Call-by-need, neededness and all that*

Delia Kesner[1], Alejandro Ríos[2], and Andrés Viso[3]

[1] IRIF, CNRS and Univ. Paris-Diderot
[2] Universidad de Buenos Aires
[3] CONICET and Universidad de Buenos Aires

Abstract. We show that call-by-need is observationally equivalent to weak-head needed reduction. The proof of this result uses a semantical argument based on a (non-idempotent) intersection type system called $\mathcal{V}$. Interestingly, system $\mathcal{V}$ also allows to syntactically identify all the weak-head needed redexes of a term.

1 Introduction

One of the fundamental notions underlying this paper is the one of *needed reduction* in λ-calculus, which is to be used here to understand (lazy) evaluation of functional programs. Key notions are those of reducible and non-reducible programs: the former are programs (represented by λ-terms) containing non-evaluated subprograms, called reducible expressions (redexes), whereas the latter can be seen as definitive results of computations, called normal forms. It turns out that every reducible program contains a special kind of redex known as needed or, in other words, every λ-term not in normal form contains a needed redex. A redex r is said to be *needed* in a λ-term t if r has to be contracted (*i.e.* evaluated) sooner or later when reducing t to *normal form*, or, informally said, if there is no way of avoiding r to reach a normal form.

The needed strategy, which always contracts a needed redex, is normalising [8], *i.e.* if a term can be reduced (in any way) to a normal form, then contraction of needed redexes necessarily terminates. This is an excellent starting point to design an evaluation strategy, but unfortunately, neededness of a redex is not decidable [8]. As a consequence, real implementations of functional languages cannot be directly based on this notion.

Our goal is, however, to establish a clear connection between the semantical notion of neededness and different implementations of lazy functional languages (*e.g.* Miranda or Haskell). Such implementations are based on *call-by-need calculi*, pioneered by Wadsworth [20], and extensively studied *e.g.* in [3]. Indeed, call-by-need calculi fill the gap between the well-known operational semantics of the call-by-name λ-calculus and the actual implementations of lazy functional languages. While call-by-name re-evaluates an argument each time it is used –an operation which is quite expensive– call-by-need can be seen as a *memoized* version of call-by-name, where the value of an argument is stored the first time it is

* This work was partially founded by LIA INFINIS.

evaluated for subsequent uses. For example, if $t = \Delta\,(I\,I)$, where $\Delta = \lambda x.x\,x$ and $I = \lambda z.z$, then call-by-name duplicates the argument $I\,I$, while lazy languages first reduce $I\,I$ to the value I so that further uses of this argument do not need to evaluate it again.

While the notion of needed reduction is defined with respect to (full strong) *normal forms*, call-by-need calculi evaluate programs to special values called *weak-head normal forms*, which are either abstractions or arbitrary applications headed by a variable (*i.e.* terms of the form $x\,t_1 \ldots t_n$ where $t_1 \ldots t_n$ are arbitrary terms). To overcome this shortfall, we first adapt the notion of needed redex to terms that are not going to be fully reduced to *normal forms* but only to *weak-head normal forms*. Thus, informally, a redex r is *weak-head needed* in a term t if r has to be contracted sooner or later when reducing t to a weak-head normal form. The derived notion of strategy is called a *weak-head needed strategy*, which always contracts a weak-head needed redex.

This paper introduces two independent results about weak-head neededness, both obtained by means of (non-idempotent) intersection types [13,12] (a survey can be found in [9]). We consider, in particular, typing system $\mathcal{V}$ [14] and show that it allows to identify all the weak-head needed redexes of a weak-head normalising term. This is done by adapting the classical notion of *principal type* [17] and proving that a redex in a weak-head normalising term t is weak-head needed iff it is typed in a principally typed derivation for t in $\mathcal{V}$.

Our second goal is to show observational equivalence between call-by-need and weak-head needed reduction. Two terms are observationally equivalent when all the empirically testable computations on them are identical. This means that a term t can be evaluated to a weak-head normal form using the call-by-need machinery if and only if the weak-head needed reduction normalises t.

By means of system $\mathcal{V}$ mentioned so far we use a technique to reason about observational equivalence that is flexible, general and easy to verify or even certify. Indeed, system $\mathcal{V}$ provides a semantic argument: first showing that a term t is typable in system $\mathcal{V}$ iff it is normalising for the weak-head needed strategy ($t \in \mathcal{WN}_{\mathtt{whnd}}$), then by resorting to some results in [14], showing that system $\mathcal{V}$ is complete for call-by-name, *i.e.* a term t is typable in system $\mathcal{V}$ iff t is normalising for call-by-name ($t \in \mathcal{WN}_{\mathtt{name}}$); and that t is normalising for call-by-name iff t is normalising for call-by-need ($t \in \mathcal{WN}_{\mathtt{need}}$). Thus completing the following chain of equivalences:

$$t \in \mathcal{WN}_{\mathtt{whnd}} \iff t \text{ typable in } \mathcal{V} \iff t \in \mathcal{WN}_{\mathtt{name}} \iff t \in \mathcal{WN}_{\mathtt{need}}$$

This leads to the observational equivalence between call-by-need, call-by-name and weak-head needed reduction.

Structure of the paper: Sec. 2 introduces preliminary concepts while Sec. 3 defines different notions of needed reduction. The type system $\mathcal{V}$ is studied in Sec. 4. Sec 5 extends β-reduction to derivation trees. We show in Sec. 6 how system $\mathcal{V}$ identifies weak-head needed redexes, while Sec. 7 gives a characterisation of normalisation for the weak-head needed reduction. Sec. 8 is devoted to define call-by-need. Finally, Sec. 9 presents the observational equivalence result.

2 Preliminaries

This section introduces some standard definitions and notions concerning the reduction strategies studied in this paper, that is, call-by-name, head and weak-head reduction, and neededness, this later notion being based on the *theory of residuals* [7].

2.1 The Call-By-Name Lambda-Calculus

Given a countable infinite set $\mathcal{X}$ of variables $x, y, z, \ldots$ we consider the following grammar:

$$\begin{array}{rl} \textbf{(Terms)} & t, u ::= x \in \mathcal{X} \mid t\,u \mid \lambda x.t \\ \textbf{(Values)} & v ::= \lambda x.t \\ \textbf{(Contexts)} & \mathtt{C} ::= \Box \mid \mathtt{C}\,t \mid t\,\mathtt{C} \mid \lambda x.\mathtt{C} \\ \textbf{(Name contexts)} & \mathtt{E} ::= \Box \mid \mathtt{E}\,t \end{array}$$

The set of λ-terms is denoted by $\mathcal{T}_{\mathtt{a}}$. We use I, K and Ω to denote the terms $\lambda x.x$, $\lambda x.\lambda y.x$ and $(\lambda x.x\,x)\,(\lambda x.x\,x)$ respectively. We use $\mathtt{C}\langle t\rangle$ (resp. $\mathtt{E}\langle t\rangle$) for the term obtained by replacing the hole $\Box$ of $\mathtt{C}$ (resp. $\mathtt{E}$) by t. The sets of ***free*** and ***bound variables*** of a term t, written respectively $\mathtt{fv}(t)$ and $\mathtt{bv}(t)$, are defined as usual [7]. We work with the standard notion of ***α-conversion***, *i.e.* renaming of bound variables for abstractions; thus for example $\lambda x.x\,y =_\alpha \lambda z.z\,y$.

A term of the form $(\lambda x.t)\,u$ is called a ***β-redex*** (or just ***redex*** when β is clear from the context) and λx is called the ***anchor*** of the redex. The ***one-step reduction relation*** $\to_\beta$ (resp. $\to_{\mathtt{name}}$) is given by the closure by contexts $\mathtt{C}$ (resp. $\mathtt{E}$) of the rewriting rule $(\lambda x.t)\,u \mapsto_\beta t\{x/u\}$, where $\_\{\_/\_\}$ denotes the capture-free standard higher-order substitution. Thus, call-by-name forbids reduction inside arguments and λ-abstractions, *e.g.* $(\lambda x.II)\,(II) \to_\beta (\lambda x.II)\,I$ and $(\lambda x.II)\,(II) \to_\beta (\lambda x.I)\,(II)$ but neither $(\lambda x.II)\,(II) \to_{\mathtt{name}} (\lambda x.II)\,I$ nor $(\lambda x.II)\,(II) \to_{\mathtt{name}} (\lambda x.I)\,(II)$ holds. We write $\twoheadrightarrow_\beta$ (resp. $\twoheadrightarrow_{\mathtt{name}}$) for the reflexive-transitive closure of $\to_\beta$ (resp. $\to_{\mathtt{name}}$).

