Quantitative Types for the Linear Substitution Calculus

Delia Kesner, Daniel Ventura

Quantitative Types for the Linear Substitution Calculus

Delia Kesner[1] and Daniel Ventura[2]

[1] Univ. Paris-Diderot, SPC, PPS, CNRS, France
[2] Univ. Federal de Goiás, INF, Brasil

Abstract. We define two non-idempotent intersection type systems for the linear substitution calculus, a calculus with partial substitutions acting at a distance that is a computational interpretation of linear logic proof-nets. The calculus naturally express *linear-head reduction*, a notion of evaluation of proof nets that is strongly related to abstract machines. We show that our first (resp. second) quantitave type system characterizes linear-head, head and weak (resp. strong) normalizing sets of terms. All such characterizations are given by means of combinatorial arguments, *i.e.* there is a measure based on type derivations which decreases with respect to each reduction relation considered in the paper.

1 Introduction

It is quite difficult to reason about programs by only taking into account their syntax, so that many different semantic approaches were proposed to analyze them in a more abstract way. One typical tool to analyze relevant aspects of programs is the use of *type systems*. In particular, *intersection types* allow to characterize *head/weakly/strongly* normalizing terms, *i.e.* a term t is typable in an intersection type system iff t is head/weakly/strongly normalizing; *quantitative* information about the behaviour of programs can also be obtained if the intersection types enjoy *non-idempotence*.

Intersection Types (IT): *Simply typed* terms are *strongly normalizing* (*cf.* [7]) but the converse does not hold, *e.g.* the term $t := \lambda x.xx$. *Intersection Types* [15] extend the simply typed discipline with a finitary notion of polymorphism, listing type usage, that exactly captures the set of strongly normalizing terms. This is done by introducing a new constructor of types $\wedge$ together with a corresponding set of typing rules. For instance, the previous term t is typable with $((\sigma \to \sigma)\wedge\sigma) \to \sigma$ so that the first (resp. second) occurrence of the variable x is typed with $\sigma \to \sigma$ (resp. σ). Typically, intersection types are *idempotent*, *i.e.* $\sigma \wedge \sigma = \sigma$. Moreover, the intersection constructor is usually *commutative* and *associative*. Intersection types in their full generality provide a characterization of various properties of terms: models of the λ-calculus [8], characterization of head [17] as well as weakly [13, 17] and strongly normalizing terms [33].

Non-Idempotent Intersection Types: The use of non-idempotent types [11] gives rise to resource aware semantics, which is suitable for computational complexity since it allows to extract *quantitative* information about reduction sequences. Indeed, the inequality $\sigma \wedge \sigma \neq \sigma$ can be read as the fact that two different uses of the variable x are not isomorphic to a single use. Relationship with Linear Logic [24] and Relevant Logic [23, 18] provides an insight on the information refinement aspect of non-idempotent intersection types. The relation between the size of a non-idempotent intersection typing derivation and the *head/weak*-normalization execution time of λ-terms by means of abstract machines was established by D. de Carvalho [21]. Non-idempotence is also used in [9, 20] to reason about the longest derivation of *strongly* β-normalizing terms in the λ-calculus by means of combinatorial arguments.

Calculi with Explicit Substitutions (ES) and Intersection Types: Calculi with ES refine the λ-calculus by decomposing β-reduction into small steps in order to specify different evaluation strategies implemented by abstract machines. In traditional calculi with ES [1], the operational semantics specifies the propagation of ES through the structure of the term until they reach a variable occurrence, on which they finally substitute or get garbage collected. But calculi with ES can also be interpreted in *Linear Logic* [22, 28, 26, 5] by implementing another kind of operational semantics: their dynamics is defined using contexts (*i.e.* terms with holes) that allows the ES to act directly *at a distance* on single variable occurrences, with no need to commute with any other constructor in between. In other words, the propagation of substitutions is not performed by structural induction on terms, since they are only consumed according to the multiplicity of the variables.

Idempotent intersection type systems were used to characterize strongly normalizing terms of calculi with ES [34, 27] while non-idempotence is used in [10] to prove the exact relationship between typing derivations and the number of steps of the longest reduction sequence of strongly-normalizing terms in the $\lambda\mathtt{s}$-calculus [26] and in the $\lambda\mathtt{lxr}$-calculus [28]. No study about linear-head, head and weak normalizing is provided in those works. Moreover, the systems are not *syntax-directed*, *i.e.* not all the typing derivations of t end with the same typing rule. As a consequence, the formal developments of proofs require a generation lemma which guarantees the existence of some typing derivations having a particular shape. This drawback makes the development of proofs more involved.

Contribution: This paper focuses on *functional programs* specified – via the Curry-Howard isomorphism – by *intuitionistic logic*, in *natural deduction* style. The operational semantics implements resource control by means of reduction rules describing the behaviour of *explicit* operators for *erasure* and *duplication*. The term language is the *linear substitution calculus* [3], called here `M`-*calculus*, and obtained from Milner's calculus [36] and the structural λ-calculus [5].

Partial substitution allows to express *linear-head reduction* [19, 35], a notion of evaluation of proof nets that is strongly related to significant aspects of computer science [32, 2, 4]. Linear-head reduction cannot be expressed as a simple strategy of the λ-calculus, where substitution acts on all free occurrences of a variable at once; this is probably one of the reasons why there are so few works investigating it. In this paper we use logical systems to reason about different notions of normalization of terms, including those obtained with linear-head reduction.

More precisely, the quantitative semantics of programs used in this paper is given by two non-idempotent intersection type systems. The first one, based on [21], allows a characterization of linear-head, head and weakly normalizing terms. While full logical characterizations of head/weakly β-normalizing λ-terms were already given in the literature, the use of a logical/type system to directly characterize linear-head normalization in calculi with ES is new. The second system, another main contributions of this paper, gives a characterization of strongly normalizing terms.

Our type systems use multiset notation and are syntax-directed so that no generation lemmas are needed, thus making the development of proofs much more direct. Moreover, the type systems for strong normalization make use of a special notion of *witness* derivation for the arguments (of applications and explicit substitutions) which makes them particularly natural. All the characterizations in the paper are given by means of simple combinatorial arguments, *i.e.* there is a measure that can be associated to each typing derivation which is decreasing with respect to the different reduction relations considered in the paper.

Structure of the paper: Sec. 2 presents the syntax and semantics of the `M`-calculus and both typing systems. Sec. 3 presents the Linear-Head, Head and Weak-Normalization characterizations while Sec. 4 presents the Strong-Normalization characterization. We then conclude in Sec. 5.

2 The Linear Substitution Calculus

We first describe the syntax and the operational semantics of the $\mathtt{M}$-calculus, including some particular notions of rewriting such as linear-head reduction. We then introduce a notion of type and two different type systems that play a central role in the first part of the paper.

