A resource aware computational interpretation for Herberlin's syntax

Delia Kesner[1] and Daniel Ventura[2]

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

Abstract. We investigate a new computational interpretation for an intuitionistic focused sequent calculus which is compatible with a resource aware semantics. For that, we associate to Herbelin's syntax a type system based on non-idempotent intersection types, together with a set of reduction rules –inspired from the *substitution at a distance* paradigm– that preserves (and decreases the size of) typing derivations. The non-idempotent approach allows us to use very simple combinatorial arguments, only based on this measure decreasingness, to characterize strongly normalizing terms by means of typability. For the sake of completeness, we also study typability (and the corresponding strong normalization characterization) in the reduction calculus obtained from the former one by projecting the explicit substitutions.

1 Introduction

Intuitionistic logic can be expressed in different formal systems such as natural deduction and sequent calculi. Equivalence between these two formal styles has been widely studied [21, 43, 49, 41, 19], *i.e.* every derivation in one system can be encoded into a derivation of the other one. However, this correspondence is not one-to-one, in particular several *cut-free* proofs in intuitionistic sequent calculus correspond to the same *normal* natural deduction derivation. This gives rise to a restriction of sequent calculi, the so-called *focused* sequent calculi [3], which preserves its structure and expressive power, while establishing a better relationship with natural deduction. Indeed, a one-to-one correspondence is achieved between the cut-free proofs in focused sequent calculi and the normal derivations in natural deduction.

In 1994 Herbelin [25] introduced the $\overline{\lambda}$-calculus, obtained by a computational interpretation of the focused sequent calculus for the minimal intuitionistic logic LJT. In contrast to the usual λ-calculus notation for natural deduction, $\overline{\lambda}$ notation brings head variables to the surface, treats sequences of arguments as lists, and encodes cuts with *explicit substitutions*. Its operational semantics is specified by means of a complete set of

cut-elimination rules. The calculus is permutation-free and can be used to describe proof-search in pure Prolog and some of its extensions [36]. The reduction system of the $\overline{\lambda}$-calculus was then extended [17] with permutation rules, thus showing how to model beta-reduction, *i.e.* thus giving a natural basis for implementation of functional languages.

Unfortunately, Herbelin's calculus is not compatible with a resource aware semantics, mainly because propagation of explicit cuts w.r.t. the *structure* of $\overline{\lambda}$-terms induces useless duplications of empty resources (*cf.* technical discussion in Sec. 3 after Cor. 1). This is substantiated when trying to interpret the λ-calculus by means of proof-nets [23] or non-idempotent intersection types (pioneered by [9, 28, 30]).

The first contribution of the paper is to propose a new computational interpretation for the focused intuitionistic sequent calculus LJT, called E-*calculus*, which is compatible with a resource aware semantics. The calculus keeps Herbelin's syntax but changes the operational semantics of $\overline{\lambda}$ to a resource-controlled interpretation, inspired from the structural lambda-calculus [2], and the linear substitution calculus [37, 1]. The terms of the E-calculus can be seen as λ-terms with explicit cuts of the form $t[x/u]$, where $[x/u]$ is propagated according to the number of free occurrence of x in t (and not w.r.t. the structure of terms). For the sake of completeness, we also study in the second part of the paper the I-*calculus*, a formalism using full –in contrast to partial– substitution, in which normal forms are exactly the same as those of the E-calculus, and whose reduction sequences are obtained by projecting E-reduction sequences into terms without explicit cuts. In other words, E-reduction implements the meta-level operators of the I-calculus by using a resource aware semantics specified by means of explicit reduction rules. Thus, the paper gives a self-contained study of calculi based on sequent calculus, completely independent from their isomorphic natural deduction counter-part.

The second contribution of the paper is to provide type systems based on non-idempotent intersection types for both E and I calculi. Intersection types were introduced to give characterizations of strong β-normalizing terms in the λ-calculus [11, 42, 33]; since then they have been used to characterize termination properties in a broader sense [13], as well as to construct models of the λ-calculus itself [6]. Commonly, intersection types are *idempotent*, *i.e.* $\sigma \wedge \sigma = \sigma$, but we use here *non-idempotent* types [9, 28, 30], suitable to obtain *quantitative* information about reduction sequences. The non-idempotent type systems are used in this paper to characterize strongly normalizing terms, *i.e.* an I-term (resp. E-term) is typable if and only if it is strongly I-normalizing (resp. E-normalizing). Thanks

to the non-idempotent approach, the characterization proofs use simple combinatorial arguments, and do not need any reducibility technique as required in the idempotent case. More precisely, in the case of the E-calculus, the characterization proof is based on the postponement of the erasing steps of the calculus and a combinatorial argument based on a *weighted* subject reduction property. In the case of the I-calculus, which does not admit postponement of erasing steps, a characterization of strongly normalizing I-terms is obtained from the characterization of strongly normalizing E-terms by a projection lemma.

Some related work: In the last years, there has been a growing interest in non-idempotent intersection types. The relation between the size of a non-idempotent intersection typing derivation and the head/weak-normalization execution time of lambda-terms by means of abstract machines was established by D. de Carvalho [16]. Non-idempotence is used to reason about the longest reduction sequence of strongly normalizing terms in both the lambda-calculus [7, 15] and in different lambda-calculi with explicit substitutions [8, 27]. Non-idempotent types also appear in linearization of the lambda-calculus [28], type inference [30, 38], different characterizations of solvability [40] and verification of higher-order programs [39]. While the inhabitation problem for intersection types is known to be undecidable in the idempotent case [46], decidability was recently proved [10] through a sound and complete algorithm in the non-idempotent case. Concerning the use of *idempotent* intersection types for focused intuitionistic sequent calculi, two different papers [18, 22] provide characterizations of strongly normalizing terms by means of typability, but none of them give quantitative information about reduction, as done in this paper. Moreover, in contrast to [22], which is based on explicit control operators for weakening and contraction, we keep the simple, original syntax of Herbelin.

The work presented in this paper originates from a first computational interpretation of LJT appearing in an unpublished technical report [26]. The approach in [26] gives an elegant formulation of the typing rules by introducing witness derivations *everywhere*, so that it is too costly and resource demanding (see the discussion at the end of Sec. 8). The type systems in this paper *only* require *witness* derivations for potentially erasable arguments of functions and substitutions. As a consequence, the upper bound for the longest reduction sequence of a strongly normalizing term obtained in this paper, represented just by the size of a typing derivation, is tighter than the one in [26].

Structure of the paper: The explicit E-calculus is introduced in Sec. 2 and its associated typing system in Sec. 3. The characterization of strongly E-normalizing terms is developed in Sec. 4. Sec. 5 presents the syntax and the operational semantics of the I-calculus, while Sec. 6 presents a non-idempotent typing system for I together with its properties. In Sec. 7 an inductive definition of strongly I-normalizing terms is used to complete the characterization result for the I-calculus. Finally, we conclude in Sec. 8.

2 The E-Calculus

This section introduces the syntax and the operational semantics of the E-calculus. The term language follows from [25], while the reduction rules aim to give a resource aware semantics based on the *substitution at a distance* paradigm [37, 1].