2.2 Head, Weak-Head and Leftmost Reductions

In order to introduce different notions of reduction, we start by formalising the general mechanism of reduction which consists in contracting a redex at some specific occurrence. ***Occurrences*** are finite words over the alphabet $\{0, 1\}$. We use ϵ to denote the empty word and notation $\mathtt{a}^{\mathtt{n}}$ for $\mathtt{n} \in \mathbb{N}$ concatenations of some letter $\mathtt{a}$ of the alphabet. The set of ***occurrences*** of a given term is defined by induction as follows: $\mathtt{oc}(x) \stackrel{def}{=} \{\epsilon\}$; $\mathtt{oc}(t\,u) \stackrel{def}{=} \{\epsilon\} \cup \{0\mathtt{p} \mid \mathtt{p} \in \mathtt{oc}(t)\} \cup \{1\mathtt{p} \mid \mathtt{p} \in \mathtt{oc}(u)\}$; $\mathtt{oc}(\lambda x.t) \stackrel{def}{=} \{\epsilon\} \cup \{0\mathtt{p} \mid \mathtt{p} \in \mathtt{oc}(t)\}$.

Given two occurrences $\mathtt{p}$ and $\mathtt{q}$, we use the notation $\mathtt{p} \leq \mathtt{q}$ to mean that $\mathtt{p}$ is a ***prefix*** of $\mathtt{q}$, *i.e.* there is $\mathtt{p}'$ such that $\mathtt{pp}' = \mathtt{q}$. We denote by $t|_{\mathtt{p}}$ the ***subterm of*** t ***at occurrence*** $\mathtt{p}$, defined as expected [4], thus for example $((\lambda x.y)\,z)|_{00} = y$. The set of ***redex occurrences*** of t is defined by $\mathtt{roc}(t) \stackrel{def}{=} \{\mathtt{p} \in \mathtt{oc}(t) \mid t|_{\mathtt{p}} = (\lambda x.s)\,u\}$.

We use the notation $\mathsf{r} : t \to_\beta t'$ to mean that $\mathsf{r} \in \mathtt{roc}(t)$ and t reduces to t' by ***contracting*** the redex at occurrence r, *e.g.* $\mathsf{000} : (\lambda x.(\lambda y.y)\, x\, x)\, z \to_\beta (\lambda x.x\, x)\, z$. This notion is extended to reduction sequences as expected, and noted $\rho : t \twoheadrightarrow_\beta t'$, where ρ is the list of all the redex occurrences contracted along the reduction sequence. We use *nil* to denote the empty reduction sequence, so that $\mathit{nil} : t \twoheadrightarrow_\beta t$ holds for every term t.

Any term t has exactly one of the following forms: $\lambda x_1 \ldots \lambda x_n.y\, t_1 \ldots t_m$ or $\lambda x_1 \ldots \lambda x_n.(\lambda y.s)\, u\, t_1 \ldots t_m$ with $n, m \geq 0$. In the latter case we say that $(\lambda y.s)\, u$ is the ***head redex*** of t, while in the former case there is no head redex. Moreover, if $n = 0$, we say that $(\lambda y.s)\, u$ is the ***weak-head redex*** of t. In terms of occurrences, the *head redex* of t is the *minimal* redex occurrence of the form $\mathsf{0^n}$ with $\mathsf{n} \geq 0$. In particular, if it satisfies that $t|_{\mathsf{0^k}}$ is not an abstraction for every $\mathsf{k} \leq \mathsf{n}$, it is the *weak-head redex* of t. A reduction sequence contracting at each step the head redex (resp. weak-head redex) of the corresponding term is called the ***head reduction*** (resp. ***weak-head reduction***).

Given two redex occurrences $\mathsf{r}, \mathsf{r'} \in \mathtt{roc}(t)$, we say that r is ***to-the-left of*** $\mathsf{r'}$ if the anchor of r is to the left of the anchor of $\mathsf{r'}$. Thus for example, the redex occurrence $\mathsf{0}$ is to-the-left of $\mathsf{1}$ in the term $(I\, x)\, (I\, y)$, and ϵ is to-the-left of $\mathsf{00}$ in $(\lambda x.(I\, I))\, z$. Alternatively, the relation *to-the-left* can be understood as a dictionary order between redex occurrences, *i.e.* r is *to-the-left of* $\mathsf{r'}$ if either $\mathsf{r'} = \mathsf{rq}$ with $\mathsf{q} \neq \epsilon$ (*i.e.* r is a proper prefix of $\mathsf{r'}$); or $\mathsf{r} = \mathsf{p0q}$ and $\mathsf{r'} = \mathsf{p1q'}$ (*i.e.* they share a common prefix and r is on the left-hand side of an application while $\mathsf{r'}$ is on the right-hand side). Notice that in any case this implies $\mathsf{r'} \not\leq \mathsf{r}$. Since this notion defines a total order on redexes, every term not in normal form has a unique ***leftmost redex***. The term t ***leftmost reduces*** to t' if t reduces to t' and the reduction step contracts the leftmost redex of t. For example, $(I\, x)\, (I\, y)$ leftmost reduces to $x\, (I\, y)$ and $(\lambda x.(I\, I))\, z$ leftmost reduces to $I\, I$. This notion extends to reduction sequences as expected.

3 Towards neededness

Needed reduction is based on two fundamental notions: that of residual, which describes how a given redex is traced all along a reduction sequence, and that of normal form, which gives the form of the expected result of the reduction sequence. This section extends the standard notion of needed reduction [8] to those of head and weak-head needed reductions.

3.1 Residuals

Given a term t, $\mathsf{p} \in \mathtt{oc}(t)$ and $\mathsf{r} \in \mathtt{roc}(t)$, the ***descendants of*** p ***after*** r ***in*** t, written $\mathsf{p/r}$, is the set of *occurrences* defined as follows:

$$\begin{array}{r} \varnothing \text{ if } \mathsf{p} = \mathsf{r} \text{ or } \mathsf{p} = \mathsf{r0} \\ \{\mathsf{p}\} \text{ if } \mathsf{r} \not\leq \mathsf{p} \\ \{\mathsf{rq}\} \text{ if } \mathsf{p} = \mathsf{r00q} \\ \{\mathsf{rkq} \mid s|_{\mathsf{k}} = x\} \text{ if } \mathsf{p} = \mathsf{r1q} \text{ with } t|_{\mathsf{r}} = (\lambda x.s)\, u \end{array}$$

For instance, given $t = (\lambda x.(\lambda y.x)\, x)\, z$, then $\mathtt{oc}(t) = \{\epsilon, 0, 1, 00, 000, 001, 0000\}$, $\mathtt{roc}(t) = \{\epsilon, 00\}$, $00/00 = \varnothing$, $\epsilon/00 = \{\epsilon\}$, $00/\epsilon = \{\epsilon\}$ and $1/\epsilon = \{1, 00\}$.

Notice that $\mathsf{p}/\mathsf{r} \subseteq \mathtt{oc}(t')$ where $\mathsf{r} : t \to_\beta t'$. Furthermore, if p is the occurrence of a redex in t (*i.e.* $\mathsf{p} \in \mathtt{roc}(t)$), then $\mathsf{p}/\mathsf{r} \subseteq \mathtt{roc}(t')$, and each position in p/r is called a ***residual*** of p after reducing r. This notion is extended to sets of redex occurrences, indeed, the ***residuals of*** $\mathcal{P}$ ***after*** r ***in*** t are $\mathcal{P}/\mathsf{r} \stackrel{def}{=} \bigcup_{\mathsf{p}\in\mathcal{P}} \mathsf{p}/\mathsf{r}$. In particular $\varnothing/\mathsf{r} = \varnothing$. Given $\rho : t \twoheadrightarrow_\beta t'$ and $\mathcal{P} \subseteq \mathtt{roc}(t)$, the ***residuals of*** $\mathcal{P}$ ***after the sequence*** ρ are: $\mathcal{P}/nil \stackrel{def}{=} \mathcal{P}$ and $\mathcal{P}/\mathsf{r}\rho' \stackrel{def}{=} (\mathcal{P}/\mathsf{r})/\rho'$.