Syntax: Given a countable infinite set of symbols $x, y, z, \ldots$, three different syntactic categories for terms ($\mathcal{T}_{\mathtt{M}}$) and contexts ($\mathcal{C}_{\mathtt{M}}$) are defined by the following grammars:

$$\begin{array}{rll} (\textbf{terms}) & t, u, v & ::= x \mid tt \mid \lambda x.t \mid t[x/t] \\ (\textbf{term contexts}) & \mathtt{C} & ::= \Box \mid \lambda x.\mathtt{C} \mid \mathtt{C}\ t \mid t\ \mathtt{C} \mid \mathtt{C}[x/t] \mid t[x/\mathtt{C}] \\ (\textbf{list contexts}) & \mathtt{L} & ::= \Box \mid \mathtt{L}[x/t] \end{array}$$

A term x is called a **variable**, tu an **application**, $\lambda x.t$ an **abstraction** and $t[x/u]$ a **closure** where $[x/u]$ is an **explicit substitution**. We write $tt_1 \ldots t_n$ for $(\ldots(tt_1)\ldots t_n)$. The notions of **free** and **bound** variables are defined as usual, in particular, $\mathtt{fv}(t[x/u]) := \mathtt{fv}(t) \setminus \{x\} \cup \mathtt{fv}(u)$, $\mathtt{fv}(\lambda x.t) := \mathtt{fv}(t) \setminus \{x\}$, $\mathtt{bv}(t[x/u]) := \mathtt{bv}(t) \cup \{x\} \cup \mathtt{bv}(u)$ and $\mathtt{bv}(\lambda x.t) := \mathtt{bv}(t) \cup \{x\}$. We work with the standard notion of α-conversion *i.e.* renaming of bound variables for abstractions and substitutions. We write $\mathtt{C}[t]$ (resp. $\mathtt{L}[t]$) for the term obtained by replacing the hole of $\mathtt{C}$ (resp. $\mathtt{L}$) by the term t. We write $\mathtt{C}[\![u]\!]$ or $\mathtt{L}[\![u]\!]$ 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. The set of **positions** of t, written $\mathtt{pos}(t)$, is the finite language over $\{0, 1\}$ inductively defined as follows: $\epsilon \in \mathtt{pos}(t)$ for every t; $0p \in \mathtt{pos}(\lambda x.t)$ if $p \in \mathtt{pos}(t)$; $0p \in \mathtt{pos}(tu)$ (resp. $\mathtt{pos}(t[x/u])$) if $p \in \mathtt{pos}(t)$; $1p \in \mathtt{pos}(tu)$ (resp. $\mathtt{pos}(t[x/u])$) if $p \in \mathtt{pos}(u)$. The **subterm of** t **at position** p is written $t|_p$ and defined as expected. The term u **has an occurrence** in t iff there is $p \in \mathtt{pos}(t)$ such that $t|_p = u$. We write $|t|_x$ to denote the **number of free occurrences** of the variable x in the term t. All these notions are extended to contexts as expected.

Operational Semantics: The $\mathtt{M}$**-calculus** is given by the set of terms $\mathcal{T}_{\mathtt{M}}$ and the **reduction relation** $\rightarrow_{\mathtt{dB}\cup\mathtt{c}\cup\mathtt{w}}$, the **union** of $\rightarrow_{\mathtt{dB}}$, $\rightarrow_{\mathtt{c}}$, and $\rightarrow_{\mathtt{w}}$, denoted by $\rightarrow_{\mathtt{M}}$, which are, respectively, the closure by term contexts $\mathtt{C}$ of the following rewriting rules:

$$\begin{array}{lll} \mathtt{L}[\lambda x.t]u & \mapsto_{\mathtt{dB}} \mathtt{L}[t[x/u]] & \\ \mathtt{C}[\![x]\!][x/u] & \mapsto_{\mathtt{c}} \mathtt{C}[\![u]\!][x/u] & \\ t[x/u] & \mapsto_{\mathtt{w}} t & \text{if } |t|_x = 0 \end{array}$$

The names $\mathtt{dB}$, $\mathtt{c}$ and $\mathtt{w}$ stand for $\mathtt{d}$*istant* $\mathtt{B}$*eta*, $\mathtt{c}$*ontraction* and $\mathtt{w}$*eakening*, respectively. Rule $\mapsto_{\mathtt{dB}}$ (resp. $\mapsto_{\mathtt{c}}$) comes from the structural λ-calculus [5] (resp. Milner's calculus [36]), while $\mapsto_{\mathtt{w}}$ belongs to both calculi. By α-conversion we can assume in the rule $\mathtt{dB}$ that x may only be free in t and no variable in the domain of $\mathtt{L}$, defined as expected, has free occurrences in the term u. The *pushed out* list context $\mathtt{L}$ in rule $\mathtt{dB}$ can be obtained by using an equivalence related to Regnier's σ-equivalence [38]: $\mathtt{L}[\lambda x.t]u \sim_{\sigma} \mathtt{L}[(\lambda x.t)u] \rightarrow_{\mathtt{dB}} \mathtt{L}[t[x/u]]$. We will come back on this equivalence in Sec. 4.

The reflexive-transitive (resp. transitive) closure of $\rightarrow_{\mathtt{M}}$ is denoted by $\rightarrow^{*}_{\mathtt{M}}$ (resp. $\rightarrow^{+}_{\mathtt{M}}$). Given $t\in\mathcal{T}_{\mathtt{M}}$, t is **in $\mathtt{M}$-normal form**, written $t\in\mathtt{M}$-nf, if there is no t' s.t. $t \rightarrow_{\mathtt{M}} t'$; and t **has an $\mathtt{M}$-nf** iff there is $t'\in\mathtt{M}$-nf such that $t \rightarrow^{*}_{\mathtt{M}} t'$. Moreover, t is **weakly $\mathtt{M}$-normalizing**, written $t\in\mathcal{WN}(\mathtt{M})$, iff t has an $\mathtt{M}$-nf, t is **strongly $\mathtt{M}$-normalizing** or $\mathtt{M}$-terminating, written $t\in\mathcal{SN}(\mathtt{M})$, if there is no infinite $\mathtt{M}$-reduction sequence starting at t. Every $\mathtt{M}$-term is $(\mathtt{c},\mathtt{w})$-strongly normalizing [29].

The notion of *redex occurrence* in this calculus is more subtle than the one in standard rewriting because one unique term may give rise to different reduction steps at the root, *e.g.* $(xu)[x/u] \ {}_{\mathtt{c}}\!\leftarrow (xx)[x/u] \rightarrow_{\mathtt{c}} (ux)[x/u]$. Thus, given $p \in \mathtt{pos}(t)$, p is said to be a **$\mathtt{dB}$-redex occurrence** of t if $t|_p = \mathtt{L}[\lambda x.t]u$, p is a **$\mathtt{w}$-redex occurrence** of t if $t|_p = v[x/u]$ with $|v|_x = 0$, and p is a **$\mathtt{c}$-redex occurrence** of t if $p = p_1p_2$, $t|_{p_1} = \mathtt{C}[\![x]\!][x/u]$ and $\mathtt{C}|_{p_2} = \Box$. For example 000 and 001 are both $\mathtt{c}$-redex occurrences of the term $\lambda z.(xx)[x/u]$.