Given a countable infinite set of symbols $x, y, z, \ldots$, three syntactic categories are defined as **E-objects**, **E-terms** and **E-lists**, respectively:

$$o, p ::= t \mid l \qquad t, u ::= xl \mid tl \mid \lambda x.t \mid t[x/t] \qquad l, m ::= \mathtt{nil} \mid t; l$$

The construction $[x/u]$ is said to be an **explicit cut**. Remark that the symbol x alone is not an object of the syntax (term variables in natural deduction style would be encoded by $x\,\mathtt{nil}$), and explicit cuts do not apply to lists, but only to terms, *i.e.* $l[x/u]$ is not in the grammar. We write $tl_1 \ldots l_n$ for $(\ldots(tl_1)\ldots l_n)$ and $x_{\mathtt{nil}}$ for $x\,\mathtt{nil}$. The **size** of the object o is denoted by $|o|$.

The notions of **free** and **bound** variables are defined as usual, in particular,

$$\begin{array}{llll} \mathtt{fv}(xl) & := \{x\} \cup \mathtt{fv}(l) & \mathtt{fv}(t[x/u]) & := (\mathtt{fv}(t) \setminus \{x\}) \cup \mathtt{fv}(u) \\ \mathtt{fv}(tl) & := \mathtt{fv}(t) \cup \mathtt{fv}(l) & \mathtt{fv}(\mathtt{nil}) & := \emptyset \\ \mathtt{fv}(\lambda x.t) & := \mathtt{fv}(t) \setminus \{x\} & \mathtt{fv}(t; l) & := \mathtt{fv}(t) \cup \mathtt{fv}(l) \end{array}$$

The **number of free occurrences of** x **in** o is written $|o|_x$. We work with the standard notions of α**-conversion** (*i.e.* renaming of bound variables for abstractions and substitutions), and Barendregt's convention [5].

We also consider two categories of E-contexts:

$$\begin{array}{ll} \mathtt{L} ::= \Box \mid \mathtt{L}[x/t] & \mathtt{O}, \mathtt{P} ::= \mathtt{C} \mid \mathtt{V} \\ & \mathtt{C}, \mathtt{D} ::= \Box \mid x\mathtt{V} \mid \mathtt{C}l \mid \lambda y.\mathtt{C} \mid \mathtt{C}[y/u] \mid t[y/\mathtt{C}] \mid t\mathtt{V} \\ & \mathtt{V}, \mathtt{U} ::= \mathtt{C}; l \mid t; \mathtt{V} \end{array}$$

When the replacement of the hole of $\mathtt{O}$ by the object o is well defined (*i.e.* gives an object), then we denote it by $\mathtt{O}[o]$. Similarly, $\mathtt{L}[t]$ denotes the

term obtained by replacing the hole of $\mathtt{L}$ by the term t. We write $\mathtt{C}^x$ for a context $\mathtt{C}$ which does not capture the free variable x, i.e. there are no abstractions or explicit substitutions in the context that binds the variable x. For instance, $C = \lambda y.\Box$ can be specified as $\mathtt{C}^x$ while $C = \lambda x.\Box$ cannot. In order to emphasize this particular property we write $\mathtt{C}^x[\![t]\!]$ instead of $\mathtt{C}^x[t]$, and we may omit x when it is clear from the context.

The **reduction relation** $\to_{\mathtt{E}}$ is defined as the closure by contexts $\mathtt{O}$ of the following rewriting rules:

$$\begin{array}{lll}
\mathtt{L}[\lambda x.t]\mathtt{nil} & \mapsto_{\mathtt{dB}_{\mathtt{nil}}} & \mathtt{L}[\lambda x.t] \\
\mathtt{L}[\lambda x.t](u;l) & \mapsto_{\mathtt{dB}_{\mathtt{cons}}} & \mathtt{L}[t[x/u]l] \\
\mathtt{C}^x[\![x\,l]\!][x/u] & \mapsto_{\mathtt{c}} & \mathtt{C}^x[\![u\,l]\!][x/u] \quad \text{if } |\mathtt{C}^x[\![x\,l]\!]|_x > 1 \\
\mathtt{C}^x[\![x\,l]\!][x/u] & \mapsto_{\mathtt{d}} & \mathtt{C}^x[\![u\,l]\!] \quad \text{if } |\mathtt{C}^x[\![x\,l]\!]|_x = 1 \\
t[x/u] & \mapsto_{\mathtt{w}} & t \quad \text{if } |t|_x = 0 \\
\mathtt{L}[xl]m & \mapsto_{@_{\mathtt{var}}} & \mathtt{L}[x(l@m)] \\
\mathtt{L}[tl]m & \mapsto_{@_{\mathtt{app}}} & \mathtt{L}[t(l@m)]
\end{array}$$

where the operation $\_@\_$ is defined by the following equations:

$$\mathtt{nil}@l := l \qquad\qquad (u;l)@m := u;(l@m)$$

An example of reduction sequence is

$$\begin{array}{l}
(\lambda x.x(x_{\mathtt{nil}};\mathtt{nil}))(u;\mathtt{nil}) \to_{\mathtt{dB}_{\mathtt{cons}}} x(x_{\mathtt{nil}};\mathtt{nil})[x/u]\mathtt{nil} \to_{@_{\mathtt{var}}} \\
x(x_{\mathtt{nil}};\mathtt{nil})[x/u] \to_{\mathtt{c}} x(u\,\mathtt{nil};\mathtt{nil})[x/u] \to_{\mathtt{d}} u(u\,\mathtt{nil};\mathtt{nil})
\end{array}$$

There are many differences with the reduction rules in [25]. First of all, the use of the meta-operation @ for concatenating lists in the rules $\mapsto_{@_{\mathtt{var}}}$ and $\mapsto_{@_{\mathtt{app}}}$ replaces the explicit concatenation rules in [25]. This is particularly convenient since we only reduce objects that are terms (even if these terms occur inside lists), so that the proofs are simpler/shorter because there are less rules and only of one kind. A major difference with [25] is the use of rules *at a distance*, specified by means of (term and list) contexts, where the propagation of substitutions is not performed by structural induction on terms, since they are consumed according to the multiplicity of their corresponding variables. As a consequence, the behaviour of substitution is specified by a resource aware semantics, thus preventing the useless duplication of empty resources, which happens in [25] when using reduction steps of the form $(tl)[x/u] \to t[x/u]l[x/u]$, where $|tl|_x = 0$. This is particularly unsuitable when considering non-idempotent types (*cf.* the discussion at the end of Sec. 3). In contrast to other calculi at a distance which only contains $\mathtt{w}$ and $\mathtt{c}$-rules, such as for example the linear substitution calculus [37, 1], we also consider here a dereliction rule $\mathtt{d}$. This

is appropriate to obtain a *weighted* subject reduction property relative to our typing system (*cf.* Sec. 3), which would fail for the alternative rewriting rule $\mathtt{C}[\![x\,l]\!][x/u] \mapsto_{\mathtt{c}} \mathtt{C}[\![u\,l]\!][x/u]$ when $|\mathtt{C}[\![x\,l]\!]|_x = 1$.

