pr}_\Gamma^p(\mathbf{E}) \\ \text{pr}_\Gamma^p(\text{deref}_n(\mathbf{E})) &= (\Pi^{(1,2)} \ \mathbf{v}_T \ \text{pr}_\Gamma^p(\mathbf{E})) \text{ if } \text{deref}_n(\mathbf{E}) \text{ has type } T & \text{pr}_\Gamma^p(\text{ref}_n \ \mathbf{E}) &= (\Pi^{(1,2)} \ \star \ \text{pr}_\Gamma^p(\mathbf{E})) \\ \text{pr}_\Gamma^p(\mathbf{E} :=_n M) &= (\Pi^{(1,3)} \ \star \ \text{pr}_\Gamma^p(\mathbf{E}) \ \text{pr}_\Gamma^p(M)) & \text{pr}_\Gamma^p(V :=_n \mathbf{E}) &= (\Pi^{(1,3)} \ \text{pr}_\Gamma^p(V) \ \text{pr}_\Gamma^p(\mathbf{E})) \end{aligned}$$

The target of the pruning is the full (in the sense that no strategy is involved) simply typed λ -calculus (with $\mathbb{1}$ as only base type), as expressed by Lemma 12. The definitions of values, redexes and evaluation contexts for simply-typed λ -calculus are standard.

Definition 10 (Type pruning)

Type pruning is defined by:

$$\text{pr}_\Gamma^p(\mathbb{1}) = \mathbb{1} \quad \text{pr}_\Gamma^p(T \ \text{ref}_n) = \mathbb{1} \quad \text{pr}_\Gamma^p(T_1 \ \rightarrow^n T_2) = \text{pr}_\Gamma^p(T_1) \ \rightarrow \ \text{pr}_\Gamma^p(T_2)$$

It follows immediately from Definition 10 that $\text{pr}_\Gamma^p(T)$ is a simple type.

Fact 11 (Generic value – Simple Typability)

For every type T , \mathbf{v}_T is a simply-typed λ -term of type $\text{pr}_\Gamma^p(T)$.

Proof. Easily done by induction on T . □

The following Lemma will be used to prove that the image of pruning is terminating.

Lemma 12 (Pruning – Typability)

Take $p \in \mathbb{N}$, and suppose $\vdash_\Gamma M : (T, n)$. Then $\text{pr}_\Gamma^p(M)$ is a term of the simply-typed λ -calculus, of type $\text{pr}_\Gamma^p(T)$.

Proof. By induction on the judgment $\vdash_\Gamma M : (T, n)$,

- Either M is not related to p , and $\text{pr}_\Gamma^p(M) = \mathbf{v}_T$. We conclude using Fact 11.
- Or M is related to p ,
 - Case **(Var)**. We have $\text{pr}_\Gamma^p(x) = x$. We consider x as variable of type $\text{pr}_\Gamma^p(T)$.
 - Case **(Add)**. We have $T = T' \ \text{ref}_n$ and $\text{pr}_\Gamma^p(T) = \mathbb{1}$. As $\text{pr}_\Gamma^p(u) = \star$ we conclude.
 - Case **(Abs)**. We have $M = \lambda x.M_1$ and $T = T' \ \rightarrow^n T_1$. We get $\text{pr}_\Gamma^p(T) = \text{pr}_\Gamma^p(T') \ \rightarrow \ \text{pr}_\Gamma^p(T_1)$. We consider x as a variable of type $\text{pr}_\Gamma^p(T')$ in $\text{pr}_\Gamma^p(M) = \lambda x.\text{pr}_\Gamma^p(M_1)$. By induction hypothesis, $\text{pr}_\Gamma^p(M_1)$ is of typz $\text{pr}_\Gamma^p(T_1)$. We conclude.
 - Case **(App)**. We have $M = M_1 \ M_2$ with M_1 of type $T_2 \ \rightarrow^n T_1$ and M_2 of type T_2 . By induction hypothesis, $\text{pr}_\Gamma^p(M_1)$ has type $\text{pr}_\Gamma^p(T_2 \ \rightarrow^n T_1) = \text{pr}_\Gamma^p(T_2) \ \rightarrow \ \text{pr}_\Gamma^p(T_1)$ and $\text{pr}_\Gamma^p(M_2)$ has type $\text{pr}_\Gamma^p(T_2)$. As $\text{pr}_\Gamma^p(M) = \text{pr}_\Gamma^p(M_1) \ \text{pr}_\Gamma^p(M_2)$, we conclude.
 - Case **(Ref)**. We have $M = \text{ref}_n \ M_1$, $T = T_1 \ \text{ref}_n$ and M_1 has type T_1 . The induction hypothesis gives $\text{pr}_\Gamma^p(M_1)$ of type $\text{pr}_\Gamma^p(T_1)$. We get $\text{pr}_\Gamma^p(T) = \mathbb{1}$. We consider $\Pi^{(1,2)}$ has type $\mathbb{1} \ \rightarrow \ \text{pr}_\Gamma^p(T_1) \ \rightarrow \ \mathbb{1}$ in $\text{pr}_\Gamma^p(M) = \Pi^{(1,2)} \ \star \ \text{pr}_\Gamma^p(M_1)$. We conclude.
 - Case **(Drf)**. We have $M = \text{deref}_n(M_1)$, $T_1 = T \ \text{ref}_n$ and M_1 has type T_1 . The induction hypothesis gives $\text{pr}_\Gamma^p(M_1)$ of type $\text{pr}_\Gamma^p(T_1) = \mathbb{1}$. We consider $\Pi^{(1,2)}$ has type $\text{pr}_\Gamma^p(T) \ \rightarrow \ \mathbb{1} \ \rightarrow \ \text{pr}_\Gamma^p(T)$ in $\text{pr}_\Gamma^p(M) = \Pi^{(1,2)} \ \mathbf{v}_T \ \text{pr}_\Gamma^p(M_1)$. We conclude.
 - Case **(Aff)** is similar.

□

Pruning enjoys the following properties:

Fact 13 (Pruning – Evaluation context)

For all \mathbf{E} , $\text{pr}_\Gamma^p(\mathbf{E})$ is an evaluation context.

Proof.