Stability of the to-the-left relation makes use of the notion of residual:

Lemma 1. *Given a term t, let $\mathsf{l}, \mathsf{r}, \mathsf{s} \in \mathtt{roc}(t)$ such that l is to-the-left of r, $\mathsf{s} \not\leq \mathsf{l}$ and $\mathsf{s} : t \to_\beta t'$. Then, $\mathsf{l} \in \mathtt{roc}(t')$ and l is to-the-left of r' for every $\mathsf{r}' \in \mathsf{r}/\mathsf{s}$.*

Proof. By case analysis using the definition of *to-the-left* [15]. □

Notice that this result does not only implies that the leftmost redex is preserved by reduction of other redexes, but also that the residual of the leftmost redex occurs in exactly the same occurrence as the original one.

Corollary 1. *Given a term t, and $\mathsf{l} \in \mathtt{roc}(t)$ the leftmost redex of t, if the reduction $\rho : t \twoheadrightarrow_\beta t'$ contracts neither l nor any of its residuals, then $\mathsf{l} \in \mathtt{roc}(t')$ is the leftmost redex of t'.*

Proof. By induction on the length of ρ using Lem. 1. □

3.2 Notions of Normal Form

The expected result of evaluating a program is specified by means of some appropriate notion of normal form. Given any relation $\to_\mathcal{R}$, a term t is said to be in ***$\mathcal{R}$-normal form*** ($\mathcal{NF}_\mathcal{R}$) iff there is no t' such that $t \to_\mathcal{R} t'$. A term t is ***$\mathcal{R}$-normalising*** ($\mathcal{WN}_\mathcal{R}$) iff there exists $u \in \mathcal{NF}_\mathcal{R}$ such that $t \twoheadrightarrow_\mathcal{R} u$. Thus, given an $\mathcal{R}$-normalising term t, we can define the set of $\mathcal{R}$-normal forms of t as $\mathtt{nf}_\mathcal{R}(t) \stackrel{def}{=} \{t' \mid t \twoheadrightarrow_\mathcal{R} t' \wedge t' \in \mathcal{NF}_\mathcal{R}\}$.

In particular, it turns out that a term in ***weak-head β-normal form*** ($\mathcal{WHNF}_\beta$) is of the form $x\, t_1 \ldots t_n$ ($n \geq 0$) or $\lambda x.t$, where $t, t_1, \ldots, t_n$ are arbitrary terms, *i.e.* it has no weak-head redex. The set of weak-head β-normal forms of t is $\mathtt{whnf}_\beta(t) \stackrel{def}{=} \{t' \mid t \twoheadrightarrow_\beta t' \wedge t' \in \mathcal{WHNF}_\beta\}$.

Similarly, a term in ***head β-normal form*** ($\mathcal{HNF}_\beta$) turns out to be of the form $\lambda x_1 \ldots \lambda x_n.x\, t_1 \ldots t_m$ ($n, m \geq 0$), *i.e.* it has no head redex. The set of head β-normal forms of t is given by $\mathtt{hnf}_\beta(t) \stackrel{def}{=} \{t' \mid t \twoheadrightarrow_\beta t' \wedge t' \in \mathcal{HNF}_\beta\}$.

Last, any term in ***β-normal form*** ($\mathcal{NF}_\beta$) has the form $\lambda x_1 \ldots \lambda x_n.x\, t_1 \ldots t_m$ ($n, m \geq 0$) where $t_1, \ldots, t_m$ are themselves in β-normal form. It is well-known that the set $\mathtt{nf}_\beta(t)$ is a singleton, so we may use it either as a set or as its unique element.

It is worth noticing that $\mathcal{NF}_\beta \subset \mathcal{HNF}_\beta \subset \mathcal{WHNF}_\beta$. Indeed, the inclusions are strict, for instance $\lambda x.(\lambda y.y)\, z$ is in weak-head but not in head β-normal form, while $x\, ((\lambda y.y)\, x)\, z$ is in head but not in β-normal form.

3.3 Notions of Needed Reduction

The different notions of normal form considered in Sec. 3.2 suggest different notions of needed reduction, besides the standard one in the literature [8]. Indeed, consider $\mathsf{r} \in \mathtt{roc}(t)$. We say that r is ***used*** in a reduction sequence ρ iff ρ reduces r or some residual of r. Then:

1. r is ***needed*** in t if every reduction sequence from t to β-normal form uses r;
2. r is ***head needed*** in t if every reduction sequence from t to head β-normal form uses r;
3. r is ***weak-head needed*** in t if every reduction sequence of t to weak-head β-normal form uses r.

Notice in particular that $\mathtt{nf}_\beta(t) = \varnothing$ (resp. $\mathtt{hnf}_\beta(t) = \varnothing$ or $\mathtt{whnf}_\beta(t) = \varnothing$) implies every redex in t is needed (resp. head needed or weak-head needed).

A ***one-step reduction*** $\to_\beta$ is ***needed*** (resp. ***head*** or ***weak-head needed***), noted $\to_{\mathtt{nd}}$ (resp. $\to_{\mathtt{hnd}}$ or $\to_{\mathtt{whnd}}$), if the contracted redex is needed (resp. head or weak-head needed). A ***reduction sequence*** $\twoheadrightarrow_\beta$ is ***needed*** (resp. ***head*** or ***weak-head needed***), noted $\twoheadrightarrow_{\mathtt{nd}}$ (resp. $\twoheadrightarrow_{\mathtt{hnd}}$ or $\twoheadrightarrow_{\mathtt{whnd}}$), if every reduction step in the sequence is needed (resp. head or weak-head needed).

For instance, consider the reduction sequence:

$$(\lambda y.\lambda x.I\, x\, (\underline{I\, I}_{\mathsf{r}_1}))\, (I\, I) \to_{\mathtt{nd}} (\lambda y.\lambda x.\underline{I\, x}_{\mathsf{r}_2}\, I)\, (I\, I) \to_{\mathtt{nd}} \underline{(\lambda y.\lambda x.x\, I)\, (I\, I)}_{\mathsf{r}_3} \to_{\mathtt{nd}} \lambda x.x\, I$$

which is needed but not head needed, since redex r_1 might not be contracted to reach a head normal form:

$$(\lambda y.\lambda x.\underline{I\, x}_{\mathsf{r}_2}\, (I\, I))\, (I\, I) \to_{\mathtt{hnd}} \underline{(\lambda y.\lambda x.x\, (I\, I))\, (I\, I)}_{\mathsf{r}_3} \to_{\mathtt{hnd}} \lambda x.x\, (I\, I)$$

Moreover, this second reduction sequence is head needed but not weak-head needed since only redex r_3 is needed to get a weak-head normal form:

$$\underline{(\lambda y.\lambda x.I\, x\, (I\, I))\, (I\, I)}_{\mathsf{r}_3} \to_{\mathtt{whnd}} \lambda x.I\, x\, (I\, I)$$

Notice that the following equalities hold: $\mathcal{NF}_{\mathtt{nd}} = \mathcal{NF}_\beta$, $\mathcal{NF}_{\mathtt{hnd}} = \mathcal{HNF}_\beta$ and $\mathcal{NF}_{\mathtt{whnd}} = \mathcal{WHNF}_\beta$.

Leftmost redexes and reduction sequences are indeed needed:

Lemma 2. *The leftmost redex in any term not in normal form (resp. head or weak-head normal form) is needed (resp. head or weak-head needed).*

Proof. By contradiction using the definition of *needed* [15]. □

Theorem 1. *Let $\mathsf{r} \in \mathtt{roc}(t)$ and $\rho : t \twoheadrightarrow_\beta t'$ be the leftmost reduction (resp. head reduction or weak-head reduction) starting with t such that $t' = \mathtt{nf}_\beta(t)$ (resp. $t' \in \mathtt{hnf}_\beta(t)$ or $t' \in \mathtt{whnf}_\beta(t)$). Then, r is needed (resp. head or weak-head needed) in t iff r is used in ρ.*

Proof. By definition of *needed* using Lem. 2 [15]. □

Notice that the weak-head reduction is a prefix of the head reduction, which is in turn a prefix of the leftmost reduction to normal form. As a consequence, it is immediate to see that every weak-head needed redex is in particular head needed, and every head needed redex is needed as well. For example, consider:

$$\underline{(\lambda y.\lambda x.\overline{I\,x}^{\mathsf{r}_2}\,(\overline{I\,I}^{\mathsf{r}_3}))\,(\overline{I\,I}^{\mathsf{r}_4})}_{\mathsf{r}_1}$$

where r_3 is a needed redex but not head needed nor weak-head needed. However, r_2 is both needed and head needed, while r_1 is the only weak-head needed redex in the term, and r_4 is not needed at all.