The reduction relation $\to_{\mathtt{E}}$ can also be refined. We write $\to_X$ for the closure by contexts $\mathtt{O}$ of the rewriting rule $\mapsto_X$ for every X. We define $\mathtt{B@} := \{\mathtt{dB_{nil}} \cup \mathtt{dB_{cons}} \cup @_{\mathtt{var}} \cup @_{\mathtt{app}}\}$ and $\to_{\mathtt{B@}} := \bigcup_{X \in \mathtt{B@}} \to_X$. The **non-erasing** reduction relation $\to_{\mathtt{E\setminus w}}$ is given by $\to_{\mathtt{B@} \cup \{\mathtt{d},\mathtt{c}\}}$, *i.e.* $\to_{\mathtt{E\setminus w}} = \to_{\mathtt{E}} \setminus \to_{\mathtt{w}}$, and plays a key role in the characterization of strongly $\mathtt{E}$-normalizing terms (*cf.* Sec. 4).

Let $\mathcal{R}$ be any reduction system. We denote by $\to_{\mathcal{R}}^*$ (resp. $\to_{\mathcal{R}}^+$) the **reflexive-transitive** (resp. **transitive**) closure of a given reduction relation $\to_{\mathcal{R}}$. The reduction relation $\mathcal{R}$ is **confluent** if and only if for all objects o_1, o_2, o_3 such that $o_1 \to_{\mathcal{R}}^* o_2$ and $o_1 \to_{\mathcal{R}}^* o_3$, there is o_4 being able to close the diagram, *i.e.* $o_2 \to_{\mathcal{R}}^* o_4$ and $o_3 \to_{\mathcal{R}}^* o_4$. An object o is **strongly $\mathcal{R}$-normalizing**, written $o \in \mathcal{SN}(\mathcal{R})$, if there is no infinite $\mathcal{R}$-reduction sequence starting at o, and o is **$\mathcal{R}$-finitely branching** if the set $\{o' \mid o \to_{\mathcal{R}} o'\}$ is finite. If an object o is $\mathcal{R}$-strongly normalizing and $\mathcal{R}$-finitely branching then the **depth of** o, written $\eta_{\mathcal{R}}(o)$, is the maximal length of $\mathcal{R}$-reduction sequences starting at o.

3 A Non-Idempotent Typing System for E-Terms

This section introduces the typing system $\mathcal{Q}_{\mathtt{E}}$ for the $\mathtt{E}$-calculus. Given a countable infinite set of *base types* $\alpha, \beta, \gamma, \ldots$ we consider **types** and **multiset types** defined as follows:

(**types**)	τ, σ, ρ	$::= \alpha \mid \mathcal{M} \to \tau$	
(**multiset types**)	$\mathcal{M}$	$::= [\tau_i]_{i \in I}$	where I is a finite set

Our types are *strict* [12, 48], *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 [16], so that $[\,]$ denotes the empty multiset, and $[\sigma, \sigma, \tau]$ must be understood as $\sigma \wedge \sigma \wedge \tau$, where the symbol $\wedge$ enjoys commutativity and associativity but not idempotence, *i.e.* $\sigma \wedge \sigma$ is not equal to σ.

Type assignments, 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 $\mathtt{dom}(\Gamma) := \{x \mid \Gamma(x) \neq [\,]\}$. The **intersection of type assignments**, written $\Gamma + \Delta$, is defined by $(\Gamma + \Delta)(x) := \Gamma(x) + \Delta(x)$, where the symbol $+$ denotes multiset union. Hence, $\mathtt{dom}(\Gamma + \Delta) = \mathtt{dom}(\Gamma) \cup \mathtt{dom}(\Delta)$. When $\mathtt{dom}(\Gamma)$ and $\mathtt{dom}(\Delta)$ are disjoint we write $\Gamma; \Delta$ instead of $\Gamma + \Delta$. We write $\Gamma \setminus\!\setminus x$ for the assignment $(\Gamma \setminus\!\setminus x)(x) = [\,]$ and $(\Gamma \setminus\!\setminus x)(y) = \Gamma(y)$ if $y \neq x$.

$$\frac{}{\emptyset \mid \tau \vdash \mathtt{nil}{:}\tau}\,(\mathtt{ax}) \qquad \frac{\Gamma \mid \_ \vdash t{:}\tau}{\Gamma \setminus\!\setminus x \mid \_ \vdash \lambda x.t{:}\Gamma(x) \to \tau}\,(\to \mathtt{r}) \qquad \frac{\Gamma \mid \sigma \vdash l{:}\tau}{\Gamma + \{x{:}[\sigma]\} \mid \_ \vdash xl{:}\tau}\,(\mathtt{hlist})$$

$$\frac{\Gamma \mid \_ \vdash t{:}\sigma \qquad \Delta \mid \sigma \vdash l{:}\tau}{\Gamma + \Delta \mid \_ \vdash tl{:}\tau}\,(\mathtt{app}) \qquad \frac{\Gamma \mid \_ \vdash t{:}\rho \qquad \Delta \mid \tau \vdash l{:}\sigma}{\Delta + \Gamma \mid [\,] \to \tau \vdash t; l{:}\sigma}\,(\to \mathtt{l}_{\notin})$$

$$\frac{(\Gamma_j \mid \_ \vdash t{:}\tau_j)_{j\in J} \quad J \neq \emptyset \qquad \Delta \mid \tau \vdash l{:}\sigma}{\Delta +_{j\in J} \Gamma_j \mid [\tau_j]_{j\in J} \to \tau \vdash t; l{:}\sigma}\,(\to \mathtt{l}_{\in})$$

$$\frac{\Delta \mid \_ \vdash u{:}\sigma \qquad \Gamma \mid \_ \vdash t{:}\tau \qquad x \notin \mathtt{dom}(\Gamma)}{\Gamma + \Delta \mid \_ \vdash t[x/u]{:}\tau}\,(\mathtt{es}_{\notin})$$

$$\frac{(\Delta_j \mid \_ \vdash u{:}\sigma_j)_{j\in J} \qquad J \neq \emptyset \qquad x{:}[\sigma_j]_{j\in J}; \Gamma \mid \_ \vdash t{:}\tau}{\Gamma +_{j\in J} \Delta_j \mid \_ \vdash t[x/u]{:}\tau}\,(\mathtt{es}_{\in})$$

Fig. 1. The Type System $\mathcal{Q}_{\mathtt{E}}$ for the E-Calculus

The symbol $\_$ is called the **empty stoup**. A **stoup** Σ is either a type σ or the empty stoup. **Type environments** are pairs of the form $\Gamma \mid \Sigma$, where Γ is a type assignment and Σ is a stoup. **Type judgments** are triples of the form $\Gamma \mid \Sigma \vdash o{:}\tau$, where o is an object, $\Gamma \mid \Sigma$ a type environment and τ a type. The $\mathcal{Q}_{\mathtt{E}}$ type system for the E-calculus is given in Fig. 1 ; it derives type judgments of the form $\Gamma \mid \_ \vdash t{:}\tau$ and $\Gamma \mid \sigma \vdash l{:}\tau$, where t is a term and l is a list. We write $\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o{:}\tau$ or $\Phi \triangleright_{\mathcal{Q}_{\mathtt{E}}} \Gamma \mid \Sigma \vdash o{:}\tau$ to denote derivability in system $\mathcal{Q}_{\mathtt{E}}$. The **hlist-size** of the type derivation Φ is a positive natural number written $\mathtt{sz2}(\Phi)$ which denotes the size of Φ where every node `hlist` counts 2 [3]. Example 1 further justifies the use of this `hlist`-size function.