By structural induction on \mathbf{E} , noticing that determinism does not hold for simply-typed λ (if R_1 and R_2 are two redexes, $R_1 R_2$ can be written as $([]) R_2][R_1]$ or $(R_1 [])][R_2]$). \square

Fact 14 (Pruning - Substitution)

For every well-typed term $\lambda x.M_1 V$, we have $\text{pr}_\Gamma^p(M_1\{V/x\}) = \text{pr}_\Gamma^p(M_1)\{\text{pr}_\Gamma^p(V)/x\}$.

Proof. By induction on the typing judgment for M_1 , the interesting case being $M_1 = x$. Two sub-cases can occur:

- Either x is not related to p , which means that $\text{reg}(T') < p$, if T' is the type of x and thus V , whose effect is 0 and whose type is T' is not related to p neither. We have $\text{pr}_\Gamma^p(x) = \mathbf{v}_{T'}$ and $\text{pr}_\Gamma^p(V) = \mathbf{v}_{T'}$. Thus $\text{pr}_\Gamma^p(x)\{\text{pr}_\Gamma^p(V)/x\} = \text{pr}_\Gamma^p(x\{V/x\})$.
- Or x is related to p , which means that $\text{reg}(T') \geq p$, if T' is the type of x and thus V , whose type is T' is related to p too. We have $\text{pr}_\Gamma^p(x)$. Thus $\text{pr}_\Gamma^p(x)\{\text{pr}_\Gamma^p(V)/x\} = x\{\text{pr}_\Gamma^p(V)/x\} = \text{pr}_\Gamma^p(V) = \text{pr}_\Gamma^p(x\{V/x\})$.

\square

We write \rightarrow for the reduction relation in the simply-typed λ -calculus, and \rightarrow^+ (resp. \rightarrow^*) for its transitive (resp. reflexive-transitive) closure. We use these relations to state the simulation properties presented below.

Simulation Result. As in the π -calculus case, the pruning function enjoys simulation properties which allow us to deduce from an infinite computation a divergence involving pruned terms. As the definition of pruning is more involved in the present setting, this result is technically more difficult to obtain.

In order to reason on the transitions of pruned terms, the main point is to understand how pruning interacts with the decomposition of a term into an evaluation context and a redex (Definition 9 is extended to evaluation contexts in a natural way).

The lemma below explains how the pruning function is propagated within a term of the form $\mathbf{E}[M]$.

There are, intuitively, two possibilities, depending only of the context and the level of the pruning: either \mathbf{E} is such that $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$ for all M , that is, pruning is always propagated in the hole to M , or the context is such that, if the effect of M is too small, the pruning inserts a generic value before reaching the hole in \mathbf{E} , in which case $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[V]$, where \mathbf{E}_1 is an ‘initial part’ of \mathbf{E} , and this equality holds independently from M (as long as, like we said, the effect of M is small in some sense).

Note that, in the latter case, if the effect of M is high, the pruning will not stop before reaching the hole of \mathbf{E} .

Lemma 15 (Pruning – Context effect)

If $\vdash_\Gamma \mathbf{E} : (T, m)$, $\Gamma([]) = T_1 \vdash_\Gamma M_1 : (T_1, m_1)$, then $\vdash_\Gamma \mathbf{E}[M_1] : (T, \max(m, m_1))$.

Proof. By structural induction on \mathbf{E} :

- Case $[]$. We have $T = T_1$ and $m = 0$. As $\vdash_\Gamma ([])[M_1] : (T_1, \max(m_1, 0))$ we conclude.
- Case $\mathbf{E}_2 M$. We have $\vdash_\Gamma M : (T', n)$, $\vdash_\Gamma \mathbf{E}_2 : (T' \rightarrow^k T, m_2)$ and $m = \max(m_2, n, k)$. We use the induction hypothesis to get $\vdash_\Gamma \mathbf{E}_2[M_1] : (T' \rightarrow^k T, \max(m_2, m_1))$. We use rule **(App)** to get $\vdash_\Gamma \mathbf{E}_2[M_1] M : (T, \max(\max(m_1, m_2), n, k))$. We conclude, as $\mathbf{E}[M_1] = (\mathbf{E}_2[M_1] M)$ and $\max(\max(m_1, m_2), n, k) = \max(m, m_1)$.
- Case $\text{deref}_n(\mathbf{E}_2)$. We have $\vdash_\Gamma \mathbf{E}_2 : (T \text{ ref}_n, m_2)$ and $m = \max(m_2, n)$. We use the induction hypothesis to get $\vdash_\Gamma \mathbf{E}_2[M_1] : (T \text{ ref}_n, \max(m_2, m_1))$. We use rule **(Drf)** to get $\vdash_\Gamma \text{deref}_n(\mathbf{E}_2[M_1]) : (T, \max(\max(m_2, m_1), n))$. We conclude, as $\mathbf{E}[M_1] = \text{deref}_n(\mathbf{E}_2[M_1])$ and $\max(\max(m_2, m_1), n) = \max(m, m_1)$.

- Other cases are similar.

□

Lemma 16 (Pruning – Context decomposition)

Consider a well-typed context \mathbf{E} and fix a integer p . Then:

1. Either for all well-typed process M , $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$
2. Or there exists \mathbf{E}_1 and $\mathbf{E}_2 \neq []$ s.t. $\mathbf{E} = \mathbf{E}_1[\mathbf{E}_2]$ and, for all M , we are in one of the two following cases:
 - (a) If M has an effect $\geq p$, then $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$.
 - (b) If M has an effect $< p$, then $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$ (where T'' is the type of \mathbf{E}_2).