4 The Type System $\mathcal{V}$

In this section we recall the (non-idempotent) intersection type system $\mathcal{V}$ [14] –an extension of those in [13,12]– used here to characterise normalising terms w.r.t. the weak-head strategy. More precisely, we show that t is typable in system $\mathcal{V}$ if and only if t is normalising when only weak-head needed redexes are contracted. This characterisation is used in Sec. 9 to conclude that the weak-head needed strategy is observationally equivalent to the call-by-need calculus (to be introduced in Sec. 8).

Given a constant type $\mathtt{a}$ that denotes *answers* and a countable infinite set $\mathcal{B}$ of base type variables $\alpha, \beta, \gamma, \ldots$, we define the following sets of types:

$$\begin{array}{rll} \textbf{(Types)} & \tau, \sigma ::= \mathtt{a} \mid \alpha \in \mathcal{B} \mid \mathcal{M} \to \tau & \\ \textbf{(Multiset types)} & \mathcal{M}, \mathcal{N} ::= \{\!\{\tau_i\}\!\}_{i \in I} & \text{where } I \text{ is a finite set} \end{array}$$

The empty multiset is denoted by $\{\!\{\}\!\}$. We remark that types are *strict* [18], *i.e.* the right-hand sides of functional types are never multisets. Thus, the general form of a type is $\mathcal{M}_1 \to \ldots \to \mathcal{M}_n \to \tau$ with τ being the constant type or a base type variable.

Typing contexts (or just ***contexts***), written Γ, Δ, are functions from variables to multiset types, assigning the empty multiset to all but a finite set of variables. The domain of Γ is given by $\mathtt{dom}(\Gamma) \stackrel{def}{=} \{x \mid \Gamma(x) \neq \{\!\{\}\!\}\}$. The ***union of contexts***, written $\Gamma + \Delta$, is defined by $(\Gamma + \Delta)(x) \stackrel{def}{=} \Gamma(x) \sqcup \Delta(x)$, where $\sqcup$ denotes multiset union. An example is $(x : \{\!\{\sigma\}\!\}, y : \{\!\{\tau\}\!\}) + (x : \{\!\{\sigma\}\!\}, z : \{\!\{\tau\}\!\}) = (x : \{\!\{\sigma, \sigma\}\!\}, y : \{\!\{\tau\}\!\}, z : \{\!\{\tau\}\!\})$. This notion is extended to several contexts as expected, so that $+_{i \in I} \Gamma_i$ denotes a finite union of contexts (when $I = \varnothing$ the notation is to be understood as the empty context). We write $\Gamma \setminus\!\setminus x$ for the context $(\Gamma \setminus\!\setminus x)(x) = \{\!\{\}\!\}$ and $(\Gamma \setminus\!\setminus x)(y) = \Gamma(y)$ if $y \neq x$.

Type judgements have the form $\Gamma \vdash t : \tau$, where Γ is a typing context, t is a term and τ is a type. The intersection type system $\mathcal{V}$ for the λ-calculus is given in Fig. 1.

The constant type $\mathtt{a}$ in rule ($\mathtt{val}$) is used to type values. The axiom ($\mathtt{ax}$) is relevant (there is no weakening) and the rule ($\to\mathtt{e}$) is multiplicative. Note that the argument of an application is typed $\#(I)$ times by the premises of rule ($\to\mathtt{e}$). A particular case is when $I = \varnothing$: the subterm u occurring in the typed term $t\,u$ turns out to be untyped.

$$\frac{}{x : \{\!\{\tau\}\!\} \vdash x : \tau}\,(\mathtt{ax}) \qquad \frac{\Gamma \vdash t : \tau}{\Gamma \setminus\!\setminus x \vdash \lambda x.t : \Gamma(x) \to \tau}\,(\to\mathtt{i})$$

$$\frac{}{\vdash \lambda x.t : \mathtt{a}}\,(\mathtt{val}) \qquad \frac{\Gamma \vdash t : \{\!\{\sigma_i\}\!\}_{i\in I} \to \tau \quad (\Delta_i \vdash u : \sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash t\,u : \tau}\,(\to\mathtt{e})$$

Fig. 1. The non-idempotent intersection type system $\mathcal{V}$.

A ***(type) derivation*** is a tree obtained by applying the (inductive) typing rules of system $\mathcal{V}$. The notation $\rhd_{\mathcal{V}} \Gamma \vdash t : \tau$ means there is a derivation of the judgement $\Gamma \vdash t : \tau$ in system $\mathcal{V}$. The term t is typable in system $\mathcal{V}$, or $\mathcal{V}$-typable, iff t is the ***subject*** of some derivation, *i.e.* iff there are Γ and τ such that $\rhd_{\mathcal{V}} \Gamma \vdash t : \tau$. We use the capital Greek letters $\Phi, \Psi, \ldots$ to name type derivations, by writing for example $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$. For short, we usually denote with Φ_t a derivation with subject t for some type and context. The ***size of the derivation*** Φ, denoted by $\mathtt{sz}(\Phi)$, is defined as the number of nodes of the corresponding derivation tree. We write $\mathtt{RULE}(\Phi) \in \{(\mathtt{ax}), (\to\mathtt{i}), (\to\mathtt{e})\}$ to access the last rule applied in the derivation Φ. Likewise, $\mathtt{PREM}(\Phi)$ is the *multiset* of proper maximal subderivations of Φ. For instance, given

$$\Phi = \frac{\Phi_t \quad (\Phi^i_u)_{i\in I}}{\Gamma \vdash t\,u : \tau}\,(\to\mathtt{e})$$

we have $\mathtt{RULE}(\Phi) = (\to\mathtt{e})$ and $\mathtt{PREM}(\Phi) = \{\!\{\Phi_t\}\!\} \sqcup \{\!\{\Phi^i_u \mid i \in I\}\!\}$. We also use functions $\mathtt{CTXT}(\Phi)$, $\mathtt{SUBJ}(\Phi)$ and $\mathtt{TYPE}(\Phi)$ to access the context, subject and type of the judgement in the root of the derivation tree respectively. For short, we also use notation $\Phi(x)$ to denote the type associated to the variable x in the typing environment of the conclusion of Φ (*i.e.* $\Phi(x) \stackrel{def}{=} \mathtt{CTXT}(\Phi)(x)$).

Intersection type systems can usually be seen as models [11], *i.e.* typing is stable by convertibility: if t is typable and $t =_\beta t'$, then t' is typable too. This property splits in two different statements known as *subject reduction* and *subject expansion* respectively, the first one giving stability of typing by reduction, the second one by expansion. In the particular case of *non-idempotent types*, subject reduction refines to *weighted subject-reduction*, stating that not only typability is stable by reduction, but also that the size of type derivations is decreasing. Moreover, this decrease is strict when reduction is performed on special occurrences of redexes, called *typed occurrences*. We now introduce all these concepts.

Given a type derivation Φ, the set $\mathtt{TOC}(\Phi)$ of ***typed occurrences*** of Φ, which is a subset of $\mathtt{oc}(\mathtt{SUBJ}(\Phi))$, is defined by induction on the last rule of Φ.