Notice that the system $\mathcal{Q}_{\mathtt{E}}$ is syntax oriented, *i.e.* for each type judgment of the form $\Gamma \mid \Sigma \vdash o{:}\tau$ there is a *unique* typing rule whose conclusion matches the type judgment. Indeed, there are two different rules to type a list $t; l$, but the type in the stoup Σ discriminates between them. There is a similar distinction for terms of the form $t[x/u]$ depending on whether x is in the free variables of t or not. A consequence of this property is that statements usually proved in generation lemmas (*cf.* [6, 35]) are straightforward in system $\mathcal{Q}_{\mathtt{E}}$.

[3] The node `hlist` counts 2 since it corresponds, in the standard sequent calculus, to an application of an axiom rule followed by a contraction.

The (app) typing rule is the *head-cut* rule in the underlying logical system; similarly, ($\mathtt{es}_{\notin}$) and ($\mathtt{es}_{\in}$) give an interpretation of the so-called *mid-cut* [25]. The type derivation for t (resp. for u) in rule ($\to \mathtt{l}_{\notin}$) (resp. ($\mathtt{es}_{\notin}$)) is called a **witness** derivation, and turns out to be essential to guarantee strong-normalization of the whole typed term $t;l$ (resp. $t[x/u]$). Indeed, if witness derivations are not required, then non-terminating terms like $t(\Omega;l)$ or $t[x/\Omega]$ would be typable in the system, for $\Omega = (\lambda x.xx_{\mathtt{nil}})\lambda x.xx_{\mathtt{nil}}$. The rules ($\to \mathtt{l}_{\notin}$) and ($\to \mathtt{l}_{\in}$) (resp. ($\mathtt{es}_{\notin}$) and ($\mathtt{es}_{\in}$)) can be specified by means of a unique typing rule ($\to \mathtt{l}$) (resp. ($\mathtt{es}$)), usually used in the proofs in order to save some space. They have the form:

$$\frac{(\Gamma_j \mid \_ \vdash t{:}\tau_j)_{j\in J} \qquad \Delta \mid \tau \vdash l{:}\sigma}{\Delta +_{j\in J} \Gamma_j \mid [\tau_i]_{i\in I} \to \tau \vdash t;l{:}\sigma} \ (\to \mathtt{l})$$

$$\frac{(\Delta_j \mid \_ \vdash u{:}\sigma_j)_{j\in J} \qquad x{:}[\sigma_i]_{i\in I};\Gamma \mid \_ \vdash t{:}\tau}{\Gamma +_{j\in J} \Delta_j \mid \_ \vdash t[x/u]{:}\tau} \ (\mathtt{es})$$

where $(I = \emptyset \Rightarrow |J| = 1)$ and $(I \neq \emptyset \Rightarrow I = J)$.

The system $\mathcal{Q}_{\mathtt{E}}$ is *relevant* [14], *i.e.* typing environments only contain the consumed premises. Moreover, in contrast to [6, 8], no subtyping relation is needed for abstractions and/or applications.

Lemma 1. *If* $\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o{:}\tau$, *then* $\mathtt{dom}(\Gamma) = \mathtt{fv}(o)$.

We now introduce a technical tool which will be used in Sec. 4 in order to give a characterization of strongly E-normalizing terms. Indeed, we do not want to distinguish terms having explicit cuts at different *head positions*, mainly because they do have exactly the same maximal reduction lengths. More precisely, the **head graphical equivalence** $\sim$ on E-terms, inspired from the σ-equivalence on λ-terms [44] and the σ-equivalence on λ-terms with explicit substitutions [2], is given by the contextual, transitive, symmetric and reflexive closure of the following axiom

$$(tl)[x/u] \approx t[x/u]l, \text{ where } |l|_x = 0$$

Notice that $(xl)[x/u]$ cannot be $\sim$-converted into $x[x/u]l$ when $x \notin \mathtt{fv}(l)$, since x alone is not a term of the calculus.

The main properties of system $\mathcal{Q}_{\mathtt{E}}$ for E-terms follow.

Lemma 2 (Invariance for $\sim$). *Let* o, o' *be* E*-objects s.t.* $o \sim o'$. *Then,*

1. $o \to_{\mathtt{E}\setminus\mathtt{w}} o_0$ *iff* $o' \to_{\mathtt{E}\setminus\mathtt{w}} o'_0$, *where* $o_0 \sim o'_0$. *In particular,* $\eta_{\mathtt{E}\setminus\mathtt{w}}(o) = \eta_{\mathtt{E}\setminus\mathtt{w}}(o')$.

2. $\Phi \rhd \Gamma \vdash_{\mathcal{Q}_{\mathtt{E}}} o{:}\tau$ *iff* $\Phi' \rhd \Gamma \vdash_{\mathcal{Q}_{\mathtt{E}}} o'{:}\tau$*. Moreover,* $\mathtt{sz2}(\Phi) = \mathtt{sz2}(\Phi')$*.*

Lemma 3 (Weighted Subject Reduction). *Let* $\Phi \rhd \Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o{:}\tau$*. If* $o \to_{\mathtt{E}\setminus\mathtt{w}} o'$*, then* $\Phi' \rhd \Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o'{:}\tau$ *and* $\mathtt{sz2}(\Phi) > \mathtt{sz2}(\Phi')$*.*

The $\mathtt{sz2}$ function plays a central role in obtaining a strictly decreasing measure in the subject reduction lemma above. More precisely, if we consider the standard measure on typing derivations, written $\mathtt{sz}$, which counts 1 for every node of the derivation tree, then the weighted subject reduction property does not hold. Here is an example.

Example 1. Let t be the term $x(x_{\mathtt{nil}}; \mathtt{nil})$, and consider the following type derivation Φ_t such that $\mathtt{sz}(\Phi_t) = 5$.

$$\Phi_t := \dfrac{\dfrac{\dfrac{\overline{\emptyset \mid \sigma \vdash \mathtt{nil}{:}\sigma}}{x{:}[\sigma] \mid \_ \vdash x_{\mathtt{nil}}{:}\sigma} \qquad \overline{\emptyset \mid \tau \vdash \mathtt{nil}{:}\tau}}{x{:}[\sigma] \mid [\sigma]\to\tau \vdash x_{\mathtt{nil}}; \mathtt{nil}{:}\tau}}{x{:}[\sigma, [\sigma]\to\tau] \mid \_ \vdash x(x_{\mathtt{nil}}; \mathtt{nil}){:}\tau}$$

Let u be a term s.t. $\Phi_u^1 \rhd \Delta_1 \mid \_ \vdash u{:}\sigma$ and $\Phi_u^2 \rhd \Delta_2 \mid \_ \vdash u{:}[\sigma]\to\tau$. Therefore,

$$\Phi := \dfrac{\Phi_u^1 \rhd \Delta_1 \mid \_ \vdash u{:}\sigma \qquad \Phi_u^2 \rhd \Delta_2 \mid \_ \vdash u{:}[\sigma]\to\tau \qquad \Phi_t \rhd x{:}[\sigma, [\sigma]\to\tau] \mid \_ \vdash t{:}\tau}{\Delta_1 + \Delta_2 \mid \_ \vdash t[x/u]{:}\tau}$$

where $\mathtt{sz}(\Phi) = 6 + _{i=1,2}\, \mathtt{sz}(\Phi_u^i)$.