Proof. By induction on $\vdash_\Gamma \mathbf{E} : (T, m)$:

- Either \mathbf{E} is not related to T , which means $m < p$ and $\text{reg}(T) < p$. We set $\mathbf{E}_1 = []$ and $\mathbf{E}_2 = \mathbf{E}$. We have $\mathbf{E} = \mathbf{E}_1[\mathbf{E}_2]$ and T is the type of \mathbf{E}_2 .
 - Suppose M has an effect $n < p$, then by Lemma 15, $\vdash_\Gamma \mathbf{E}[M] : (T, \max(m, n))$. As $\text{reg}(T) < p$ and $\max(m, n) < p$, $\mathbf{E}[M]$ is not related to p and $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \mathbf{v}_T$. We conclude, as we are in case 2b.
 - Suppose M has an effect $n \geq p$, then by Lemma 15, $\vdash_\Gamma \mathbf{E}[M] : (T, \max(m, n))$. Thus $\mathbf{E}[M]$ is related to p . We discuss on the structure of \mathbf{E} :
 - * If $\mathbf{E} = []$ then $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(M)$. We conclude, as we are in case 2a.
 - * If $\mathbf{E} = (\mathbf{E}_3 M_3)$ we use the induction hypothesis. As M has an effect $\geq p$, we are either in case 1 or in case 2a. In both cases, $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. As $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\text{pr}_\Gamma^p(\mathbf{E}_3[M]) \text{pr}_\Gamma^p(M_3))$ (remember $\mathbf{E}[M]$ is related to p) and $\text{pr}_\Gamma^p(\mathbf{E}) = (\text{pr}_\Gamma^p(\mathbf{E}_3) \text{pr}_\Gamma^p(M_3))$, we conclude, as we are in case 2a.
 - * If $\mathbf{E} = \text{deref}_{n_3}(\mathbf{E}_3)$ we use the induction hypothesis. As M has an effect $\geq p$, we are either in case 1 or in case 2a. In both cases, $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. As $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3[M_3]))$ (remember $\mathbf{E}[M]$ is related to p) and $\text{pr}_\Gamma^p(\mathbf{E}) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3))$, we conclude, as we are in case 2a.
 - * Other cases are similar.
- Or \mathbf{E} is related to p . We discuss on the structure of \mathbf{E} :
 - Case $[]$. Then $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p([])[\text{pr}_\Gamma^p(M)]$. We are in case 1 and we conclude.
 - Case $(\mathbf{E}_3 M_3)$. We use the induction hypothesis on \mathbf{E}_3 .
 - * Either we are in case 1, for all well-typed process M , $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. Thus for all well-typed process M , $\text{pr}_\Gamma^p(\mathbf{E}[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3[M]) \text{pr}_\Gamma^p(M_3)$ (clearly, $\mathbf{E}[M]$ is related to p , as \mathbf{E} is). As, $\text{pr}_\Gamma^p(\mathbf{E}) = (\text{pr}_\Gamma^p(\mathbf{E}_3) \text{pr}_\Gamma^p(M_3))$, we get $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)] \text{pr}_\Gamma^p(M_3)) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$. We are in case 1 and we conclude.
 - * Or we get $\mathbf{E}_{(1)}$ and \mathbf{E}_2 s.t. $\mathbf{E}_3 = \mathbf{E}_{(1)}[\mathbf{E}_2]$ and the corresponding properties. We set $\mathbf{E}_1 = (\mathbf{E}_{(1)} M_3)$. Clearly, $\mathbf{E}_1[\mathbf{E}_2] = \mathbf{E}$.
 - Suppose M has an effect $< p$. Then $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_{(1)})[\mathbf{v}_{T''}]$ where T'' is the type on \mathbf{E}_2 . As $\text{pr}_\Gamma^p(\mathbf{E}_1) = (\text{pr}_\Gamma^p(\mathbf{E}_{(1)}) \text{pr}_\Gamma^p(M_3))$, we have $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\text{pr}_\Gamma^p(\mathbf{E}_3[M]) \text{pr}_\Gamma^p(M_3)) = \text{pr}_\Gamma^p(\mathbf{E}_{(1)})[\mathbf{v}_{T''}] \text{pr}_\Gamma^p(M_3) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$. We are in case 2b and we conclude.
 - Suppose M has an effect $\geq p$. Then $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. We have $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\text{pr}_\Gamma^p(\mathbf{E}_3[M]) \text{pr}_\Gamma^p(M_3)) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$. We are in case 2a and we conclude.
 - Case $\text{deref}_{n_3}(\mathbf{E}_3)$. We use the induction hypothesis on \mathbf{E}_3 .

- * Either we are in case 1, for all well-typed process M , $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. Thus for all well-typed process M , $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3[M]))$ (clearly, $\mathbf{E}[M]$ is related to p , as \mathbf{E} is). As, $\text{pr}_\Gamma^p(\mathbf{E}) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3))$, we get $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$. We are in case 1 and we conclude.
 - * Or we get $\mathbf{E}_{(1)}$ and \mathbf{E}_2 s.t. $\mathbf{E}_3 = \mathbf{E}_{(1)}[\mathbf{E}_2]$ and the corresponding properties. We set $\mathbf{E}_1 = (\Pi^{(1,2)} \mathbf{v}_T \mathbf{E}_{(1)})$. Clearly, $\mathbf{E}_1[\mathbf{E}_2] = \mathbf{E}$.
 - Suppose M has an effect $< p$. Then $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_{(1)}[\mathbf{v}_{T''}])$ where T'' is the type on \mathbf{E}_2 . As $\text{pr}_\Gamma^p(\mathbf{E}_1) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_{(1)}))$, we have $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3[M])) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_{(1)}[\mathbf{v}_{T''}])) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$. We are in case 2b and we conclude.
 - Suppose M has an effect $\geq p$. Then $\text{pr}_\Gamma^p(\mathbf{E}_3[M]) = \text{pr}_\Gamma^p(\mathbf{E}_3)[\text{pr}_\Gamma^p(M)]$. We have $\text{pr}_\Gamma^p(\mathbf{E}[M]) = (\Pi^{(1,2)} \mathbf{v}_T \text{pr}_\Gamma^p(\mathbf{E}_3[M])) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M)]$. We are in case 2a and we conclude.
- Other cases are similar. □

The properties we now establish correspond to the situation, in the previous lemma, where M is an imperative redex acting at level p . By our typing rules, firing the redex yields a term which is not related to p via its effect: depending on the kind of imperative operator that is executed, it might either be related to p via its type, or not related to p at all (this appears more clearly in the proof of Lemma ??).

In such case, we are able to show that the pruned versions of the two terms are related by \rightarrow^+ , which allows us to establish a simulation property.

Fact 17 (Pruning not related to p)

If \mathbf{E}_2 is not related to p :

1. If $\mathbf{E}_2 = (V_3 \mathbf{E}_3)$ then V_3 is not related to p .
2. If $\mathbf{E}_2 = (\mathbf{E}_3 M_3)$ then \mathbf{E}_3 is not related to p .