- If $\mathtt{RULE}(\Phi) \in \{(\mathtt{ax}), (\mathtt{val})\}$, then $\mathtt{TOC}(\Phi) \stackrel{def}{=} \{\epsilon\}$.
- If $\mathtt{RULE}(\Phi) = (\to\mathtt{i})$ with $\mathtt{SUBJ}(\Phi) = \lambda x.t$ and $\mathtt{PREM}(\Phi) = \{\!\{\Phi_t\}\!\}$, then $\mathtt{TOC}(\Phi) \stackrel{def}{=} \{\epsilon\} \cup \{\mathtt{0p} \mid \mathtt{p} \in \mathtt{TOC}(\Phi_t)\}$.
- If $\mathtt{RULE}(\Phi) = (\to\mathtt{e})$ with $\mathtt{SUBJ}(\Phi) = t\,u$ and $\mathtt{PREM}(\Phi) = \{\!\{\Phi_t\}\!\} \sqcup \{\!\{\Phi^i_u \mid i \in I\}\!\}$, then $\mathtt{TOC}(\Phi) \stackrel{def}{=} \{\epsilon\} \cup \{\mathtt{0p} \mid \mathtt{p} \in \mathtt{TOC}(\Phi_t)\} \cup (\bigcup_{i\in I} \{\mathtt{1p} \mid \mathtt{p} \in \mathtt{TOC}(\Phi^i_u)\})$.

Remark that there are two kind of untyped occurrences, those inside untyped arguments of applications, and those inside untyped bodies of abstractions. For instance consider the following type derivations:

$$\Phi_K = \dfrac{\dfrac{\dfrac{}{x : \{\!\{\mathtt{a}\}\!\} \vdash x : \mathtt{a}}\,(\mathtt{ax})}{x : \{\!\{\mathtt{a}\}\!\} \vdash \lambda y.x : \{\!\{\}\!\} \to \mathtt{a}}\,(\to\mathtt{i})}{\vdash K : \{\!\{\mathtt{a}\}\!\} \to \{\!\{\}\!\} \to \mathtt{a}}\,(\to\mathtt{i}) \qquad \Phi_{KI} = \dfrac{\Phi_K \qquad \dfrac{}{\vdash I : \mathtt{a}}\,(\mathtt{val})}{\vdash K\,I : \{\!\{\}\!\} \to \mathtt{a}}\,(\to\mathtt{e}) \qquad \Phi_{KI\Omega} = \dfrac{\Phi_{KI}}{\vdash K\,I\,\Omega : \mathtt{a}}\,(\to\mathtt{e})$$

Then, $\mathtt{TOC}(\Phi_{KI\Omega}) = \{\epsilon, \mathtt{0}, \mathtt{00}, \mathtt{01}, \mathtt{000}, \mathtt{0000}\} \subseteq \mathtt{oc}(KI\Omega)$.

Remark 1. The weak-head redex of a typed term is always a typed occurrence.

Given Φ and $\mathtt{p} \in \mathtt{TOC}(\Phi)$, we define $\Phi|_{\mathtt{p}}$ as the *multiset* of ***all the subderivations of Φ at occurrence*** $\mathtt{p}$ (a formal definition can be found in [15]). Note that $\Phi|_{\mathtt{p}}$ is a multiset since the subterm of $\mathtt{SUBJ}(\Phi)$ at position $\mathtt{p}$ may be typed several times in Φ, due to rule $(\to\mathtt{e})$.

We can now state the two main properties of system $\mathcal{V}$, whose proofs can be found in Sec. 7 of [9].

Theorem 2 (Weighted Subject Reduction). *Let $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$. If $\mathtt{r} : t \to_\beta t'$, then there exists Φ' s.t. $\Phi' \rhd_{\mathcal{V}} \Gamma \vdash t' : \tau$. Moreover,*

1. *If $\mathtt{r} \in \mathtt{TOC}(\Phi)$, then $\mathtt{sz}(\Phi) > \mathtt{sz}(\Phi')$.*
2. *If $\mathtt{r} \notin \mathtt{TOC}(\Phi)$, then $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi')$.*

Theorem 3 (Subject Expansion). *Let $\Phi' \rhd_{\mathcal{V}} \Gamma \vdash t' : \tau$. If $t \to_\beta t'$, then there exists Φ s.t. $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$.*

Note that weighted subject reduction implies that reduction of typed redex occurrences turns out to be normalising.

5 Substitution and Reduction on Derivations

In order to relate typed redex occurrences of convertible terms, we now extend the notion of β-reduction to derivation trees, by making use of a natural and basic concept of typed substitution. In contrast to substitution and β-reduction on *terms*, these operations are now both non-deterministic on derivation trees (see [19] for discussions and examples). Given a variable x and type derivations Φ_t and $(\Phi_u^i)_{i\in I}$, the ***typed substitution*** of x by $(\Phi_u^i)_{i\in I}$ in Φ_t, written $\Phi_t\left\{x \,/ (\Phi_u^i)_{i\in I}\right\}$ by making an abuse of notation, is a type derivation inductively defined on Φ_t, only if $\Phi_t(x) = \{\!\{\mathtt{TYPE}(\Phi_u^i)\}\!\}_{i\in I}$. This non-deterministic construction may be non-trivial but it can be naturally formalised in a quite straightforward way (full details can be found in [15]). Intuitively, the typed substitution replaces typed occurrences of x in Φ_t by a corresponding derivation Φ_u^i matching the same type, where such a matching is chosen in a non-deterministic way. Moreover, it also substitutes all untyped occurrences of x by u, where this

untyped operation is completely deterministic. Thus, for example, consider the following substitution, where Φ_{KI} is defined in Sec. 4:

$$\left(\dfrac{\dfrac{}{x : \{\!\{\{\!\{\}\!\} \to \mathtt{a}\}\!\} \vdash x : \{\!\{\}\!\} \to \mathtt{a}}\,(\mathtt{ax})}{x : \{\!\{\{\!\{\}\!\} \to \mathtt{a}\}\!\} \vdash x\,x : \mathtt{a}}\,(\to\mathtt{e})\right)\{x\,/\Phi_{KI}\} = \dfrac{\Phi_{KI}}{\vdash (KI)\,(KI) : \mathtt{a}}\,(\to\mathtt{e})$$

The following lemma relates the typed occurrences of the trees composing a substitution and those of the substituted tree itself:

Lemma 3. *Let Φ_t and $(\Phi^i_u)_{i\in I}$ be derivations such that $\Phi_t\left\{x\,/(\Phi^i_u)_{i\in I}\right\}$ is defined, and $\mathtt{p} \in \mathtt{oc}(t)$. Then,*

1. $\mathtt{p} \in \mathtt{TOC}(\Phi_t)$ *iff* $\mathtt{p} \in \mathtt{TOC}(\Phi_t\left\{x\,/(\Phi^i_u)_{i\in I}\right\})$.
2. $\mathtt{q} \in \mathtt{TOC}(\Phi^k_u)$ *for some* $k \in I$ *iff there exists* $\mathtt{p} \in \mathtt{TOC}(\Phi_t)$ *such that* $t|_{\mathtt{p}} = x$ *and* $\mathtt{pq} \in \mathtt{TOC}(\Phi_t\left\{x\,/(\Phi^i_u)_{i\in I}\right\})$.

Proof. By induction on Φ_t. □

Based on the previous notion of substitutions on derivations, we are now able to introduce (non-deterministic) reduction on derivation trees. The ***reduction relation*** $\to_\beta$ on derivation trees is then defined by first considering the following basic rewriting rules.

1. For typed β-redexes:

$$\dfrac{\dfrac{\Phi_t \rhd_{\mathcal{V}} \Gamma; x : \{\!\{\sigma_i\}\!\}_{i\in I} \vdash t : \tau}{\Gamma \vdash \lambda x.t : \{\!\{\sigma_i\}\!\}_{i\in I} \to \tau} \qquad (\Phi^i_u \rhd_{\mathcal{V}} \Delta_i \vdash u : \sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash (\lambda x.t)\,u : \tau} \mapsto_\beta \ \Phi_t\left\{x\,/(\Phi^i_u)_{i\in I}\right\}$$

2. For β-redexes in untyped occurrences, with $u \to_\beta u'$:

$$\dfrac{\Gamma \vdash t : \{\!\{\}\!\} \to \tau}{\Gamma \vdash t\,u : \tau} \mapsto_\nu \dfrac{\Gamma \vdash t : \{\!\{\}\!\} \to \tau}{\Gamma \vdash t\,u' : \tau} \qquad \dfrac{}{\vdash \lambda x.u : \mathtt{a}} \mapsto_\xi \dfrac{}{\vdash \lambda x.u' : \mathtt{a}}$$

As in the case of the λ-calculus, where reduction is closed under usual *term* contexts, we need to close the previous relation under *derivation tree* contexts. However, a one-step reduction on a given subterm causes many one-step reductions in the corresponding derivation tree (recall $\Phi|_{\mathtt{p}}$ is defined to be a multiset). Then, informally, given a redex occurrence $\mathtt{r}$ of t, a type derivation Φ of t, and the multiset of minimal subderivations of Φ containing $\mathtt{r}$, written $\mathscr{M}$, we apply the reduction rules $\mapsto_{\beta,\nu,\xi}$ to all the elements of $\mathscr{M}$, thus obtaining a multiset $\mathscr{M}'$, and we recompose the type derivation of the reduct of t (see [15] for a formal definition). This gives the reduction relation $\to_\beta$ on trees. A reduction sequence on derivation trees contracting only redexes in typed positions is dubbed a ***typed reduction sequence***.