The $\mathtt{M}$-calculus enjoys good properties required for calculi with ES (including simulation of β-reduction, preservation of strong normalization, confluence on terms and metaterms and full composition) [29]. It was recently used in different investigations of computer science [4, 2, 3].

The reduction relation $\rightarrow_{\mathtt{M}}$ can be refined in different ways, where the (reflexive-)transitive closures and normal-forms are defined as expected. The **non-erasing** reduction relation $\rightarrow_{\mathtt{M}\setminus\mathtt{w}}$ is given by $\rightarrow_{\mathtt{dB}\cup\mathtt{c}}$, and plays a key role in the characterization of strongly normalizing terms in Sec. 4. Another key subrelation studied in this paper is *linear-head reduction* [19, 35], a strategy related to abstract machines [19] and linear logic [24]. To introduce this notion, we first define the set of **linear-head contexts** that are generated by the following grammar:

$$\mathtt{L_H} ::= \Box \mid \lambda x.\mathtt{L_H} \mid \mathtt{L_H}t \mid \mathtt{L_H}[x/t]$$

Linear-head $\mathtt{M}$-reduction, written $\rightarrow_{\mathtt{L_H M}}$, is the closure under *linear-head contexts* of the rewriting rules $\{\mapsto_{\mathtt{dB}}, \mapsto_{\mathtt{c}|_{\mathtt{L_H}}}\}$, where $\mapsto_{\mathtt{c}|_{\mathtt{L_H}}}$ is the following variation of the rewriting rule $\mapsto_{\mathtt{c}}$:

$$\mathtt{L_H}[\![x]\!]x[x/u] \mapsto_{\mathtt{c}|_{\mathtt{L_H}}} \mathtt{L_H}[\![u]\!][x/u]$$

Indeed, the leftmost (*i.e. head*) occurrence of the variable x in $\mathtt{L_H}[\![x]\!]$ is substituted by u and this partial (*i.e. linear*) substitution is only performed on that head occurrence. The notion of $\mathtt{c}|_{\mathtt{L_H}}$**-redex occurrence** is defined as for the $\mathtt{c}$-rule. A term t is **linear-head** $\mathtt{M}$**-normalizing**, written $t \in \mathcal{L_H N}(\mathtt{M})$, iff t has an $\mathtt{L_H}\mathtt{M}$-nf. For example, if $t_0 := \lambda x.xy$ and $t_1 := x[y/z](\mathtt{II})$, where $\mathtt{I} := \lambda w.w$, then $t_0 \in \mathtt{M}$-nf, and so also $t_0 \in \mathtt{L_H}\mathtt{M}$-nf, while $t_1 \notin \mathtt{M}$-nf but $t_1 \in \mathtt{L_H}\mathtt{M}$-nf.

Types: We denote finite multisets by brackets, so that $[\,]$ denotes the empty multiset; $[a, a, b]$ denotes a multiset having two occurrences of the element a and one occurrence of b. We use $+$ for multiset union. Given a countable infinite set of *base types* $\alpha, \beta, \gamma, \ldots$ we consider **types** and **multiset types** defined by the following grammars:

$$\begin{array}{lll} (\textbf{types}) & \tau, \sigma, \rho ::= \alpha \mid \mathcal{M} \to \tau & \\ (\textbf{multiset types}) & \mathcal{M} \quad ::= [\tau_i]_{i \in I} & \text{where } I \text{ is a finite set} \end{array}$$

Observe that our types are *strict* [16, 6], *i.e.* the type on the right hand side of a functional type is never a multiset. They also make use of usual notations for multisets, as in [21], so that $[\sigma, \sigma, \tau]$ must be understood as $\sigma \wedge \sigma \wedge \tau$, where the symbol $\wedge$ is defined to enjoy commutative and associative axioms. When $\wedge$ verifies the axiom $\sigma \wedge \sigma = \sigma$, the underlying type system is **idempotent**, otherwise, like in this paper, it is **non-idempotent**.

Type assignments, written Γ, Δ, are functions from variables to multiset types, assigning the empty multiset to all but a finite set of the variables. The domain of Γ is given by $\mathtt{dom}(\Gamma) := \{x \mid \Gamma(x) \neq [\,]\}$. The **intersection of type assignments**, written $\Gamma + \Delta$, is defined by $(\Gamma + \Delta)(x) := \Gamma(x) + \Delta(x)$, where the symbol $+$ denotes multiset union. As a consequence $\mathtt{dom}(\Gamma + \Delta) = \mathtt{dom}(\Gamma) \cup \mathtt{dom}(\Delta)$. When $\mathtt{dom}(\Gamma)$ and $\mathtt{dom}(\Gamma)$ are disjoint we write $\Gamma; \Delta$ instead of $\Gamma + \Delta$. We write $\Gamma \setminus\!\!\setminus x$ for the assignement $(\Gamma \setminus\!\!\setminus x)(x) = [\,]$ and $(\Gamma \setminus\!\!\setminus x)(y) = \Gamma(y)$ if $y \neq x$.

The Type Systems: Type judgments have the form $\Gamma \vdash t{:}\tau$, where Γ is a type assignment, t is a term and τ is a type. The type systems $\mathcal{HW}$, after $\mathcal{H}$ead-$\mathcal{W}$eak, and $\mathcal{S}$, after $\mathcal{S}$trong, for the $\mathtt{M}$-calculus are given respectively in Fig. 1 and 2. A **(typing) derivation** in system X is a tree obtained by applying the (inductive) typing rules of system X. The notation $\Gamma \vdash_X t{:}\tau$ means there is a derivation of the judgment $\Gamma \vdash t{:}\tau$ in system X. The term t is **typable** in system X, or X**-typable**, iff there

$$\frac{}{x{:}[\tau] \vdash x{:}\tau}\ (\mathtt{ax}) \qquad \frac{x{:}[\sigma_i]_{i\in I}; \Gamma \vdash t{:}\tau \qquad (\Delta_i \vdash u{:}\sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash t[x/u]{:}\tau}\ (\mathtt{cut}_{\mathcal{HW}})$$

$$\frac{\Gamma \vdash t{:}\tau}{\Gamma \setminus\!\!\setminus x \vdash \lambda x.t{:}\Gamma(x) \to \tau}\ (\to \mathtt{i}) \qquad \frac{\Gamma \vdash t{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash u{:}\sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash tu{:}\tau}\ (\to \mathtt{e}_{\mathcal{HW}})$$

Fig. 1. The Type System $\mathcal{HW}$ for the M-Calculus

Typing Rules $\{(\mathtt{ax}), (\to \mathtt{i})\}$ plus

$$\frac{x{:}[\sigma_i]_{i\in I}; \Gamma \vdash t{:}\tau \qquad (\Delta_i \vdash u{:}\sigma_i)_{i\in I\cup\{w\}}}{\Gamma +_{i\in I\cup\{w\}} \Delta_j \vdash t[x/u]{:}\tau}\ (\mathtt{cut}_{\mathcal{S}})$$

$$\frac{\Gamma \vdash t{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash u{:}\sigma_i)_{i\in I\cup\{w\}}}{\Gamma +_{i\in I\cup\{w\}} \Delta_i \vdash tu{:}\tau}\ (\to \mathtt{e}_{\mathcal{S}})$$

Fig. 2. The Type System $\mathcal{S}$ for the M-Calculus

are Γ and τ s.t. $\Gamma \vdash_X t{:}\tau$. We use the capital Greek letters $\Phi, \Psi, \ldots$ to name type derivations, *e.g.* we write $\Phi \triangleright \Gamma \vdash_X t{:}\tau$. The **size** of a type derivation Φ is a positive natural number $\mathtt{sz}(\Phi)$ defined as expected.