Given the reduction step $t[x/u] \to_{\mathtt{c}} x(u\,\mathtt{nil}; \mathtt{nil})[x/u] = t'$, there is a derivation Φ' typing t' such that $\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi)$. Indeed,

$$\Phi' := \dfrac{\Phi_u^2 \rhd \Delta_2 \mid \_ \vdash u{:}[\sigma]\to\tau \qquad \dfrac{\dfrac{\Phi_u^1 \rhd \Delta_1 \mid \_ \vdash u{:}\sigma \qquad \overline{\emptyset \mid \sigma \vdash \mathtt{nil}{:}\sigma}}{\Delta_1 \mid \_ \vdash u\,\mathtt{nil}{:}\sigma} \qquad \overline{\emptyset \mid \tau \vdash \mathtt{nil}{:}\tau}}{\Delta_1 \mid [\sigma]\to\tau \vdash u\,\mathtt{nil}; \mathtt{nil}{:}\tau}}{x{:}[[\sigma]\to\tau] + \Delta_1 \mid \_ \vdash x(u\,\mathtt{nil}; \mathtt{nil}){:}\tau}}{\Delta_1 + \Delta_2 \mid \_ \vdash t'{:}\tau}$$

Corollary 1. *If* o *is* $\mathcal{Q}_{\mathtt{E}}$*-typable then* $o \in \mathcal{SN}(\mathtt{E}\setminus\mathtt{w})$*.*

As we mentioned in the introduction, the $\overline{\lambda}$-calculus [25] is not compatible with a resource aware semantics, as illustrated by the following example. Consider a $\overline{\lambda}$-reduction of the form $o = (tl)[x/u] \to t[x/u]l[x/u] = o'$, and

suppose $|tl|_x = 0$. Let Φ be a typing derivation for the object o, thus having the following form:

$$\Phi := \dfrac{\dfrac{\Phi_t \triangleright \Gamma_t \mid \_ \vdash t{:}\sigma_t \qquad \Phi_l \triangleright \Gamma_l \mid \sigma_t \vdash l{:}\tau}{\Gamma_t + \Gamma_l \mid \_ \vdash tl{:}\tau} \qquad \Phi_u \triangleright \Delta \mid \_ \vdash u{:}\sigma}{(\Gamma_t + \Gamma_l) + \Delta \mid \_ \vdash (tl)[x/u]{:}\tau}$$

The typing derivation for the object o', let say Φ', must use *twice* the typing tree Φ_u, and thus $\mathtt{sz2}(\Phi) > \mathtt{sz2}(\Phi')$ cannot hold. In other words, propagation of substitution w.r.t. the structure of terms induces useless duplications of empty resources, turning out to be inappropriate in the framework of a resource aware semantics.

The last key property relates typing with expansion:

Lemma 4 (Subject Expansion). *Let $\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o'{:}\tau$. If $o \to_{\mathtt{E}\setminus\mathtt{w}} o'$, then $\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{E}}} o{:}\tau$.*

4 Characterizing Strongly E-Normalizing Terms

This section is devoted to the characterization of E-strong normalization. We use the technical tools developed in Sec. 3 to characterize strongly normalizing E-terms by means of $\mathcal{Q}_{\mathtt{E}}$-typability, namely, the subject reduction and expansion properties.

We start by giving an alternative definition of $\mathcal{SN}(\mathtt{E}\setminus\mathtt{w})$, where $\to_{\mathtt{E}\setminus\mathtt{w}}$ is defined as $\to_{\mathtt{E}} \setminus \to_{\mathtt{w}}$. Indeed, the **inductive set of $\mathtt{E}\setminus\mathtt{w}$-strongly-normalizing objects**, written $\mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, is the smallest subset of objects satisfying the following properties:

- (EL) $\mathtt{nil} \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- (NEL) If $t, l \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $t; l \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$,
- (L) If $t \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $\lambda x.t \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- (HL) If $l \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $xl \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- (W) If $t, s \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$ and $|t|_x = 0$, then $t[x/s] \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- $(\mathtt{dB}_{\mathtt{nil}})$ If $(\lambda x.t)l_1 \dots l_n$ $(n \geq 0) \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $(\lambda x.t)\mathtt{nil}l_1 \dots l_n \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- $(\mathtt{dB}_{\mathtt{cons}})$ If $t[x/u]ml_1 \dots l_n$ $(n \geq 0) \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $(\lambda x.t)(u; m)l_1 \dots l_n \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- $(@_{\mathtt{var}})$ If $x(m_1@m_2)l_1 \dots l_n$ $(n \geq 0) \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $(xm_1)m_2l_1 \dots l_n \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- $(@_{\mathtt{app}})$ If $t(m_1@m_2)l_1 \dots l_n$ $(n \geq 0) \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$, then $(tm_1)m_2l_1 \dots l_n \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.
- (C) If $\mathtt{C}[\![u\, l]\!][x/u] \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$ and $|\mathtt{C}[\![x\, l]\!]|_x > 1$, then $\mathtt{C}[\![x\, l]\!][x/u] \in \mathcal{ISN}(\mathtt{E}\setminus\mathtt{w})$.

(D) If $\mathtt{C}[\![u\,l]\!] \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ and $|\mathtt{C}[\![x\,l]\!]|_x = 1$, then $\mathtt{C}[\![x\,l]\!][x/u] \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$.
(E) If $(tl)[x/s] \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ and $|l|_x = 0$, then $t[x/s]l \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$.

Remark in particular that case (E) guarantees the closure of $\mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ for the head graphical equivalence.

It is not difficult to show that the sets $\mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$ and $\mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ coincide (the proof follows the same scheme used for example in [26]) so that we can show the following result:

Lemma 5. *Let o be an $\mathtt{E}$-object. If $o \in \mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$ then o is $\mathcal{Q}_{\mathtt{E}}$-typable.*

Proof. By induction on the structure of $o \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w}) = \mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$.