Proof. Easily done by examining the rule (**App**) and the definition of being related to p . □

Lemma 18 (Pruning – Reduction to a value)

If $\vdash_\Gamma \mathbf{E}_2 : (T'', m)$ and \mathbf{E}_2 is not related to p , for any adequate M, M' ,

1. $\text{pr}_\Gamma^p(\mathbf{E}_2)[(\Pi^{(1,2)} \mathbf{v}_T M)] \rightarrow^+ \mathbf{v}_{T''}$;
2. $\text{pr}_\Gamma^p(\mathbf{E}_2)[(\Pi^{(1,3)} \mathbf{v}_T M M')] \rightarrow^+ \mathbf{v}_{T''}$.

Proof.

1. By structural induction on \mathbf{E} ,

- Case \square . Then $T'' = T$. $(\Pi^{(1,2)} \mathbf{v}_T N) \rightarrow \mathbf{v}_T = \mathbf{v}_{T''}$.
- Case $\mathbf{E}_3 M_3$. Then $\text{pr}_\Gamma^p(\mathbf{E}_2)[(\Pi^{(1,2)} \mathbf{v}_T N)] = (\text{pr}_\Gamma^p(\mathbf{E}_3)[(\Pi^{(1,2)} \mathbf{v}_T N)] \text{pr}_\Gamma^p(M_3))$ with \mathbf{E}_3 of type $T_3 \rightarrow T''$. Thanks to Fact 17, we can use the induction hypothesis on \mathbf{E}_3 and we get $\text{pr}_\Gamma^p(\mathbf{E}_3)[(\Pi^{(1,2)} \mathbf{v}_T N)] \rightarrow^+ \mathbf{v}_{T_3 \rightarrow T''}$. By examining Definition 7, we get $(\mathbf{v}_{T_3 \rightarrow T''} \text{pr}_\Gamma^p(M_3)) \rightarrow \mathbf{v}_{T''}$ and we conclude.
- Case $V_3 \mathbf{E}_3$. Then $\text{pr}_\Gamma^p(\mathbf{E}_2)[(\Pi^{(1,2)} \mathbf{v}_T N)] = (\text{pr}_\Gamma^p(V_3) \text{pr}_\Gamma^p(\mathbf{E}_3)[(\Pi^{(1,2)} \mathbf{v}_T N)])$ with \mathbf{E}_3 of type T_3 . We use Fact 17 to obtain that V_3 is not related to K . Thus $\text{pr}_\Gamma^p(V_3) = \mathbf{v}_{T_3 \rightarrow T''}$. By examining Definition 7, we get $(\mathbf{v}_{T_3 \rightarrow T''} \text{pr}_\Gamma^p(\mathbf{E}_3)[(\Pi^{(1,2)} \mathbf{v}_T N)]) \rightarrow \mathbf{v}_{T''}$ and we conclude.
- Case $\text{deref}_{n_3}(\mathbf{E}_3)$. Then $\text{pr}_\Gamma^p(\mathbf{E}_3)[(\Pi^{(1,2)} \mathbf{v}_T N)] = (\Pi^{(1,2)} \mathbf{v}_{T''} \text{pr}_\Gamma^p(\mathbf{E}_2)[(\Pi^{(1,2)} \mathbf{v}_T N)]) \rightarrow \mathbf{v}_{T''}$.
- Other cases are similar.

2. This case is very similar.

□

Lemmas 16 and 18 allow us to derive the desired simulation property for λ_{ref} ; in particular, the main point is that a \mapsto_p reduction is pruned into at least one reduction in the target calculus (case 4 below). Note however that we do not rely on behavioural equivalences here.

Lemma 19 (Simulation) *Consider $p \in \mathbb{N}$ and $\vdash_{\Gamma} M : (T, m)$.*

1. *If $(M, \delta) \mapsto_{\Gamma}^p (M', \delta')$ and $n < p$, then $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$.*
2. *If $(M, \delta) \mapsto_{\Gamma}^p (M', \delta')$, then $\text{pr}_{\Gamma}^p(M) \rightarrow^+ \text{pr}_{\Gamma}^p(M')$.*
3. *If $(M, \delta) \mapsto_{\Gamma}^p (M', \delta')$ and $n < p$, then $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$.*
4. *If $(M, \delta) \mapsto_{\Gamma}^p (M', \delta')$, then $\text{pr}_{\Gamma}^p(M) \rightarrow \text{pr}_{\Gamma}^p(M')$.*

Proof.

1. *By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (**deref**), (**store**) or (**ref**). Thus $M = \mathbf{E}[R]$ with R being a redex of type T' of store operation, dereferencing or reference.*

- *Case (**ref**). We have $M = \mathbf{E}[\text{ref}_n V]$, $M = \mathbf{E}[u_{(n, T_0)}]$ and $\delta' = \delta \langle u_{(n, T_0)} \rightsquigarrow V \rangle$ with $T' = T_0 \text{ ref}_n$. As we supposed that types are well-formed and $n < p$, $T = T_0 \text{ ref}_n$ is not related to p . Moreover, the effect of $\text{ref}_n V$ is $n < p$ and the effect of $u_{(n, T_0)}$ is 0. We use Lemma 16 on \mathbf{E} :*

– *Either we are in case 1. We get $\text{pr}_{\Gamma}^p(\mathbf{E}[\text{ref}_n V]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\text{pr}_{\Gamma}^p(\text{ref}_n V)] = \text{pr}_{\Gamma}^p(\mathbf{E})[\mathbf{V}_{T_0 \text{ ref}_n}] = \text{pr}_{\Gamma}^p(\mathbf{E})[\star]$ and $\text{pr}_{\Gamma}^p(\mathbf{E}[u_{(n, T_0)}]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\star]$. As $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$, we conclude.*

– *Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 and their corresponding properties. Both terms M and M' correspond to case 2b and we get $\text{pr}_{\Gamma}^p(\mathbf{E}[\text{ref}_n V]) = \text{pr}_{\Gamma}^p(\mathbf{E}_1)[\mathbf{V}_{T''}]$ and $\text{pr}_{\Gamma}^p(\mathbf{E}[u_{(n, T_0)}]) = \text{pr}_{\Gamma}^p(\mathbf{E}_1)[\mathbf{V}_{T''}]$. As $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$, we conclude.*