The rules $(\mathtt{ax})$, $(\to \mathtt{i})$ and $(\to \mathtt{e}_{\mathcal{HW}})$ in system $\mathcal{HW}$ come from a relational semantics for linear logic [21]. Remark in particular that the axiom is *relevant* (so there is no weakening) and the rules for application and substitution are *multiplicative*, both characteristics are related to the *resource aware* semantics. A particular case of rule $(\to \mathtt{e}_{\mathcal{HW}})$ is when $I = \emptyset$: the subterm u occuring in the *typed* term tu turns out to be *untyped*. Thus for example, from the derivation $x{:}[\sigma] \vdash_{\mathcal{HW}} \lambda y.x{:}[\,] \to \sigma$ we can construct $x{:}[\sigma] \vdash_{\mathcal{HW}} (\lambda y.x)\Omega{:}\sigma$, where Ω is the non-terminating term $(\lambda z.zz)(\lambda z.zz)$. This is precisely the reason why rules $(\to \mathtt{e}_{\mathcal{S}})$ and $(\mathtt{cut}_{\mathcal{S}})$ in Fig. 2, the system which characterizes strongly-normalizing terms, always asks a **witness** typing derivation for the arguments of applications and substitutions. Indeed, if $I = \emptyset$, then the argument u will be typed with the witness derivation $\Delta_w \vdash u{:}\sigma_w$, whatever the type σ_w is. This witness derivation for u is essential to guarantee strong-normalization of u (and thus of tu and $t[x/u]$). When $I \neq \emptyset$ the rules $(\to \mathtt{e}_{\mathcal{S}})$ and $(\mathtt{cut}_{\mathcal{S}})$ also require a witness derivation for u, whose use is necessary in order to deal with the c-rule when $|\mathtt{C}[\![x]\!]|_x = 1$ (see discussion after Lem. 4). Last, remark that an alternative definition of rules $(\to \mathtt{e}_{\mathcal{S}})$ and $(\mathtt{cut}_{\mathcal{S}})$ given by adding $I \neq \emptyset$ to rules $(\to \mathtt{e}_{\mathcal{HW}})$ and $(\mathtt{cut}_{\mathcal{HW}})$, respectively, would not be complete: terms like $x[y/z]$ or $(\lambda y.x)z$ become untypable.

Given $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\sigma$, not every free variable of t necessarily appears in the domain of Γ, this is for example the case in $x{:}[\sigma] \vdash_{\mathcal{HW}} (\lambda y.x)z{:}\sigma$. More precisely, the systems enjoy the following (weak/strong) relevance properties, that can be easily shown by induction on derivations.

Lemma 1. *If* $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\sigma$ *then* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$. *If* $\Phi \triangleright \Gamma \vdash_{\mathcal{S}} t{:}\sigma$, *then* $\mathtt{dom}(\Gamma) = \mathtt{fv}(t)$.

In contrast to other intersection type systems for ES in the literature, the typing rules of our systems are syntax oriented, so that generation lemmas are not needed to distinguish particular syntactical forms of derivations.

3 About Linear-Head, Head and Weak M-Normalization

We show in this section two main results. The first one (Sec. 3.1) characterizes linear-head and head $\mathtt{M}$-normalizing terms by means of $\mathcal{HW}$-typability. This result generalizes to calculi with ES the well-known *logical* characterization of *head* β-normalizing λ-terms [17, 21]. The $\mathcal{HW}$-type system is known to type also some *non* weakly $\mathtt{M}$-normalizing terms: for instance, if Ω is any non-terminating term, then $x{:}[\,]\to\sigma \vdash_{\mathcal{HW}} x\Omega{:}\sigma$. We then characterize the set of weakly $\mathtt{M}$-normalizing terms, our second result (Sec. 3.2), by restricting the $\mathcal{HW}$-typing derivations to some particular ones. But first, let us develop some key technical tools.

To understand which are the redex occurrences actually constrained by the type system, let us consider a derivation $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\tau$. A position $p \in \mathtt{pos}(t)$ is a **typed occurrence** of Φ if either $p = \epsilon$, or $p = ip'$ $(i = 0, 1)$ and $p' \in \mathtt{pos}(t|_i)$ is a typed occurrence of *some* of their corresponding subderivations of Φ. A redex occurrence of t which is also a typed occurrence of Φ is a **redex T-occurrence** of t in Φ. Thus for example, given the following derivations Φ and Φ', we have that ϵ, 0, 1 and 10 are T-occurrences in Φ and Φ', while 11 is a T-occurrence in Φ but not in Φ'.

$$\Phi \triangleright \frac{\dfrac{}{x{:}[[\tau,\tau]\to\tau] \vdash x{:}[\tau,\tau]\to\tau} \quad \dfrac{y{:}[[\,]\to\tau] \vdash y{:}[\,]\to\tau}{y{:}[[\,]\to\tau] \vdash yz{:}\tau} \quad \dfrac{y{:}[[\tau]\to\tau] \vdash y{:}[\tau]\to\tau \qquad z{:}[\tau] \vdash z{:}\tau}{y{:}[[\tau]\to\tau], z{:}[\tau] \vdash yz{:}\tau}}{x{:}[[\tau,\tau]\to\tau], y{:}[[\,]\to\tau,[\tau]\to\tau], z{:}[\tau] \vdash x(yz){:}\tau}$$

$$\Phi' \triangleright \frac{\dfrac{}{x{:}[[\tau,\tau]\to\tau] \vdash x{:}[\tau,\tau]\to\tau} \quad \dfrac{y{:}[[\,]\to\tau] \vdash y{:}[\,]\to\tau}{y{:}[[\,]\to\tau] \vdash yz{:}\tau} \quad \dfrac{y{:}[[\,]\to\tau] \vdash y{:}[\,]\to\tau}{y{:}[[\,]\to\tau] \vdash yz{:}\tau}}{x{:}[[\tau,\tau]\to\tau], y{:}[[\,]\to\tau,[\,]\to\tau] \vdash x(yz){:}\tau}$$

The notion of T-occurrence plays a key role in the Subject Reduction (SR) lemma, which is based on a subtle *partial* substitution lemma, a refinement of the standard substitution lemma used in the λ-calculus.