Note that typed reductions are normalising by Thm. 2, yielding a special kind of derivation. Indeed, given a type derivation $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$, we say that Φ is ***normal*** iff $\mathtt{TOC}(\Phi) \cap \mathtt{roc}(t) = \varnothing$. Reduction on trees induces reduction on terms: when $\rho : \Phi \twoheadrightarrow_\beta \Phi'$, then $\mathtt{SUBJ}(\Phi) \twoheadrightarrow_\beta \mathtt{SUBJ}(\Phi')$. By abuse of notation we may denote both sequences with the same letter ρ.

6 Weak-Head Neededness and Typed Occurrences

This section presents one of our main results. It establishes a connection between weak-head needed redexes and typed redex occurrences. More precisely, we first show in Sec. 6.1 that every weak-head needed redex occurrence turns out to be a typed occurrence, whatever its type derivation is. The converse does not however hold. But, we show in Sec. 6.2 that any typed occurrence in a special kind of typed derivation (that we call principal) corresponds to a weak-head needed redex occurrence. We start with a technical lemma.

Lemma 4. *Let* $\mathsf{r} : \Phi_t \to_\beta \Phi_{t'}$ *and* $\mathsf{p} \in \mathtt{oc}(t)$ *such that* $\mathsf{p} \neq \mathsf{r}$ *and* $\mathsf{p} \neq \mathsf{r}0$*. Then,* $\mathsf{p} \in \mathtt{TOC}(\Phi_t)$ *iff there exists* $\mathsf{p}' \in \mathsf{p}/\mathsf{r}$ *such that* $\mathsf{p}' \in \mathtt{TOC}(\Phi_{t'})$*.*

Proof. By induction on r using Lem. 3. □

6.1 Weak-Head Needed Redexes are Typed

In order to show that every weak-head needed redex occurrence corresponds to a typed occurrence in some type derivation we start by proving that typed occurrences do not come from untyped ones.

Lemma 5. *Let* $\rho : \Phi_t \twoheadrightarrow_\beta \Phi_{t'}$ *and* $\mathsf{p} \in \mathtt{oc}(t)$*. If there exists* $\mathsf{p}' \in \mathsf{p}/\rho$ *such that* $\mathsf{p}' \in \mathtt{TOC}(\Phi_{t'})$*, then* $\mathsf{p} \in \mathtt{TOC}(\Phi_t)$*.*

Proof. Straightforward induction on ρ using Lem. 4. □

Theorem 4. *Let* r *be a weak-head needed redex in* t*. Let* Φ *be a type derivation of* t*. Then,* $\mathsf{r} \in \mathtt{TOC}(\Phi)$*.*

Proof. By Thm. 1, r is used in the weak-head reduction from t to $t' \in \mathcal{WHNF}_\beta$. By Rem. 1, the weak-head reduction contracts only typed redexes. Thus, r or some of its residuals is a typed occurrence in its corresponding derivation tree. Finally, we conclude by Lem. 5, $\mathsf{r} \in \mathtt{TOC}(\Phi)$. □

6.2 Principally Typed Redexes are Weak-Head Needed

As mentioned before, the converse of Thm. 4 does not hold: there are some typed occurrences that do not correspond to any weak-head needed redex occurrence. This can be illustrated in the following examples (recall $\Phi_{KI\Omega}$ defined in Sec. 4):

$$\frac{\Phi_{KI\Omega}}{\vdash \lambda y.KI\Omega : \{\!\{\}\!\} \to \mathtt{a}}\,(\to\mathtt{i}) \qquad \frac{\dfrac{}{y : \{\!\{\{\!\{\mathtt{a}\}\!\} \to \mathtt{a}\}\!\} \vdash y : \{\!\{\mathtt{a}\}\!\} \to \mathtt{a}}\,(\mathtt{ax}) \qquad \Phi_{KI\Omega}}{y : \{\!\{\{\!\{\mathtt{a}\}\!\} \to \mathtt{a}\}\!\} \vdash y\,(KI\Omega) : \mathtt{a}}\,(\to\mathtt{e})$$

Indeed, the occurrence $\mathtt{0}$ (resp $\mathtt{1}$) in the term $\lambda y.KI\Omega$ (resp. $y\,(KI\Omega)$) is typed but not weak-head needed, since both terms are already in weak-head normal form. Fortunately, typing relates to redex occurrences if we restrict type derivations to *principal* ones: given a term t in weak-head β-normal form, the derivation $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$ is ***normal principally typed*** if:

– $t = x\,t_1 \dots t_n$ $(n \geq 0)$, and $\Gamma = \{x : \{\!\{\overbrace{\{\!\{\}\!\} \to \dots \to \{\!\{\}\!\}}^{n \text{ times}} \to \tau\}\!\}\}$ and τ is a type variable α (*i.e.* none of the t_i are typed), or
– $t = \lambda x.t'$, and $\Gamma = \varnothing$ and $\tau = \mathtt{a}$.

Given a weak-head normalising term t such that $\Phi_t \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$, we say that Φ_t is ***principally typed*** if $\Phi_t \twoheadrightarrow_\beta \Phi_{t'}$ for some $t' \in \mathtt{whnf}_\beta(t)$ implies $\Phi_{t'}$ is normal principally typed.

Note in particular that the previous definition does not depend on the chosen weak-head normal form t': suppose $t'' \in \mathtt{whnf}_\beta(t)$ is another weak-head normal form of t, then t' and t'' are convertible terms by the Church-Rosser property [7] so that t' can be normal principally typed iff t'' can, by Thm. 2 and 3.

Lemma 6. *Let Φ_t be a type derivation with subject t and $\mathtt{r} \in \mathtt{roc}(t) \cap \mathtt{TOC}(\Phi_t)$. Let $\rho : \Phi_t \twoheadrightarrow_\beta \Phi_{t'}$ such that $\Phi_{t'}$ is normal. Then, $\mathtt{r}$ is used in ρ.*

Proof. Straightforward induction on ρ using Lem. 4. □

The notions of leftmost and weak-head needed reductions on (untyped) terms naturally extends to *typed* reductions on tree derivations. We thus have:

Lemma 7. *Let t be a weak-head normalising term and Φ_t be principally typed. Then, a leftmost typed reduction sequence starting at Φ_t is weak-head needed.*

Proof. By induction on the leftmost typed sequence (called ρ). If ρ is empty the result is immediate. If not, we show that t has a typed weak-head needed redex (which is leftmost by definition) and conclude by inductive hypothesis. Indeed, assume $t \in \mathcal{WHNF}_\beta$. By definition Φ_t is normal principally typed and thus it has no typed redexes. This contradicts ρ being non-empty. Hence, t has a weak-head redex $\mathtt{r}$ (*i.e.* $t \notin \mathcal{WHNF}_\beta$). Moreover, $\mathtt{r}$ is both typed (by Rem. 1) and weak-head needed (by Lem. 2). Thus, we conclude. □

Theorem 5. *Let t be a weak-head normalising term, Φ_t be principally typed and $\mathtt{r} \in \mathtt{roc}(t) \cap \mathtt{TOC}(\Phi_t)$. Then, $\mathtt{r}$ is a weak-head needed redex in t.*

Proof. Let $\rho : \Phi_t \twoheadrightarrow_\beta \Phi_{t'}$ be the leftmost typed reduction sequence where $\Phi_{t'}$ is normal. Note that $\Phi_{t'}$ exists by definition of *principally typed*. By Lem. 7, ρ is a weak-head needed reduction sequence. Moreover, by Lem. 6, $\mathtt{r}$ is used in ρ. Hence, $\mathtt{r}$ is a weak-head needed redex in t. □

As a direct consequence of Thm. 4 and 5, given a weak-head normalising term t, the typed redex occurrences in its principally typed derivation (which always exists) correspond to its weak-head needed redexes. Hence, system $\mathcal{V}$ allows to identify all the weak-head needed redexes of a weak-head normalising term.