- *Case (**deref**). We have $M = \mathbf{E}[\text{deref}_n(u_{(n, T')})]$ and $M' = \mathbf{E}[V]$ with $\delta(u_{(n, T)}) = V$. As we supposed that types are well-formed and $n < p$, $T' \text{ ref}_n$ is not related to p . Moreover, the effect of $\text{deref}_n(u_{(n, T')})$ is $n < p$ and the effect of V is 0. We use Lemma 16 on \mathbf{E} :*

– *Either we are in case 1. We get $\text{pr}_{\Gamma}^p(\mathbf{E}[\text{deref}_n(u_{(n, T')})]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\text{pr}_{\Gamma}^p(\text{deref}_n(u_{(n, T')}))] = \text{pr}_{\Gamma}^p(\mathbf{E})[\mathbf{V}_{T'}]$ and $\text{pr}_{\Gamma}^p(\mathbf{E}[V]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\text{pr}_{\Gamma}^p(V)] = \text{pr}_{\Gamma}^p(\text{conte})[\mathbf{V}_{T'}]$. As $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$, we conclude.*

– *Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 and their corresponding properties. Both terms M and M' correspond to case 2b and we get $\text{pr}_{\Gamma}^p(\mathbf{E}[\text{deref}_n(u_{(n, T')})]) = \text{pr}_{\Gamma}^p(\mathbf{E}_1)[\mathbf{V}_{T''}]$ and $\text{pr}_{\Gamma}^p(\mathbf{E}[V]) = \text{pr}_{\Gamma}^p(\mathbf{E}_1)[\mathbf{V}_{T''}]$. As $\text{pr}_{\Gamma}^p(M) = \text{pr}_{\Gamma}^p(M')$, we conclude.*

- *Case (**store**) is similar.*

2. *By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (**deref**), (**store**) or (**ref**). Thus $M = \mathbf{E}[R]$ with R of type T' being a redex of store operation, dereferencing or reference.*

- *Case (**deref**). We have $M = \mathbf{E}[\text{deref}_p(u_{(p, T')})]$ and $M' = \mathbf{E}[V]$ with $\delta(u_{(p, T')}) = V$. We remark that $\text{deref}_p(u_{(p, T')})$ has an effect $\geq p$ but V (whose effect is 0) has not. Moreover, as type $T' \text{ ref}_p$ is well-formed, V is not related to p . We use Lemma 16 on \mathbf{E} :*

– *Either we are in case 1. We get $\text{pr}_{\Gamma}^p(\mathbf{E}[\text{deref}_p(u_{(p, T')})]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\text{pr}_{\Gamma}^p(\text{deref}_p(u_{(p, T')}))] = \text{pr}_{\Gamma}^p(\mathbf{E})[\Pi^{(1,2)} \mathbf{V}_{T'} \text{ pr}_{\Gamma}^p(V)]$ and $\text{pr}_{\Gamma}^p(\mathbf{E}[V]) = \text{pr}_{\Gamma}^p(\mathbf{E})[\text{pr}_{\Gamma}^p(V)] = \text{pr}_{\Gamma}^p(\mathbf{E})[\mathbf{V}_{T'}]$. We use Fact 13 to get $\text{pr}_{\Gamma}^p(M) \rightarrow \text{pr}_{\Gamma}^p(M')$ and conclude.*

- Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 s.t. (i) $\mathbf{E} = \mathbf{E}_1[\mathbf{E}_2]$ and (ii) \mathbf{E}_2 is not related to p . Term M corresponds to case 2a and term M' corresponds to case 2b: we get $\text{pr}_\Gamma^p(\mathbf{E}[\text{deref}_p(u_{(p,T')})]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\text{deref}_p(u_{(p,T')})p)] = \text{pr}_\Gamma^p(\mathbf{E}_1)[\text{pr}_\Gamma^p(\mathbf{E}_2)[\Pi^{(1,2)} \mathbf{v}_{T'} \star]]$ (using (i)) and $\text{pr}_\Gamma^p(\mathbf{E}[V]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$ (using (ii)). We use Fact 13 and Lemma 18 to get $\text{pr}_\Gamma^p(M) \rightarrow^* \text{pr}_\Gamma^p(M')$ and we conclude.
 - Case (**ref**). We have $M = \mathbf{E}[\text{ref}_p V]$ and $M' = \mathbf{E}[u_{(p,T_0)}]$ with $T' = T_0 \text{ ref}_p$. Term $\text{ref}_p V$ has effect p (and thus, is related to p) but $u_{(p,T_0)}$ (whose effect is 0) has not. Moreover, $u_{(p,T_0)}$, whose type is $T_0 \text{ ref}_p$, is related to p . We use Lemma 16 on \mathbf{E} :
 - Either we are in case 1. We get $\text{pr}_\Gamma^p(\mathbf{E}[\text{ref}_p V]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\text{ref}_p p)] = \text{pr}_\Gamma^p(\mathbf{E})[\Pi^{(1,2)} \star \text{pr}_\Gamma^p(V)]$ and $\text{pr}_\Gamma^p(\mathbf{E}[u_{(p,T_0)}]) = \text{pr}_\Gamma^p(\mathbf{E})[\star]$. We use Fact 13 to get $\text{pr}_\Gamma^p(M) \rightarrow \text{pr}_\Gamma^p(M')$ and conclude.
 - Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 s.t. (i) $\mathbf{E} = \mathbf{E}_1[\mathbf{E}_2]$ and (ii) \mathbf{E}_2 is not related to p . Term M corresponds to case 2a and term M' corresponds to case 2b: we get $\text{pr}_\Gamma^p(\mathbf{E}[\text{ref}_p V]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\text{ref}_p p)] = \text{pr}_\Gamma^p(\mathbf{E}_1)[\text{pr}_\Gamma^p(\mathbf{E}_2)[\Pi^{(1,2)} \star \text{pr}_\Gamma^p(V)]]$ (using (i)) and $\text{pr}_\Gamma^p(\mathbf{E}[V]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$ (using (ii)). We use Fact 13 and Lemma 18 to get $\text{pr}_\Gamma^p(M) \rightarrow^* \text{pr}_\Gamma^p(M')$ and we conclude.
 - Case (**store**) is similar.
3. By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (β). Thus $M = \mathbf{E}[\lambda x.M_1 V]$, $M' = \mathbf{E}[M_1\{V/x\}]$. The term $\lambda x.M_1$ has effect 0 and type $T_2 \rightarrow^n T'$ (for some T_2 and $n < p$) and the term V has type T_2 and effect 0; by rule *App*, $\lambda x.M_1 V$ has effect $n < p$, by rule (**Abs**), M_1 has effect $n < K$ and by Lemma 5, $M_1\{V/x\}$ has effect $n < K$. Thus, as type $T_2 \rightarrow^n T'$ is well-formed, neither $\lambda x.M_1 V$, nor $M_1\{V/x\}$ are related to K (notice that V could). We use Lemma 16 on \mathbf{E} :
- Either we are in case 1. We get $\text{pr}_\Gamma^p(\mathbf{E}[\lambda x.M_1 V]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\lambda x.M_1 V)] = \text{pr}_\Gamma^p(\mathbf{E})[\mathbf{v}_{T'}]$ and $\text{pr}_\Gamma^p(\mathbf{E}[M_1\{V/x\}]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M_1\{V/x\})] = \text{pr}_\Gamma^p(\mathbf{E})[\mathbf{v}_{T'}]$. We have $\text{pr}_\Gamma^p(M) = \text{pr}_\Gamma^p(M')$ and we conclude.
 - Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 and their corresponding properties. Both terms M and M' correspond to case 2b and we get $\text{pr}_\Gamma^p(\mathbf{E}[\lambda x.M_1 V]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$ and $\text{pr}_\Gamma^p(\mathbf{E}[M_1\{V/x\}]) = \text{pr}_\Gamma^p(\mathbf{E}_1)[\mathbf{v}_{T''}]$. As $\text{pr}_\Gamma^p(M) = \text{pr}_\Gamma^p(M')$, we conclude.
4. By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (β). Thus $M = \mathbf{E}[\lambda x.M_1 V]$, $M' = \mathbf{E}[M_1\{V/x\}]$. The term $\lambda x.M_1$ has effect 0 and type $T_2 \rightarrow^p T'$ (for some T_2) and the term V has type T_2 and effect 0. By rule *App*, $\lambda x : T_2.M_1 V$ has effect p (and thus is related to p), by rule (**Abs**), M_1 has effect p and by Lemma 5, $M_1\{V/x\}$ has effect p (and thus is related to p). We use Lemma 16:
- Either we are in case 1. We get $\text{pr}_\Gamma^p(\mathbf{E}[\lambda x.M_1 V]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\lambda x.M_1 V)] = \text{pr}_\Gamma^p(\mathbf{E})[\lambda x.\text{pr}_\Gamma^p(M_1) \text{pr}_\Gamma^p(V)]$ and $\text{pr}_\Gamma^p(\mathbf{E}[M_1\{V/x\}]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M_1\{V/x\})]$. We use Fact 14 and Fact 13 to get have $\text{pr}_\Gamma^p(M) \rightarrow \text{pr}_\Gamma^p(M')$ and we conclude.
 - Or we are in case 2. We get \mathbf{E}_1 and \mathbf{E}_2 and their corresponding properties. Both terms M and M' correspond to case 2a and we get $\text{pr}_\Gamma^p(\mathbf{E}[\lambda x.M_1 V]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(\lambda x.M_1 V)] = \text{pr}_\Gamma^p(\mathbf{E})[\lambda x.\text{pr}_\Gamma^p(M_1) \text{pr}_\Gamma^p(V)]$ and $\text{pr}_\Gamma^p(\mathbf{E}[M_1\{V/x\}]) = \text{pr}_\Gamma^p(\mathbf{E})[\text{pr}_\Gamma^p(M_1\{V/x\})]$. We use Fact 14 and Fact 13 to get have $\text{pr}_\Gamma^p(M) \rightarrow \text{pr}_\Gamma^p(M')$ and we conclude.

□

We rely on termination of the simply-typed λ -calculus (written **ST**) to obtain the soundness of our system.

Theorem 20 (Termination ST)

If $M \in \mathbf{ST}$, M terminates.

Proof. See [GTL89]. □

Fact 21 (Pruning image)

If $\vdash_{\Gamma} M : (T, m)$, then $\text{pr}_{\Gamma}^p(M) \in \mathbf{ST}$.

Proof. Easily done by induction on the typing judgment. □

Definition 22 (Active imperative operators) The number of active imperative operators of level p in M is defined on typed process as follows:

$$\begin{aligned} \mathbf{Os}^p(x) &= \mathbf{Os}^p(\lambda x.M) = \mathbf{Os}^p(u_{(n,T)}) = 0 & \mathbf{Os}^p(M N) &= \mathbf{Os}^p(M) + \mathbf{Os}^p(N) \\ \mathbf{Os}^p(\text{deref}_n(M)) &= \mathbf{Os}^p(\text{ref}_n M) = \mathbf{Os}^p(M) & \text{if } n \neq p \\ \mathbf{Os}^p(\text{deref}_p(M)) &= \mathbf{Os}^p(\text{ref}_p M) = 1 + \mathbf{Os}^p(M) \\ \mathbf{Os}^p(M :=_n N) &= \mathbf{Os}^p(M) + \mathbf{Os}^p(N) & \text{if } n \neq p \\ \mathbf{Os}^p(M :=_p N) &= 1 + \mathbf{Os}^p(M) + \mathbf{Os}^p(N) \end{aligned}$$

We extend this definition to context by $\mathbf{Os}^p([\]) = 0$.

Fact 23

We have $\mathbf{Os}^p(\mathbf{E}[M]) = \mathbf{Os}^p(\mathbf{E}) + \mathbf{Os}^p(M)$

Proof.

By structural induction on contexts. □

Lemma 24 (Effect and active imperative operators)

If $\vdash_{\Gamma} M : (T, m)$ and $m < p$ then $\mathbf{Os}^p(M) = 0$.