Lemma 2 (SR I). *Let $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\tau$. If $t \rightarrow_{\mathtt{M}} t'$ reduces a $(\mathtt{dB}, \mathtt{c}, \mathtt{w})$-redex T-occurrence of t in Φ then $\Phi' \triangleright \Gamma \vdash_{\mathcal{HW}} t'{:}\tau$ and $\mathtt{sz}(\Phi) > \mathtt{sz}(\Phi')$.*

As an example, consider $\Phi'' \triangleright y{:}[[\,] \rightarrow [\,] \rightarrow \tau] \vdash_{\mathcal{HW}} (xxx)[x/y]{:}\tau$. Then the (typed) reduction step $(xxx)[x/y] \rightarrow_{\mathtt{c}} (yxx)[x/y]$ decreases the measure of Φ'' but thereafter $(yxx)[x/y] \rightarrow_{\mathtt{c}} (yyx)[x/y] \rightarrow_{\mathtt{c}} (yyy)[x/y]$ are not decreasing reduction steps since they act on untyped occurrences.

As a corollary, termination holds for *any* strategy reducing only redexes T-occurrences, an important key point used in Sec. 3.1 and 3.2.

Corollary 1. *If $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\tau$, then any $\mathtt{M}$-reduction sequence contracting only $(\mathtt{dB}, \mathtt{c}, \mathtt{w})$-redex T-occurrences is finite.*

Types of terms can also be recovered by means of Subject Expansion (SE), a property which will be particularly useful in Sec. 3.1 and 3.2.

Lemma 3 (SE I). *If $\Gamma \vdash_{\mathcal{HW}} t'{:}\tau$ and $t \rightarrow_{\mathtt{M}} t'$ then $\Gamma \vdash_{\mathcal{HW}} t{:}\tau$.*

3.1 Linear-Head and Head M-Normalization

Linear-head reduction [19, 35] comes from a fine notion of evaluation for proof nets [25]. It is a particular reduction strategy of the M-calculus although it is not a strategy of β-reduction. In contrast to *head*-reduction for λ-calculus the reduction relation $\rightarrow_{\mathtt{L_H M}}$ for M-terms is non-deterministic: $y[y/w][x/z] \;{}_{\mathtt{L_H M}}\!\!\leftarrow (\lambda x.y[y/w])z \rightarrow_{\mathtt{L_H M}} (\lambda x.w[y/w])z$. This behaviour is however safe since $\rightarrow_{\mathtt{L_H M}}$ has the diamond property [7].

Another remarkable property of linear-head reduction is that the hole of the contexts $\mathtt{L_H}$ cannot be duplicated nor erased. This is related to a recent result [3] stating that linear-head reduction is *standard* for the M-calculus, exactly as *left-to-right* reduction is standard for the λ-calculus.

We now refine a known result in the λ-calculus which characterizes *head*-normalizing terms by means of intersection types, either idempotent [17, 8][3] or non-idempotent [21]. Indeed, the set of linear-head M-normalizing terms coincides with the set of $\mathcal{HW}$-typable terms.

Lemma 4. *If $\Phi \triangleright \Gamma \vdash_{\mathcal{HW}} t{:}\tau$ and t has no $(\mathtt{dB}, \mathtt{c}|_{\mathtt{L_H}})$-redexes T-occurrences in Φ, then $t \in \mathtt{L_H M}$-nf.*

[3] Although idempotency was not explicity mentioned in [17], a remark on pp. 55 points out the meaninglessness of duplication of types in a sequence.

It is worth noticing that Lem. 4 does not hold for *head*-nfs. Indeed, the term $(yxx)[x/y]$ in the example just after Lem. 2 does not have any redex T-occurrence (the only two $\mathtt{c}$-redexes occurrences are untyped), and is not a *head*-nf. This emphasizes the fact that linear-head reduction is more pertinent for calculi with ES than head reduction. We conclude by

Theorem 1. *Let $t \in \mathcal{T}_{\mathtt{M}}$. Then $t \in \mathcal{L_H N}(\mathtt{M})$ iff t is $\mathcal{HW}$-typable.*

Proof. Let $t \in \mathcal{L_H N}(\mathtt{M})$. We proceed by induction on the length of the linear-head $\mathtt{M}$-normalizing reduction using Lem. 3 (see [30] for details).

Let t be $\mathcal{HW}$-typable. By Cor. 1 the strategy consisting in the contraction of $(\mathtt{dB}, \mathtt{c}|_{\mathtt{L_H}})$-redex T-occurrences terminates in a term t' without such redexes. The term t' is typable by Lem. 2 and t' is a $\mathtt{L_H M}$-nf by Lem. 4. Thus, $t \in \mathcal{L_H N}(\mathtt{M})$.

We can finally characterize head-normalization. A term t is **head-normalizing**, written $t \in \mathcal{HN}(\mathtt{M})$, iff t $\mathtt{M}$-reduces to a term of the form $\lambda x_1 \ldots x_n.yu_1 \ldots u_m$ for some $n \geq 0, m \geq 0$.

Theorem 2. *Let $t \in \mathcal{T}_{\mathtt{M}}$. Then $t \in \mathcal{HN}(\mathtt{M})$ iff t is $\mathcal{HW}$-typable.*

Proof. For the if implication we have $\mathcal{HN}(\mathtt{M}) \subseteq \mathcal{L_H N}(\mathtt{M})$ so we conclude by Thm. 1. Otherwise, t $\mathcal{HW}$-typable implies by Thm. 1 that $t \to^*_{\mathtt{M}} t'$, where $t' \in \mathtt{L_H M}$-nf. The (terminating) reduction relation $\to_{(\mathtt{c},\mathtt{w})}$ on t' gives a term of the required form.

3.2 Weak $\mathtt{M}$-Normalization

In this section we use the type system $\mathcal{HW}$ to characterize weakly $\mathtt{M}$-normalizing terms, a result that extends the well-known characterization [17] of weakly β-normalizing in the λ-calculus. As in [17, 13], $\mathcal{HW}$-typability alone does not suffice to characterize weak $\mathtt{M}$-normalizing terms (see an example at the beginning of Sec. 3). The type $[\,]$ plays a similar rôle to the universal ω type in [17, 13], although it is restricted to occur only in the domain type of a function that accepts any kind of argument. We then restrict the allowed typing derivations in order to recover such a characterization. Indeed, the set of **positive** (resp. **negative**) subtypes of a type is the smallest set satisfying the following conditions (*cf.*[13]).

- $A \in \mathcal{P}(A)$.
- $A \in \mathcal{P}([\sigma_i]_{i\in I})$ if $\exists i\ A \in \mathcal{P}(\sigma_i)$; $A \in \mathcal{N}([\sigma_i]_{i\in I})$ if $\exists i\ A \in \mathcal{N}(\sigma_i)$.
- $A \in \mathcal{P}(\mathcal{M} \to \tau)$ if $A \in \mathcal{N}(\mathcal{M})$ or $A \in \mathcal{P}(\tau)$; $A \in \mathcal{N}(\mathcal{M} \to \tau)$ if $A \in \mathcal{P}(\mathcal{M})$ or $A \in \mathcal{N}(\tau)$.