- If $o = \mathtt{nil}$, $o = t; l$, $o = \lambda x.t$, $o = xl$, or $o = u[x/v]$ with $|u|_x = 0$, then the proof is straightforward by using the *i.h.*
- If $o \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ comes from one of the rules $(\mathtt{dB}_{\mathtt{nil}})$, $(\mathtt{dB}_{\mathtt{cons}})$, (C), (D), $(@_{\mathtt{var}})$ or $(@_{\mathtt{app}})$, then the property holds by the *i.h.* and the Subject Expansion Lem. 4.
- If $t[x/s]l \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$ comes from the rule (E), then $(tl)[x/s] \in \mathcal{ISN}(\mathtt{E}\backslash\mathtt{w})$, so that $(tl)[x/s]$ is $\mathcal{Q}_{\mathtt{E}}$-typable by the *i.h.* and the property holds by Lem. 2.2.

Strong $\mathtt{E}$-normalization can be now obtained from strong $\mathtt{E}\backslash\mathtt{w}$-normalization as follows:

Lemma 6 (From $\mathtt{E}\backslash\mathtt{w}$ to $\mathtt{E}$). *Let o be an $\mathtt{E}$-object. If $o \in \mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$, then $o \in \mathcal{SN}(\mathtt{E})$.*

Proof. One first shows a postponement property for $\mathtt{w}$-reduction steps given by: if $o \to_{\mathtt{w}}^{+} \to_{\mathtt{E}\backslash\mathtt{w}} o'$, then $o \to_{\mathtt{E}\backslash\mathtt{w}} \to_{\mathtt{w}}^{+} o'$. Then the property is proved by contradiction using the postponement property.

We can now conclude with the main result of this section.

Theorem 1. *Let o be an $\mathtt{E}$-object. Then o is $\mathcal{Q}_{\mathtt{E}}$-typable iff $o \in \mathcal{SN}(\mathtt{E})$.*

Proof. Let o be $\mathcal{Q}_{\mathtt{E}}$-typable. Then $o \in \mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$ by Cor. 1 and $o \in \mathcal{SN}(\mathtt{E})$ by Lem. 6. For the converse, $o \in \mathcal{SN}(\mathtt{E}) \subseteq \mathcal{SN}(\mathtt{E}\backslash\mathtt{w})$ because $\to_{\mathtt{E}\backslash\mathtt{w}} \subseteq \to_{\mathtt{E}}$. We conclude by Lem. 5.

5 The I-Calculus

We now introduce the syntax and the operational semantics of the I-calculus, slightly differently defined in [20]. The I-calculus can be obtained from E by an appropriate projection function (*cf.* Lem. 8).

Given a countable infinite set of symbols $x, y, z, \ldots$, three syntactic categories as **I-objects**, **I-terms** and **I-lists** are respectively defined by the following grammars:

$$o ::= t \mid l \qquad t, u, v ::= \lambda x.t \mid xl \mid (\lambda x.t)l \qquad l, m ::= \mathtt{nil} \mid t; l$$

Remark that general terms of the form tl are not I-terms.

As before, we work with Barendregt's convention and the standard notion of α-conversion. The notions of **I-term** and **I-list contexts** are defined as expected according to the grammars above. The **reduction relation** $\to_{\mathtt{I}}$ is given by the context closure of the following rules:

$$(\lambda x.t)\mathtt{nil} \mapsto_{\beta_{\mathtt{nil}}} \lambda x.t \qquad (\lambda x.t)(u; l) \mapsto_{\beta_{\mathtt{cons}}} t\{x/u\} \circ l$$

where the operations $\_ \circ \_$ and $\{\_/\_\}$ are defined as follows:

$$\begin{array}{lll}
(xl) \circ m & := x\ (l@m) & \mathtt{nil}\{x/v\} := \mathtt{nil} \\
((\lambda y.t)l) \circ m & := (\lambda y.t)(l@m) & (u; l)\{x/v\} := u\{x/v\}; l\{x/v\} \\
(\lambda x.t) \circ m & := (\lambda x.t)m & (y\ l)\{x/v\} := yl\{x/v\} \\
 & & (xl)\{x/v\} := v\ \circ l\{x/v\} \\
 & & ((\lambda y.t)l)\{x/v\} := (\lambda y.t\{x/v\})l\{x/v\} \\
 & & (\lambda y.t)\{x/v\} := \lambda y.t\{x/v\}
\end{array}$$

The substitution operator $\{\_/\_\}$ is defined on α-equivalence classes of terms in order to avoid the capture of free variables. Notice that substitution distributes with respect to @ and $\circ$, *i.e.* one can show that $(t@l)\{x/u\} = t\{x/u\}@l\{x/u\}$ and $(t \circ l)\{x/u\} = t\{x/u\} \circ l\{x/u\}$. As expected, the I-calculus enjoys confluence.

An **erasing step** is the closure by contexts of the reduction rule $(\lambda x.t)(u; l) \mapsto t\{x/u\} \circ l$, where $x \notin \mathtt{fv}(t)$, *i.e.* an erasing step discards the argument u since $x \notin \mathtt{fv}(o)$ implies $o\{x/u\} = o$. Notice that erasing steps cannot be postponed, so that we cannot apply to the I-calculus the same (simple) proof technique used in Sec. 4 to characterize E-strong normalization. An example of non-erasing step is

$$(\lambda y.y_{\mathtt{nil}})(x_{\mathtt{nil}}; x_{\mathtt{nil}}; \mathtt{nil}) \to x(x_{\mathtt{nil}}; \mathtt{nil})$$

while $(\lambda y.z_{\mathtt{nil}})(x_{\mathtt{nil}}; x_{\mathtt{nil}}; \mathtt{nil}) \to z(x_{\mathtt{nil}}; \mathtt{nil})$ is an erasing step. Non-erasing steps play a key role in Lem. 11.

Typing Rules $\{(\mathtt{ax}), (\to \mathtt{r}), (\mathtt{hlist}), (\to \mathtt{l}_{\notin}), (\to \mathtt{l}_{\in})\}$ plus

$$\frac{\Gamma \mid \_ \vdash \lambda x.t{:}\sigma \qquad \Delta \mid \sigma \vdash l{:}\tau}{\Gamma + \Delta \mid \_ \vdash (\lambda x.t)l{:}\tau}\ (\mathtt{app})$$

Fig. 2. The Type System $\mathcal{Q}_{\mathtt{I}}$ for the I-Calculus

The I-calculus can be simulated in the E-calculus in terms of more atomic steps. Indeed,

Lemma 7. *Let o be an* I*-term. If $o \to_{\mathtt{I}} o'$, then $o \to_{\mathtt{E}}^{+} o'$.*

Proof. One first shows that for all I-objects t, u, l, $t[x/u] \to_{\mathtt{E}}^{*} t\{x/u\}$ and $tl \to_{\mathtt{E}}^{*} t \circ l$. The proof of the statement of the lemma then proceeds by induction on I-reduction. The interesting case is when $o = (\lambda x.t)(u; l) \to_{\mathtt{I}} t\{x/u\} \circ l = o'$, for which we conclude by $o = (\lambda x.t)(u; l) \to_{\mathtt{E}} t[x/u]l \to_{\mathtt{E}}^{*} t\{x/u\}l \to_{\mathtt{E}}^{*} t\{x/u\} \circ l = o'$ using the properties mentioned above.