7 Characterising Weak-Head Needed Normalisation

This section presents one of the main pieces contributing to our observational equivalence result. Indeed, we relate typing with weak-head neededness by showing that any typable term in system $\mathcal{V}$ is normalising for weak-head needed reduction. This characterisation highlights the power of intersection types. We start by a technical lemma.

Lemma 8. *Let $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$. Then, Φ normal implies $t \in \mathcal{WHNF}_\beta$.*

Proof. By induction on Φ analysing the last rule applied. □

Let $\rho : t_1 \twoheadrightarrow_\beta t_n$. We say that ρ is a ***left-to-right*** reduction sequence iff for every $i < n$ if $\mathsf{r_i} : t_i \to_\beta t_{i+1}$ and $\mathsf{l_i}$ is to the left of $\mathsf{r_i}$ then, for every $j > i$ such that $\mathsf{r_j} : t_j \to_\beta t_{j+1}$ we have that $\mathsf{r_j} \notin \{\mathsf{l_i}\}/\rho_{ij}$ where $\rho_{ij} : t_i \twoheadrightarrow_\beta t_j$ is the corresponding subsequence of ρ. In other words, for every j and every $i < j$, $\mathsf{r_j}$ is not a residual of a redex to the left of $\mathsf{r_i}$ (relative to the given reduction subsequence from t_i to t_j) [7].

Left-to-right reductions define in particular standard strategies, which give canonical ways to construct reduction sequences from one term to another:

Theorem 6 ([7]). *If $t \twoheadrightarrow_\beta t'$, there exists a left-to-right reduction from t to t'.*

Theorem 7. *Let $t \in \mathcal{T}_{\mathtt{a}}$. Then, $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$ iff $t \in \mathcal{WN}_{\mathtt{whnd}}$.*

Proof. $\Rightarrow$) By Thm. 2 we know that the strategy reducing only typed redex occurrences is normalising, *i.e.* there exist t' and Φ' such that $t \twoheadrightarrow_\beta t'$, $\Phi' \rhd_{\mathcal{V}} \Gamma \vdash t' : \tau$ and Φ' normal. Then, by Lem. 8, $t' \in \mathcal{WHNF}_\beta$. By Thm. 6, there exists a left-to-right reduction $\rho : t \twoheadrightarrow_\beta t'$. Let us write

$$\rho : t = t_1 \twoheadrightarrow_\beta t_n \twoheadrightarrow_\beta t'$$

such that $t_1, \ldots, t_{n-1} \notin \mathcal{WHNF}_\beta$ and $t_n \in \mathcal{WHNF}_\beta$.

We claim that all reduction steps in $t_1 \twoheadrightarrow_\beta t_n$ are leftmost. Assume towards a contradiction that there exists $k < n$ such that $\mathsf{r} : t_k \to_\beta t_{k+1}$ and r is not the leftmost redex of t_k (written $\mathsf{l_k}$). Since ρ is a left-to-right reduction, no residual of $\mathsf{l_k}$ is contracted after the k-th step. Thus, there is a reduction sequence from $t_k \notin \mathcal{WHNF}_\beta$ to $t_n \in \mathcal{WHNF}_\beta$ such that $\mathsf{l_k}$ is not used in it. This leads to a contradiction with $\mathsf{l_k}$ being weak-head needed in t_k by Lem. 2.

As a consequence, there is a leftmost reduction sequence $t \twoheadrightarrow_\beta t_n$. Moreover, by Lem. 2, $t \twoheadrightarrow_{\mathtt{whnd}} t_n \in \mathcal{WHNF}_\beta = \mathcal{NF}_{\mathtt{whnd}}$. Thus, $t \in \mathcal{WN}_{\mathtt{whnd}}$.

$\Leftarrow$) Consider the reduction $\rho : t \twoheadrightarrow_{\mathtt{whnd}} t'$ with $t' \in \mathtt{whnf}_\beta(t)$. Let $\Phi' \rhd_{\mathcal{V}} \Gamma \vdash t' : \tau$ be the normal principally typed derivation for t' as defined in Sec. 6.2. Finally, we conclude by induction in ρ using Thm. 3, $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$. □

8 The Call-by-Need Lambda-Calculus

This section describes the syntax and the operational semantics of the call-by-need lambda-calculus introduced in [1]. It is more concise than previous specifications of call-by-need [3,2,16,10], but it is operationally equivalent to them [6], so that our results could also be presented by using alternative specifications.

Given a countable infinite set $\mathcal{X}$ of variables $x, y, z, \ldots$ we define different syntactic categories for terms, values, list contexts, answers and need contexts:

$$\begin{array}{rl}
\textbf{(Terms)} & t, u ::= x \in \mathcal{X} \mid t\, u \mid \lambda x.t \mid t[x\backslash u] \\
\textbf{(Values)} & v ::= \lambda x.t \\
\textbf{(List contexts)} & \mathtt{L} ::= \Box \mid \mathtt{L}[x\backslash t] \\
\textbf{(Answers)} & a ::= \mathtt{L}\langle \lambda y.t \rangle \\
\textbf{(Need contexts)} & \mathtt{M}, \mathtt{N} ::= \Box \mid \mathtt{N}\, t \mid \mathtt{N}[x\backslash t] \mid \mathtt{N}\langle\!\langle x \rangle\!\rangle [x\backslash \mathtt{M}]
\end{array}$$

We denote the set of terms by $\mathcal{T}_{\mathtt{e}}$. Terms of the form $t[x\backslash u]$ are ***closures***, and $[x\backslash u]$ is called an ***explicit substitution*** (ES). The set of $\mathcal{T}_{\mathtt{e}}$-terms without ES is the set of *terms of the λ-calculus*, *i.e.* $\mathcal{T}_{\mathtt{a}}$. The notions of ***free*** and ***bound*** variables are defined as expected, in particular, $\mathtt{fv}(t[x\backslash u]) \stackrel{def}{=} \mathtt{fv}(t) \setminus \{x\} \cup \mathtt{fv}(u)$, $\mathtt{fv}(\lambda x.t) \stackrel{def}{=} \mathtt{fv}(t) \backslash \{x\}$, $\mathtt{bv}(t[x\backslash u]) \stackrel{def}{=} \mathtt{bv}(t) \cup \{x\} \cup \mathtt{bv}(u)$ and $\mathtt{bv}(\lambda x.t) \stackrel{def}{=} \mathtt{bv}(t) \cup \{x\}$. We extend the standard notion of ***α-conversion*** to ES, as expected.

We use the special notation $\mathtt{N}\langle\!\langle u \rangle\!\rangle$ or $\mathtt{L}\langle\!\langle u \rangle\!\rangle$ when the free variables of u are not captured by the context, *i.e.* there are no abstractions or explicit substitutions in the context that binds the free variables of u. Thus for example, given $\mathtt{N} = (\Box\, x)[x\backslash z]$, we have $(y\, x)[x\backslash z] = \mathtt{N}\langle y \rangle = \mathtt{N}\langle\!\langle y \rangle\!\rangle$, but $(x\, x)[x\backslash z] = \mathtt{N}\langle x \rangle$ cannot be written as $\mathtt{N}\langle\!\langle x \rangle\!\rangle$. Notice the use of this special notation in the last case of needed contexts, an example of such case being $(x\, y)[y\backslash t][x\backslash \Box]$.