- $A \in \mathcal{P}(\Gamma)$ if $\exists\, y \in \mathtt{dom}(\Gamma)$ s.t. $A \in \mathcal{N}(\Gamma(y))$; $A \in \mathcal{N}(\Gamma)$ if $\exists\, y \in \mathtt{dom}(\Gamma)$ s.t. $A \in \mathcal{P}(\Gamma(y))$.
- $A \in \mathcal{P}(\langle \Gamma, \tau\rangle)$ if $A \in \mathcal{P}(\Gamma)$ or $A \in \mathcal{P}(\tau)$; $A \in \mathcal{N}(\langle \Gamma, \tau\rangle)$ if $A \in \mathcal{N}(\Gamma)$ or $A \in \mathcal{N}(\tau)$.

As an example, $[\,] \in \mathcal{P}([\,])$, so that $[\,] \in \mathcal{N}([\,] \to \sigma)$, $[\,] \in \mathcal{P}(x{:}[[\,] \to \sigma])$ and $[\,] \in \mathcal{P}(\langle x{:}[[\,] \to \sigma], \sigma\rangle)$.

Lemma 5. *Let $\Phi \rhd \Gamma \vdash_{\mathcal{HW}} t{:}\tau$ s.t. $[\,] \notin \mathcal{P}(\langle \Gamma, \tau\rangle)$. If t has no $(\mathtt{dB}, \mathtt{c}, \mathtt{w})$-redex T-occurrences in Φ, then $t \in \mathtt{M}$-nf.*

Theorem 3. *Let $t \in \mathcal{T}_{\mathtt{M}}$. Then, $t \in \mathcal{WN}(\mathtt{M})$ iff $\Gamma \vdash_{\mathcal{HW}} t{:}\tau$ and $[\,] \notin \mathcal{P}(\langle \Gamma, \tau\rangle)$.*

Proof. If $t \in \mathcal{WN}(\mathtt{M})$, we proceed by induction on the length of the $\mathtt{M}$-normalizing reduction sequence using Lem. 3 (see in [30] for details).

Suppose $\Gamma \vdash_{\mathcal{HW}} t{:}\tau$ and $[\,] \notin \mathcal{P}(\langle \Gamma, \tau\rangle)$. By Cor. 1 the strategy of contracting only redex T-occurrences terminates in a term t' without such redexes. The term t' is typable by Lem. 2 and then t' turns out to be a $\mathtt{M}$-nf by Lem. 5. Thus, $t \in \mathcal{WN}(\mathtt{M})$.

4 About Strong $\mathtt{M}$-Normalization

In this section we show the third main result of the paper which is a characterization of the set of strongly $\mathtt{M}$-normalizing terms by means of $\mathcal{S}$-typability. The proof is done in several steps. The first key point is the characterization of the set of strongly $\mathtt{M}\backslash\mathtt{w}$-normalizing terms (instead of $\mathtt{M}$-normalizing terms). For that, SR and SE lemmas for the $\mathcal{S}$-type system are needed, and an inductive characterization of the set $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ turns out to be helpful to obtain them. The second key point is the equivalence between strongly $\mathtt{M}$ and $\mathtt{M}\backslash\mathtt{w}$-normalizing terms. While the inclusion $\mathcal{SN}(\mathtt{M}) \subseteq \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ is straightforward, the fact that every $\mathtt{w}$-reduction step can be *postponed* w.r.t. any $\mathtt{M}\backslash\mathtt{w}$-step (Lem. 11) turns out to be crucial to show $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w}) \subseteq \mathcal{SN}(\mathtt{M})$.

We first introduce the **graphical equivalence** $\sim$ on $\mathtt{M}$-terms, given by the contextual, transitive, symmetric and reflexive closure of the following three axioms[4]

$$\begin{array}{lll} t[x/u][y/v] \approx_{\mathtt{CS}} t[y/v][x/u] & \text{if } y \notin \mathtt{fv}(u) \;\&\; x \notin \mathtt{fv}(v) \\ (\lambda y.t)[x/u] \approx_{\sigma_1} \lambda y.t[x/u] & \text{if } y \notin \mathtt{fv}(u) \\ (tv)[x/u] \;\;\approx_{\sigma_2} t[x/u]v & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

[4] Eventhough only σ_2 will be used later to give an inductive definition of $\mathcal{SN}(\mathtt{M})$, the equivalence is presented as a whole.

This equivalence, related to Regnier's σ-equivalence [38] on λ-terms (resp. σ-equivalence on terms with ES [5]), preserves types, a property used to perform some *safe* transformations of terms in order to inductively define the set $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ (*cf.* clause (E)). Note that, for any $t \in \mathcal{T}_{\mathtt{M}}$, we have that the set $\{t' \mid t \to_{\mathtt{M}\backslash\mathtt{w}} t'\}$ is finite. Therefore, for any $t \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$, the **depth of** t can be defined as the maximal length of $\mathtt{M}\backslash\mathtt{w}$-reduction sequences starting at t, denoted by $\eta_{\mathtt{M}\backslash\mathtt{w}}(t)$.

Lemma 6 (Invariance for $\sim$). *Let $t, t' \in \mathcal{T}_{\mathtt{M}}$ s.t. $t \sim t'$. Then, 1) $\eta_{\mathtt{M}\backslash\mathtt{w}}(t) = \eta_{\mathtt{M}\backslash\mathtt{w}}(t')$. 2) If $\Phi \rhd \Gamma \vdash_{\mathcal{S}} t{:}\tau$, then $\Phi' \rhd \Gamma \vdash_{\mathcal{S}} t'{:}\tau$. Moreover, $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi')$.*

In contrast to system $\mathcal{HW}$, whose typing measure $\mathtt{sz}()$ is only decreasing w.r.t. reduction of redex *typed* occurrences, the system $\mathcal{S}$ enjoys a stronger subject reduction property, guaranteeing that *every* reduction decreases the measure $\mathtt{sz}()$ of terms (whose redexes are all typed now).

Lemma 7 (SR II). *Let $\Phi \rhd \Gamma \vdash_{\mathcal{S}} t{:}\tau$. If $t \to_{\mathtt{M}\backslash\mathtt{w}} t'$ then $\Phi' \rhd \Gamma \vdash_{\mathcal{S}} t'{:}\tau$ and $\mathtt{sz}(\Phi) > \mathtt{sz}(\Phi')$.*

Notice that the previous lemma does not hold if the witness derivation in the rules $(\to \mathtt{e}_{\mathcal{S}})$ and $(\mathtt{cut}_{\mathcal{S}})$ in Fig. 2 is only required for the case $I = \emptyset$. For example, given $x[x/y] \to_{\mathtt{c}} y[x/y]$ and their respective typing derivations Φ and Φ', one would have $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_y) + 2 = 2 \cdot \mathtt{sz}(\Phi_y) + 1 = \mathtt{sz}(\Phi')$. One can even have $\mathtt{sz}(\Phi) < \mathtt{sz}(\Phi')$ if y is replaced by an arbitrary bigger term. Notice that an erasing step $v[x/u] \to_{\mathtt{w}} v$ also decreases $\mathtt{sz}(\_)$ but the type assignment for u may change w.r.t. that of $v[x/u]$.