Reciprocally, the E-calculus can be projected into the I-calculus. For that, we first remark that the system $\mathtt{sub} = \{\mathtt{w}, \mathtt{d}, \mathtt{c}, @_{\mathtt{var}}, @_{\mathtt{app}}\}$ is locally confluent and terminating. Hence sub-normal forms of objects are unique; we thus write $\mathtt{sub}(o)$ for the sub-normal forms of the object o.

Lemma 8 (Projection). *Let o be an* E*-term. If $o \to_{\mathtt{E}} o'$, then $\mathtt{sub}(o) \to_{\mathtt{I}}^{*} \mathtt{sub}(o')$.*

Proof. One first shows that for all E-terms t, u, one has $\mathtt{sub}(t[x/u]) = \mathtt{sub}(t)\{x/\mathtt{sub}(u)\}$. The proof of the statement of the lemma is then by induction on E-reduction using the previous remark.

Using confluence of I, together with Lemmas 7 and 8 we obtain the following property.

Corollary 2. *The reduction relation $\to_{\mathtt{E}}$ is confluent.*

6 A Non-Idempotent Typing System for I-Terms

In this section we restrict to I-objects the system $\mathcal{Q}_{\mathtt{E}}$ introduced in Sec. 3. The resulting system $\mathcal{Q}_{\mathtt{I}}$ is given in Fig. 2. As before, relevance holds for I-objects.

Lemma 9. *If $\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{I}}} o{:}\tau$, then $\mathtt{dom}(\Gamma) = \mathtt{fv}(o)$.*

In order to characterize the set $\mathcal{SN}(\mathtt{I})$ of strongly $\mathtt{I}$-normalizing term by means of $\mathcal{Q}_{\mathtt{I}}$-typability, we first need to show that every $\mathcal{Q}_{\mathtt{I}}$-typable term is strongly $\mathtt{I}$-normalizing. This is obtained as follows:

Lemma 10. *Let o be an $\mathtt{I}$-term. Then, $o \in \mathcal{SN}(\mathtt{E})$ implies $o \in \mathcal{SN}(\mathtt{I})$.*

Proof. A direct consequence of Lem. 7.

Corollary 3. *If $\Phi \rhd_{\mathcal{Q}_{\mathtt{I}}} \Gamma \mid \Sigma \vdash o{:}\tau$, then $o \in \mathcal{SN}(\mathtt{I})$.*

Proof. If o is $\mathcal{Q}_{\mathtt{I}}$-typable, then o is also trivially $\mathcal{Q}_{\mathtt{E}}$-typable. Thm. 1 gives $o \in \mathcal{SN}(\mathtt{E})$ and Lem. 10 gives $o \in \mathcal{SN}(\mathtt{I})$.

The converse property, *i.e.* the fact that every strongly $\mathtt{I}$-normalizing term is $\mathcal{Q}_{\mathtt{I}}$-typable, will be proved in Sec. 7. For that, the following key property will be used.

Lemma 11 (Subject Expansion for Non-Erasing Reductions). *If $\Phi \rhd_{\mathcal{Q}_{\mathtt{I}}} \Gamma \mid \Sigma \vdash o'{:}\tau$ and $o \to_{\mathtt{I}} o'$ is a non-erasing step, then there exists Φ' such that $\Phi' \rhd_{\mathcal{Q}_{\mathtt{I}}} \Gamma \mid \Sigma \vdash o{:}\tau$.*

Proof. By induction on the non-erasing reduction relation $\to_{\mathtt{I}}$.

7 Characterizing Strongly I-Normalizing Terms

This section completes the characterization result for the $\mathtt{I}$-calculus, namely, we show that every strongly $\mathtt{I}$-normalizing term is $\mathcal{Q}_{\mathtt{I}}$-typable, so that, together with Cor. 3, we obtain a full characterization of strongly $\mathtt{I}$-normalizing terms by means of $\mathcal{Q}_{\mathtt{I}}$-typability.

In order to achieve the main result of this section, we define an inductive set of objects $\mathcal{ISN}(\mathtt{I})$ containing the set of strongly $\mathtt{I}$-normalizing objects and contained in the set of those that are $\mathcal{Q}_{\mathtt{I}}$-typable. The set $\mathcal{ISN}(\mathtt{I})$ is inspired by the idempotent intersection typing system proposed by Valentini [47], then revisited by Kikuchi [31].

We start by defining the set $\mathcal{ISN}(\mathtt{I})$ as the smallest subset of $\mathtt{I}$-objects satisfying the following properties:

($\mathtt{ax}$) $\mathtt{nil} \in \mathcal{ISN}(\mathtt{I})$.
($\to \mathtt{r}$) If $t \in \mathcal{ISN}(\mathtt{I})$, then $\lambda x.t \in \mathcal{ISN}(\mathtt{I})$.
($\mathtt{hlist}$) If $l \in \mathcal{ISN}(\mathtt{I})$, then $xl \in \mathcal{ISN}(\mathtt{I})$.
($\to \mathtt{l}$) If $t \in \mathcal{ISN}(\mathtt{I})$ and $l \in \mathcal{ISN}(\mathtt{I})$, then $t; l \in \mathcal{ISN}(\mathtt{I})$.
($\mathtt{app}_{nil}$) If $\lambda x.t \in \mathcal{ISN}(\mathtt{I})$, then $(\lambda x.t)\mathtt{nil} \in \mathcal{ISN}(\mathtt{I})$
($\mathtt{app}_{\in}$) If $t\{x/u\} \circ l \in \mathcal{ISN}(\mathtt{I})$ and $x \in \mathtt{fv}(t)$, then $(\lambda x.t)(u; l) \in \mathcal{ISN}(\mathtt{I})$.

($\mathtt{app}_{\notin}$) If $t \circ l \in \mathcal{ISN}(\mathtt{I})$ and $u \in \mathcal{ISN}(\mathtt{I})$ and $x \notin \mathtt{fv}(t)$, then $(\lambda x.t)(u;l) \in \mathcal{ISN}(\mathtt{I})$.

Every strongly $\mathtt{I}$-normalizing object o turns out to be in $\mathcal{ISN}(\mathtt{I})$.

Theorem 2. *If $o \in \mathcal{SN}(\mathtt{I})$, then $o \in \mathcal{ISN}(\mathtt{I})$.*

Proof. By induction on $\langle \eta_{\mathtt{I}}(o), |o| \rangle$.

If $o = \mathtt{nil}$, then the statement is trivial. If $o = u;l$, then the *i.h.* gives u and l in $\mathcal{ISN}(\mathtt{I})$ so that $u;l \in \mathcal{ISN}(\mathtt{I})$ using rule $(\to \mathtt{l})$. The same reasoning can be applied if $o = \lambda x.t$ or $o = xl$. If $o = (\lambda x.t)l$, we consider two cases.