Proof. By induction on the typing judgment $\vdash_{\Gamma} M : (T, m)$,

- Case **(App)**. We have $M = M_1 M_2$ with M_1 of type $m_1 \leq m < p$ and M_2 of type $m_2 \leq m < p$. Thus, we are able to use the induction hypothesis to get $\mathbf{Os}^p(M) = \mathbf{Os}^p(M_1) + \mathbf{Os}^p(M_2) = 0 + 0$.
- Case **(Ref)**. We have $M = \text{ref}_{m_1} M_1$. Typing gives $m_1 \leq m < p$. We use the induction hypothesis and conclude, as $\mathbf{Os}^p(M) = \mathbf{Os}^p(M_1) = 0$.
- Cases **(Asg)** and **(Drf)** are similar.
- Cases **(Add)**, **(Var)**, **(Uni)** and **(Abs)** are easy, as every value V has effect 0 and $\mathbf{Os}^K(V) = 0$. □

Lemma 25 (Active imperative operators decreasing)

If $\vdash_{\Gamma} M : (T, m)$ then

1. if $(M, \delta) \mapsto_{\mathbb{F}}^n (M', \delta')$ with $n < p$ then $\mathbf{Os}^p(M') \leq \mathbf{Os}^p(M)$.
2. if $(M, \delta) \mapsto_{\mathbb{I}}^n (M', \delta')$ with $n < p$ then $\mathbf{Os}^p(M') \leq \mathbf{Os}^p(M)$.
3. if $(M, \delta) \mapsto_{\mathbb{I}}^p (M', \delta')$ then $\mathbf{Os}^p(M') < \mathbf{Os}^p(M)$.

Proof.

1. By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (β). Thus $M = \mathbf{E}[\lambda x.M_1 V]$, $M' = \mathbf{E}[M_1\{V/x\}]$. The term $\lambda x.M_1$ has effect 0 and type $T_2 \rightarrow^n T'$ (for some T_2 and $n < p$) and the term V has type T_2 and effect 0; by rule **App**, $\lambda x.M_1 V$ has effect $n < p$, by rule (**Abs**), M_1 has effect $n < K$ and by Lemma 5, $M_1\{V/x\}$ has effect $n < K$. We use Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M) = \mathbf{Os}^p(\mathbf{E}) + 0$ and we use again Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M') = \mathbf{Os}^p(\mathbf{E}) + 0$. We conclude.
2. By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (**deref**), (**store**) or (**ref**). Thus $M = \mathbf{E}[R]$ with R being a redex of type T' of store operation, dereferencing or reference.
 - Case (**ref**). We have $M = \mathbf{E}[\mathbf{ref}_n V]$, $M = \mathbf{E}[u_{(n,T')}]$ and $\delta' = \delta\langle u_{(n,T_0)} \rightsquigarrow V \rangle$ with $T' = T_0 \mathbf{ref}_n$. The effect of $\mathbf{ref}_n V$ is $n < p$ and the effect of $u_{(n,T_0)}$ is 0. We use Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M) = \mathbf{Os}^p(\mathbf{E}) + 0$ and we use again Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M') = \mathbf{Os}^p(\mathbf{E}) + 0$. We conclude.
 - Case (**deref**). We have $M = \mathbf{E}[\mathbf{deref}_n(u_{(n,T')})]$ and $M' = \mathbf{E}[V]$ with $\delta(u_{(n,T)}) = V$. The effect of $\mathbf{deref}_n(u_{(n,T')})$ is $n < p$ and the effect of V is 0. We use Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M) = \mathbf{Os}^p(\mathbf{E}) + 0$ and we use again Lemma 24 and Fact 23 to get $\mathbf{Os}^p(M') = \mathbf{Os}^p(\mathbf{E}) + 0$. We conclude.
 - Case (**store**) is similar.
3. By definition of the semantics, this reduction derivation is composed by an application of rule (**context**) and an application of rule (**deref**), (**store**) or (**ref**). Thus $M = \mathbf{E}[R]$ with R of type T' being a redex of store operation, dereferencing or reference.
 - Case (**deref**). We have $M = \mathbf{E}[\mathbf{deref}_p(u_{(p,T')})]$ and $M' = \mathbf{E}[V]$ with $\delta(u_{(p,T')}) = V$. We remark that $\mathbf{deref}_p(u_{(p,T')})$ has an effect $\geq p$ but V (whose effect is 0) has not. By definition $\mathbf{Os}^p(\mathbf{deref}_p(u_{(p,T')})) = 1 + 0$ and by Lemma 24 $\mathbf{Os}^p(V) = 0$. We use twice Fact 23 to conclude $\mathbf{Os}^p(M) = \mathbf{Os}^p(\mathbf{E}) + 1 < \mathbf{Os}^p(M') = \mathbf{Os}^p(\mathbf{E}) + 0$.
 - Case (**ref**). We have $M = \mathbf{E}[\mathbf{ref}_p V]$ and $M' = \mathbf{E}[u_{(p,T_0)}]$ with $T' = T_0 \mathbf{ref}_p$. Term $\mathbf{ref}_p V$ has effect p (and thus, is related to p) but $u_{(p,T_0)}$ (whose effect is 0) has not. By definition $\mathbf{Os}^p(\mathbf{ref}_p V) = 1 + 0$ and by Lemma 24 $\mathbf{Os}^p(u_{(p,T_0)}) = 0$. We use twice Fact 23 to conclude $\mathbf{Os}^p(M) = \mathbf{Os}^p(\mathbf{E}) + 1 < \mathbf{Os}^p(M') = \mathbf{Os}^p(\mathbf{E}) + 0$.
 - Case (**store**) is similar.