Lemma 8 (SE II). *Let $\Gamma \vdash_{\mathcal{S}} t'{:}\tau$. If $t \to_{\mathtt{M}\backslash\mathtt{w}} t'$ then $\Gamma \vdash_{\mathcal{S}} t{:}\tau$.*

Notice that expansion does not hold for $\to_{\mathtt{w}}$-reduction. For example $x : [\sigma] \vdash_{\mathcal{S}} x{:}\sigma$ and $x[y/\Omega] \to_{\mathtt{w}} x$, but $x : [\sigma] \not\vdash_{\mathcal{S}} x[y/\Omega]{:}\sigma$.

These technical tools are now used to prove that $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ coincides exactly with the set of $\mathcal{S}$-typable terms. To close the picture, *i.e.* to show that also $\mathcal{SN}(\mathtt{M})$ coincides with the set of $\mathcal{S}$-typable terms, we establish an equivalence between $\mathcal{SN}(\mathtt{M})$ and $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$. This is done constructively thanks to the use of an inductive definition for $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$. Indeed, the **inductive set of $\mathtt{M}\backslash\mathtt{w}$-strongly-normalizing terms** is the smallest subset of $\mathcal{T}_{\mathtt{M}}$ that satisfies the following properties:

(V) If $t_1, \ldots, t_n \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$, then $xt_1 \ldots t_n \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.
(L) If $t \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$, then $\lambda x.t \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.
(W) If $t, s \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$ and $|t|_x = 0$, then $t[x/s] \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.

(B) If $u[x/v]t_1,\ldots,t_n \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$, then $(\lambda x.u)vt_1,\ldots,t_n \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.
(C) If $\mathtt{C}[\![u]\!][x/u] \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$, then $\mathtt{C}[\![x]\!][x/u] \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.
(E) If $(tu)[x/s] \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$ and $|u|_x = 0$, then $t[x/s]u \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.

Note the use of the σ_2-axiom in the last clause of the definition. It is not surprising that $\mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$ turns out to be equivalent to $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$, a property which considerably simplifies the proof of Lemma 10.

Lemma 9. $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w}) = \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$

Proof. Given $o \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$, we show $o \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$ by induction on $\langle \eta_{\mathtt{M}\backslash\mathtt{w}}(o), |o|\rangle$. The converse uses induction on the definition of $\mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$.

Lemma 10. *Let $t \in \mathcal{T}_{\mathtt{M}}$. If $t \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ then t is $\mathcal{S}$-typable.*

Proof. Use the equality $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w}) =_{L.\ 9} \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$ to reason by induction on $t \in \mathcal{ISN}(\mathtt{M}\backslash\mathtt{w})$. The proof also uses Lem. 6 and 8 (see [30] for details).

In order to infer $\mathcal{SN}(\mathtt{M}\backslash\mathtt{w}) \subseteq \mathcal{SN}(\mathtt{M})$, the following postponement property is crucial.

Lemma 11 (Postponement). *Let $v \in \mathcal{T}_{\mathtt{M}}$. If $v \rightarrow_{\mathtt{w}}^{+} \rightarrow_{\mathtt{M}\backslash\mathtt{w}} v'$ then $v \rightarrow_{\mathtt{M}\backslash\mathtt{w}} \rightarrow_{\mathtt{w}}^{+} v'$.*

Proof. We first show by cases $v \rightarrow_{\mathtt{w}} \rightarrow_{\mathtt{M}\backslash\mathtt{w}} v'$ implies $v \rightarrow_{\mathtt{M}\backslash\mathtt{w}} \rightarrow_{\mathtt{w}}^{+} v'$. Then, the statement holds by induction on the number of $\mathtt{w}$-steps from v.

Lemma 12 (From $\mathtt{M}\backslash\mathtt{w}$ to $\mathtt{M}$). *Let $t \in \mathcal{T}_{\mathtt{M}}$. If $t \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$, then $t \in \mathcal{SN}(\mathtt{M})$.*

Proof. We show that any reduction sequence $\rho : t \rightarrow_{\mathtt{M}} \ldots$ is finite by induction on the pair $\langle t, n\rangle$, where n is the maximal number such that ρ can be decomposed as $\rho : t \rightarrow_{\mathtt{w}}^{n} t' \rightarrow_{\mathtt{M}\backslash\mathtt{w}} t'' \rightarrow \ldots$ (this is well-defined since $\rightarrow_{\mathtt{w}}$ is trivially terminating). We compare the pair $\langle t, n\rangle$ using $\rightarrow_{\mathtt{M}\backslash\mathtt{w}}$ for the first component (this is well-founded since $t \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ by hyp.) and the standard order on natural numbers for the second one. When the reduction sequence starts with at least one $\mathtt{w}$-step we conclude by Lem. 11. All the other cases are straightforward.

We conclude this section with the third main theorem for $\mathtt{M}$-calculus:

Theorem 4. *Let $t \in \mathcal{T}_{\mathtt{M}}$. Then t is $\mathcal{S}$-typable iff $t \in \mathcal{SN}(\mathtt{M})$.*

Proof. Let $\Phi \triangleright \Gamma \vdash_{\mathcal{S}} t{:}\tau$. Assume $t \notin \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ so that $\exists\infty$ sequence $t = t_0 \rightarrow_{\mathtt{M}\backslash\mathtt{w}} t_1 \rightarrow_{\mathtt{M}\backslash\mathtt{w}} t_2 \rightarrow_{\mathtt{M}\backslash\mathtt{w}} \cdots$. By Lem. 7 $\Phi_i \triangleright \Gamma \vdash t_i{:}\tau$ for every i, and $\exists\infty$ sequence $\mathtt{sz}(\Phi_0) > \mathtt{sz}(\Phi_1) > \mathtt{sz}(\Phi_2) > \ldots$, which leads to a contradiction. Therefore, $t \in \mathcal{SN}(\mathtt{M}\backslash\mathtt{w}) \subseteq_{\text{Lem. } 12} \mathcal{SN}(\mathtt{M})$.

For the converse, $t \in \mathcal{SN}(\mathtt{M}) \subseteq \mathcal{SN}(\mathtt{M}\backslash\mathtt{w})$ because $\rightarrow_{\mathtt{M}\backslash\mathtt{w}} \subseteq \rightarrow_{\mathtt{M}}$. We conclude by Lem. 10.

A corollary of this result is that M-calculus enjoys the **(IE)** Property [26], namely, if $t\{x/u\}$ and u are in $\mathcal{SN}(\texttt{M})$, then $t[x/u]$ is also in $\mathcal{SN}(\texttt{M})$. Indeed, Thm. 4 gives $t\{x/u\}$ and u typable, then Lem. 30 in [30] gives the exact premises to type $t[x/u]$, which belongs to $\mathcal{SN}(\texttt{M})$ by Thm. 4.

5 Conclusion

This paper studies quantitative types for the linear substitution calculus for which we characterized linear-head, head, weak and strongly normalizing sets of terms. In particular, the correspondence between head β-normalization for λ-terms and linear-head M-normalization for terms with ES can now be obtained by means of an *indirect* logical reasoning (*i.e.* the $\mathcal{HW}$-system), in contrast to the operational result given in [4].