The ***call-by-need calculus***, introduced in [1], is given by the set of terms $\mathcal{T}_{\mathtt{e}}$ and the ***reduction relation*** $\to_{\mathtt{need}}$, the ***union*** of $\to_{\mathtt{dB}}$ and $\to_{\mathtt{lsv}}$, which are, respectively, the closure by *need contexts* of the following rewriting rules:

$$\begin{array}{c}
\mathtt{L}\langle \lambda x.t \rangle\, u \mapsto_{\mathtt{dB}} \mathtt{L}\langle t[x\backslash u] \rangle \\
\mathtt{N}\langle\!\langle x \rangle\!\rangle [x\backslash \mathtt{L}\langle v \rangle] \mapsto_{\mathtt{lsv}} \mathtt{L}\langle \mathtt{N}\langle\!\langle v \rangle\!\rangle [x\backslash v] \rangle
\end{array}$$

These rules avoid capture of free variables. An example of `need`-reduction sequence is the following, where the redex of each step is underlined for clearness:

$$\begin{array}{lll}
\underline{(\lambda x_1.I\,(x_1\, I))\,(\lambda y.I\, y)} & \to_{\mathtt{dB}} & \underline{(I\,(x_1\, I))}[x_1\backslash \lambda y.I\, y] \to_{\mathtt{dB}} \\
\underline{x_2[x_2\backslash x_1\, I][x_1\backslash \lambda y.I\, y]} & \to_{\mathtt{lsv}} & x_2[x_2\backslash \underline{(\lambda x_3.I\, x_3)\, I}][x_1\backslash \lambda y.I\, y] \to_{\mathtt{dB}} \\
x_2[x_2\backslash \underline{(I\, x_3)}[x_3\backslash I]][x_1\backslash \lambda y.I\, y] & \to_{\mathtt{dB}} & x_2[x_2\backslash \underline{x_4[x_4\backslash x_3][x_3\backslash I]}][x_1\backslash \lambda y.I\, y] \to_{\mathtt{lsv}} \\
x_2[x_2\backslash \underline{x_4[x_4\backslash I}][x_3\backslash I]][x_1\backslash \lambda y.I\, y] & \to_{\mathtt{lsv}} & \underline{x_2[x_2\backslash I[x_4\backslash I][x_3\backslash I]]}[x_1\backslash \lambda y.I\, y] \to_{\mathtt{lsv}} \\
I[x_2\backslash I][x_4\backslash I][x_3\backslash I][x_1\backslash \lambda y.I\, y] & &
\end{array}$$

As for call-by-name, reduction preserves free variables, *i.e.* $t \to_{\mathtt{need}} t'$ implies $\mathtt{fv}(t) \supseteq \mathtt{fv}(t')$. Notice that call-by-need reduction is also weak, so that answers are not `need`-reducible.

9 Observational Equivalence

The results in Sec. 7 are used here to prove soundness and completeness of call-by-need w.r.t weak-head neededness, our second main result. More precisely, a call-by-need interpreter stops in a value if and only if the weak-head needed reduction stops in a value. This means that call-by-need and call-by-name are observationally equivalent.

Formally, given a reduction relation $\mathcal{R}$ on a term language $\mathcal{T}$, and an associated notion of context for $\mathcal{T}$, we define t to be ***observationally equivalent*** to u, written $t \cong_{\mathcal{R}} u$, iff $\mathtt{C}\langle t\rangle \in \mathcal{WN}_{\mathcal{R}} \Leftrightarrow \mathtt{C}\langle u\rangle \in \mathcal{WN}_{\mathcal{R}}$ for every context $\mathtt{C}$. In order to show our final result we resort to the following theorem:

Theorem 8 ([14]).

1. *Let $t \in \mathcal{T}_{\mathtt{a}}$. Then, $\Phi \rhd_{\mathcal{V}} \Gamma \vdash t : \tau$ iff $t \in \mathcal{WN}_{\mathtt{name}}$.*
2. *For all terms t and u in $\mathcal{T}_{\mathtt{a}}$, $t \cong_{\mathtt{name}} u$ iff $t \cong_{\mathtt{need}} u$.*

These observations allows us to conclude:

Theorem 9. *For all terms t and u in $\mathcal{T}_{\mathtt{a}}$, $t \cong_{\mathtt{whnd}} u$ iff $t \cong_{\mathtt{need}} u$.*

Proof. By Thm. 8:2 it is sufficient to show $t \cong_{\mathtt{whnd}} u$ iff $t \cong_{\mathtt{name}} u$. The proof proceeds as follows:

$$
\begin{array}{rcll}
t \cong_{\mathtt{name}} u & & \text{iff} & \text{(definition)} \\
\mathtt{C}\langle t\rangle \in \mathcal{WN}_{\mathtt{name}} & \Leftrightarrow \quad \mathtt{C}\langle u\rangle \in \mathcal{WN}_{\mathtt{name}} & \text{iff} & \text{(Thm. 8:1)} \\
\mathtt{C}\langle t\rangle \text{ typable in } \mathcal{V} & \Leftrightarrow \quad \mathtt{C}\langle u\rangle \text{ typable in } \mathcal{V} & \text{iff} & \text{(Thm. 7)} \\
\mathtt{C}\langle t\rangle \in \mathcal{WN}_{\mathtt{whnd}} & \Leftrightarrow \quad \mathtt{C}\langle u\rangle \in \mathcal{WN}_{\mathtt{whnd}} & \text{iff} & \text{(definition)} \\
t \cong_{\mathtt{whnd}} u & & &
\end{array}
$$

□

10 Conclusion

We establish a clear connection between the semantical standard notion of neededness and the syntactical concept of call-by-need. The use of non-idempotent types –a powerful technique being able to characterise different operational properties– provides a simple and natural tool to show observational equivalence between these two notions. We refer the reader to [5] for other proof techniques (not based on intersection types) used to connect semantical notions of neededness with syntactical notions of lazy evaluation.

An interesting (and not difficult) extension of our result in Sec. 6 is that call-by-need reduction (defined on λ-terms with explicit substitutions) contracts only $\mathtt{dB}$ weak-head needed redexes, for an appropriate (and very natural) notion of weak-head needed redex for λ-terms with explicit substitutions. A technical tool to obtain such a result would be the type system $\mathcal{A}$ [14], a straightforward adaptation of system $\mathcal{V}$ to call-by-need syntax.

Given the recent formulation of *strong call-by-need* [6] describing a deterministic call-by-need strategy to normal form (instead of weak-head normal form), it would be natural to extend our technique to obtain an observational equivalence result between the standard notion of needed reduction (to full normal forms) and the strong call-by-need strategy. This remains as future work.

References

1. Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In Johan Jeuring and Manuel M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 363–376. ACM, 2014.
2. Zena M. Ariola and Matthias Felleisen. The call-by-need lambda calculus. *J. Funct. Program.*, 7(3):265–301, 1997.
3. Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. In Ron K. Cytron and Peter Lee, editors, *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, pages 233–246. ACM Press, 1995.
4. Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.
5. Thibaut Balabonski. *La pleine paresse, une certaine optimalité.* Ph.D. Thesis, Université Paris-Diderot, 2012.
6. Thibaut Balabonski, Pablo Barenbaum, Eduardo Bonelli, and Delia Kesner. Foundations of strong call by need. *PACMPL*, 1(ICFP):20:1–20:29, 2017.
7. Hendrik P. Barendregt. *The Lambda Calculus Its Syntax and Semantics*, volume 103. North Holland, revised edition, 1984.
8. Hendrik P. Barendregt, Richard Kennaway, Jan Willem Klop, and M. Ronan Sleep. Needed reduction and spine strategies for the lambda calculus. *Inf. Comput.*, 75(3):191–231, 1987.
9. Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.
10. Stephen Chang and Matthias Felleisen. The call-by-need lambda calculus, revisited. In Helmut Seidl, editor, *Programming Languages and Systems - 21st European Symposium on Programming, ESOP 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, volume 7211 of *Lecture Notes in Computer Science*, pages 128–147. Springer, 2012.
11. Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.
12. Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul.* PhD thesis, Université Aix-Marseille II, 2007.
13. Philippa Gardner. Discovering needed reductions using type theory. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 1994.
14. Delia Kesner. Reasoning about call-by-need by means of types. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 2016.

15. Delia Kesner, Alejandro Ríos, and Andrés Viso. Call-by-need, neededness and all that. Extended report, 2017. https://arxiv.org/abs/1801.10519.
16. John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. *J. Funct. Program.*, 8(3):275–317, 1998.
17. Simona Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theor. Comput. Sci.*, 59:181–209, 1988.
18. Steffen van Bakel. Complete restrictions of the intersection type discipline. *Theor. Comput. Sci.*, 102(1):135–163, 1992.
19. Pierre Vial. *Non-Idempotent Intersection Types, Beyond Lambda-Calculus*. PhD thesis, Université Paris-Diderot, 2017.
20. Christopher P. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. PhD thesis, Oxford University, 1971.