□

Lemma 26 (Maximum Interesting Level) *Suppose that $\vdash_{\Gamma} M : (T, l)$, and that there exists $(M_i)_{i \in \mathbb{N}}$, an infinite reduction sequence starting from M . Then*

1. for all i , M_i is typable.
2. There exists p (called the maximum interesting level of the sequence) and i_o s.t.
 - (a) if $i > i_o$ and $(M_i, \delta_i) \mapsto_1^n (M_{i+1}, \delta_{i+1})$ then $n \leq p$.
 - (b) if $i > i_o$ and $(M_i, \delta_i) \mapsto_{\mathbb{F}}^n (M_{i+1}, \delta_{i+1})$ then $n \leq p$.
 - (c) there exists an infinite set of indexes \mathcal{I} s.t. for each $i \in \mathcal{I}$ either $(M_i, \delta_i) \mapsto_{\mathbb{F}}^p (M_{i+1}, \delta_{i+1})$ or $(M_i, \delta_i) \mapsto_1^p (M_{i+1}, \delta_{i+1})$.
 - (d) There are infinitely many $i \in \mathcal{I}$ s.t. $(M_i, \delta_i) \mapsto_{\mathbb{F}}^p (M_{i+1}, \delta_{i+1})$.

Proof.

1. Follows by Proposition 6.

2. It is easy to find p satisfying 2a, 2b and 2c, as the set of levels on which an infinite number of reductions take place is finite and thus, admits a maximum. Lemma 25 ensures that 2d holds. Consider such a p and suppose, toward a contradiction, that 2d does not hold, that is, there exists an index j s.t. for every $i > j$, either $(M_i, \delta_i) \mapsto_{\mathbb{F}}^n (M_{i+1}, \delta_{i+1})$ with $n < p$, or $(M_i, \delta_i) \mapsto_{\mathbb{I}}^n (M_{i+1}, \delta_{i+1})$ with $n < p$, or $(M_i, \delta_i) \mapsto_{\mathbb{I}}^p (M_{i+1}, \delta_{i+1})$. We use Lemma 25 to state that the sequence $(\mathbf{Os}^p(M_i))_{i>j}$ is decreasing. Moreover, as ?? holds, there is an infinite number of i s.t. $M_i \rightarrow_{\mathbb{I}}^p M_{i+1}$. Thus, we use Lemma 25 to state that the sequence $(\mathbf{Os}^p(M_i))_{i>j}$ strictly decreases an infinite number of times. Contradiction, as $>$ is well-founded. \square

It may be noticed that in this call-by-value setting, the well-formedness of types is not used here. (It is used in Lemma 19 and it is required to apply Lemma 16)

We put together Lemmas 19 and 26 to prove soundness.

Theorem 27 (Soundness) *If $\vdash_{\Gamma} M : (T, m)$ then M terminates.*

Proof. Consider, by absurd, an infinite computation $\{M_i\}_i$ starting from $M = M_0$. By Lemma 26, all the M_i 's are well-typed, and there is a maximal p s.t. for infinitely many i , $(M_i, \delta_i) \mapsto_{\mathbb{F}}^p (M_{i+1}, \delta_{i+1})$ and an index i_0 s.t. every reduction on an index greater than i_0 is performed at level $n \leq p$. Consider the sequence $(\text{pr}_{\Gamma}^p(M_i))_{i>i_0}$. By Lemma 19, we obtain that for every $i > i_0$, either $\text{pr}_{\Gamma}^p(M_i) \rightarrow^* \text{pr}_{\Gamma}^p(M_{i+1})$. Moreover, $\text{pr}_{\Gamma}^p(M_i) \rightarrow^+ \text{pr}_{\Gamma}^p(M_{i+1})$ for an infinite number of i . Thus $\text{pr}_{\Gamma}^p(M_{i_0})$ is congruent to a diverging process, thus diverging. This contradicts Theorem 20 and Fact 21. \square

Parametricity and extensions As in the π -calculus, so in the λ -calculus the method is parametric with respect to a terminating pure functional core and its termination proof. For instance, the core functional language could be System T [GLT88]. This parametrisation can be formalised and made explicit along the lines of what we did for the π -calculus.

Moreover, as in the π -calculus, the method could be extended and refined. A natural extension is to consider more sophisticated type systems, beyond simple types, for instance with polymorphism, subject to the constraints on levels discussed for π -calculus.

Other cores could be considered As the simply typed discipline is enforced by our type system, the functional calculus has to be a subset of the simply typed terms. We believe that it is possible to extend our work to polymorphic types, although this extension is not trivial, as it might involve polymorphism at the level of regions: for instance, we have to check that a type like $(\forall A. A \rightarrow^0 A) \text{ref}_n$ cannot have its A component instantiated with a type containing a level strictly greater than n .

Remark 28 (If-then-else constructs) *Some remarks can be made regarding the treatment of conditional branching using our pruning technique. More precisely, when pruning $\text{if deref}_n(u_{(n,T)}) \text{ then } M_1 \text{ else } M_2$ (whose counterpart, in the π -calculus case, could be $\text{if } a = x \text{ then } P_1 \text{ else } P_2$), we forget the potential value of $\text{deref}_n(u_{(n,T)})$ (resp. the names a and x are forgotten by the pruning). Thus it is legitimate to ask if the pruned term can “be mistaken” and choose the wrong branch, breaking the simulation property.*

The answer is given by the encoding of the conditional in λ : $[\text{if } B \text{ then } M \text{ else } N] = (B \lambda x. M \lambda x. N) \star$ with the encodings for values: $[\text{true}] = \lambda xy. x$ and $[\text{false}] = \lambda xy. y$. Adding abstractions and \star prevents the Call-by-Value strategy from reducing the branch of the conditional before the condition. The reasoning which rejects $\text{if deref}_n(u_{(n,T)}) \text{ then } M_1 \text{ else } M_2$ as a counter-example is as follows: if one of the branch is divergent, following the conditions of the proof, this branch will lead to an infinite number of reduction of level p . Thus the condition, which is a term manipulating the branches, has to be a function of at least level p , thus it cannot be stored at level strictly smaller than $p+1$. As a consequence, when we take as a starting point the index in the reduction sequence where there is no longer reduction on level greater than p , we place ourselves beyond the point where the if then else has been reduced.

Acknowledgments. We wish to thank G. Boudol for fruitful discussions in the initial stages of this work.

References

- [Ama09] R. Amadio. On Stratified Regions. In *Proc. of APLAS*, volume 5904 of *LNCS*, pages 210–225. Springer, 2009.
- [Bou07] G. Boudol. Fair Cooperative Multithreading. In *Proc. of CONCUR*, volume 4703 of *LNCS*, pages 272–286. Springer, 2007.
- [GLT88] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.
- [GTL89] J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and types*. Cambridge University Press, 1989.