The type systems are given by simple formalisms: intersection is represented by multisets, the typing rules are syntax-oriented and no subtyping relation is used. Similar ideas can be applied [30] in the framework of intuitionistic sequent style, giving rise to a reformulation of Herbelin's calculus which is interesting in its own. The $\mathcal{HW}$-system also enjoys the inhabitation property for λ-calculus [12], which is a proper sub-calculus of the linear substitution calculus.

Our strong normalization characterization implies that the number of steps of the longest reduction sequences of terminating M-terms is bounded by the the size of typing derivations. But despite the use of quantitative types, we did not give an exact upper bound, as done for example in [9, 20]. This remains as future work.

Although type inference is undecidable for any system characterizing termination properties, semi-decidable restrictions are expected to hold. Principal typing is a property (*cf.* [21]) which allows to obtain partial typing inference algorithms [40, 39, 31] and exact bounds for termination (*cf.*[10]). Moreover, relevance in the sense of [18] is a key property to obtain principal typings. Therefore semi-decidable typing inference algorithms are also expected to hold for our two non-idempotent type systems.

Neergard *et al.* [37] proved that type inference and execution of typed programs are in different (resp. the same) classes of complexity in the idempotent (resp. non-idempotent) case. However, the system introduced by Carlier *et al.* [14] allows to relax the notion of type linearity. An interesting challenge would be relax the notion of linear types in order to gain expressivity while staying in a different class.

Last but not least, the inhabitation problem for idempotent intersection types in the λ-calculus is known to be undecidable [41], while the problem was recently shown to be decidable in the non-idempotent case [12]. An interesting question concerns the inhabitation problems for our non-idempotent type systems.

6 Acknowledgment

This work was partially funded by the international project **DeCOPA STIC-AmSud 146/2012** (second author partially supported by a PDJ grant from the Brazilian agency CAPES) and the French-Argentinian Laboratory in Computer Science **INFINIS**.

References

1. M. Abadi, L. Cardelli, P-L. Curien, and J-J. Lévy. Explicit substitutions. *JFP*, 1(4):375–416, 1991.
2. B. Accattoli. Evaluating functions as processes. In *TERMGRAPH*, EPTCS 110:41–55, 2013.
3. B. Accattoli, E. Bonelli, D. Kesner, and C. Lombardi. A nonstandard standardization theorem. In *POPL*, ACM, 2014.
4. B. Accattoli and U. Dal Lago. On the invariance of the unitary cost model for head reduction. In *RTA*, LIPIcs 15:22–37, 2012.
5. B. Accattoli and D. Kesner. The structural λ-calculus. In *CSL*, LNCS 6247:381–395, 2010.
6. S. van Bakel. Complete restrictions of the intersection type discipline, *TCS* 102(1):135–163, 1992. (version with a corrected proof at `http://www.doc.ic.ac.uk/~svb/Research/Papers/TCS92corrected.pdf`)
7. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics (revised edition)*. Elsevier Science, Amsterdan, The Netherlands, 1984.
8. H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Bulletin of Symbolic Logic*, 48:931–940, 1983.
9. A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In *FOSSACS*, LNCS 6604:88–107, 2011.
10. A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *LMCS*, 9(4), 2013.
11. G. Boudol, P-L. Curien, and C. Lavatelli. A semantics for lambda calculi with resources. *MSCS*, 9(4):437–482, 1999.
12. A. Bucciarelli, D. Kesner, and S. Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. Submitted.
13. F. Cardone and M. Coppo. Two extension of Curry's type inference system. In *Logic and Computer Science*, *APIC Series* 31, pages 19–75. Academic Press, 1990.
14. S. Carlier, J. Polakow, J. B. Wells, and A. J. Kfoury. System E: Expansion variables for flexible typing with linear and non-linear types and intersection types. In *ESOP'04*, LNCS 2986:294-309. Springer-Verlag, 2004.
15. M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for lambda terms. *Archiv für Mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.

16. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the lambda-calculus *Notre Dame J. of Form. Log.* 21(4):685–693, 1980.
17. M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Mathematical Logic Quarterly*, 27(2-6):45–58, 1981.
18. F. Damiani and P. Giannini. A decidable intersection type system based on relevance. In *TACS'94*, LNCS 789:707–725. Springer, 1994.
19. V. Danos and L. Regnier. Head linear reduction, 2003. `iml.univ-mrs.fr/~regnier/articles/pam.ps.gz`.
20. E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. *ITRS*, EPTCS 121, pages 48-57, 2013.
21. D. de Carvalho. *Sémantiques de la logique linéaire et temps de calcul.* PhD Univ. Aix-Marseille II, 2007.
22. R. Di Cosmo, D. Kesner, and E. Polonovski. Proof nets and explicit substitutions. *MSCS*, 13(3):409–450, 2003.
23. D.Gabbay and R. de Queiroz. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *JSL*, 57(4):1319–1365, 1992.
24. J-Y. Girard. Linear logic. *TCS*, 50:1–102, 1987.
25. J-Y. Girard. Proof-nets: The parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124, 1996.
26. D. Kesner. The theory of calculi with explicit substitutions revisited. In *CSL*, LNCS 4646:238–252, 2007.
27. D. Kesner. A theory of explicit substitutions with safe and full composition. *LMCS*, 5(3), 2009.
28. D. Kesner and S. Lengrand. Resource operators for lambda-calculus. *IandC*, 205(4):419–473, 2007.
29. D. Kesner and S. Ó Conchúir. Milner's lambda calculus with partial substitutions, 2008. Available from `http://www.pps.univ-paris-diderot.fr/~kesner/papers`.
30. D. Kesner and D. Ventura. Quantitative Types for Intuitionistic Calculi. `http://hal.archives-ouvertes.fr/hal-00980868/`.
31. A. J. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables, *TCS*, 311(1-3):1–70, 2004.
32. J.-L. Krivine. A Call-by-Name Lambda-Calculus Machine. HOSC 20(3):199–207, 2007.
33. J.-L. Krivine. Lambda-calculus, types and models, Ellis Horwood, 1993.
34. S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *IandC*, 189:17–42, 2004.
35. G. Mascari and M. Pedicini. Head linear reduction and pure proof net extraction. *TCS*, 135(1):111–137, 1994.
36. R. Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). *ENTCS*, 175(3):65–73, 2007.
37. P. Neergaard and H. Mairson. Types, Potency, and Idempotency: Why Nonlinearity and Amnesia Make a Type System Work. In *ICFP' 04*, SIGPLAN Not. vol. 39(9):138–149, 2004.
38. L. Regnier. Une équivalence e sur les lambda-termes. *TCS*, 2(126):281292, 1994.
39. S. Ronchi Della Rocca. Principal Type scheme and unification for intersection type discipline. *TCS*, 59:1–29, 1988.
40. S. Ronchi Della Rocca and B. Venneri. Principal Type Scheme for an extended type theory. *TCS*, 28:151–169, 1984.
41. P. Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, 64(3):1195–1215, 1999.