- $l = \mathtt{nil}$. The *i.h.* gives $\lambda x.t \in \mathcal{ISN}(\mathtt{I})$, then $(\lambda x.t)\mathtt{nil} \in \mathcal{ISN}(\mathtt{I})$ using rule $(\mathtt{app}_{nil})$.
- $l = u;l'$. We consider again two cases.
  - $x \in \mathtt{fv}(t)$. Since $\eta_{\mathtt{I}}(t\{x/u\} \circ l) < \eta_{\mathtt{I}}(o)$, then by the *i.h.* we have $t\{x/u\} \circ l \in \mathcal{ISN}(\mathtt{I})$, then we obtain $o \in \mathcal{ISN}(\mathtt{I})$ using rule $(\mathtt{app}_{\in})$.
  - $x \notin \mathtt{fv}(t)$. Since $\eta_{\mathtt{I}}(t\{x/u\} \circ l) = \eta_{\mathtt{I}}(t \circ l) < \eta_{\mathtt{I}}(o)$, then by the *i.h.* we have $t \circ l \in \mathcal{ISN}(\mathtt{I})$. Moreover, $\eta_{\mathtt{I}}(u) < \eta_{\mathtt{I}}(o)$, so that also by the *i.h.* we have $u \in \mathcal{ISN}(\mathtt{I})$. Then $o \in \mathcal{ISN}(\mathtt{I})$ using rule $(\mathtt{app}_{\notin})$.

Moreover, every object in $\mathcal{ISN}(\mathtt{I})$ is $\mathcal{Q}_{\mathtt{I}}$-typable.

Theorem 3. *If $o \in \mathcal{ISN}(\mathtt{I})$, then there exists Φ' such that $\Phi' \triangleright_{\mathcal{Q}_{\mathtt{I}}} \Gamma \mid \Sigma \vdash o{:}\tau$.*

Proof. By induction on the definition of the predicate $\mathcal{ISN}(\mathtt{I})$. The cases $(\mathtt{ax})$, $(\to \mathtt{r})$, $(\mathtt{hlist})$ and $(\to \mathtt{l})$ are straightforward. Let consider $(\lambda x.t)\mathtt{nil} \in \mathcal{ISN}(\mathtt{I})$ coming from $\lambda x.t \in \mathcal{ISN}(\mathtt{I})$ by rule $(\mathtt{app}_{nil})$. By the *i.h.* we have $\Gamma \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} \lambda x.t{:}\tau$, thus we conclude by:

$$\Phi' := \dfrac{\Gamma \mid \_ \vdash \lambda x.t{:}\tau \qquad \dfrac{}{\emptyset \mid \tau \vdash \mathtt{nil}{:}\tau}\,(\mathtt{ax})}{\Gamma \mid \_ \vdash (\lambda x.t)\mathtt{nil}{:}\tau}\,(\mathtt{app})$$

Consider $(\lambda x.t)(u;l) \in \mathcal{ISN}(\mathtt{I})$ coming from $t\{x/u\} \circ l \in \mathcal{ISN}(\mathtt{I})$ and $x \in \mathtt{fv}(t)$ by rule $(\mathtt{app}_{\in})$. By the *i.h.* $\Gamma \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} t\{x/u\} \circ l{:}\tau$, thus we get $\Gamma \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} (\lambda x.t)(u;l){:}\tau$ by Lem. 11.

Consider $(\lambda x.t)(u;l) \in \mathcal{ISN}(\mathtt{I})$ coming from $t \circ l \in \mathcal{ISN}(\mathtt{I})$, $u \in \mathcal{ISN}(\mathtt{I})$ and $x \notin \mathtt{fv}(t)$ by rule $(\mathtt{app}_{\notin})$. By the *i.h.* $\Gamma_1 \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} t \circ l{:}\tau$ and $\Gamma_2 \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} u{:}\sigma$. It can be proved, by induction that there are $\Gamma_1' \mid \_ \vdash_{\mathcal{Q}_{\mathtt{I}}} t{:}\tau'$ and $\Gamma_1'' \mid \tau' \vdash_{\mathcal{Q}_{\mathtt{I}}} l{:}\tau$, where $\Gamma_1 = \Gamma_1' + \Gamma_1''$. Thus we get the following $\mathcal{Q}_{\mathtt{I}}$-derivation:

$$\Phi' := \dfrac{\dfrac{\Gamma_1' \mid \_ \vdash t{:}\tau'}{\Gamma_1' \mid \_ \vdash \lambda x.t{:}[\,] \to \tau'}\,(\to \mathtt{r}) \qquad \dfrac{\Gamma_2 \mid \_ \vdash u{:}\sigma \qquad \Gamma_1'' \mid \tau' \vdash l{:}\tau}{\Gamma_2 + \Gamma_1'' \mid [\,] \to \tau' \vdash u; l{:}\tau}\,(\to \mathtt{l}_{\notin})}{\Gamma_1' + \Gamma_1'' + \Gamma_2 \mid \_ \vdash (\lambda x.t)(u; l){:}\tau}\,(\mathtt{app})$$

We can thus conclude this section with the equivalence between strongly I-normalizing terms and typable terms in system $\mathcal{Q}_{\mathtt{I}}$.

Corollary 4. *$\Gamma \mid \Sigma \vdash_{\mathcal{Q}_{\mathtt{I}}} o{:}\tau$ if and only if $o \in \mathcal{SN}(\mathtt{I})$.*

8 Conclusion

This paper proposes a resource aware computational semantics for Herbelin's syntax. The resulting E-calculus can be seen as a refinement of the non resource aware I-calculus, whose meta-level operations are implemented by more atomic reduction rules in E. In contrast to more complex resource-controlled interpretations (*cf.* [22]) realized by means of explicit control operators for weakening and contraction, our implementation is achieved by rewriting rules inspired from the *substitution at a distance* paradigm, recently used in successful investigations in computer science.

We define typing systems for both calculi I and E, based on relevant, strict, non-idempotent intersection types. Typing rules of the systems are syntax directed. In both cases, typability is used to completely characterize strongly normalizing terms. Our results are presented in a self-contained form, without resorting to their isomorphic natural deduction counterparts. The proofs only use combinatorial arguments, neither reducibility candidates nor memory operators have been necessary.

Balance between typing and reduction systems in a resource aware framework is sensitive; this is illustrated by the approach in [26] and the one taken here. Indeed, adding a dereliction rule to the reduction system, as done in this paper, allows to restrict the witness type derivations to erasable subterms only, while dereliction can be simply omitted, as done in [26], when using a resource consuming approach based on witnesses everywhere.

References

1. B. Accattoli, E. Bonelli, D. Kesner, and C. Lombardi. A nonstandard standardization theorem. In *POPL*, pages 659–670. ACM, 2014.
2. B. Accattoli and D. Kesner. The structural lambda-calculus. In *CSL*, *LNCS* 6247, pages 381–395. Springer-Verlag, 2010.

3. J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.
4. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
5. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics (revised edition)*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, 1984.
6. 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.
7. A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In *FOSSACS*, *LNCS* 6604. Springer-Verlag, 2011.
8. A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.
9. G. Boudol, P.-L. Curien, and C. Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.
10. A. Bucciarelli, D. Kesner, and S. Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *TCS*, LNCS 8705, pages 341–354, Springer-Verlag, 2014.
11. M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for lambda terms. *Archiv für Mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.
12. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame, Journal of Formal Logic*, 21:685–693, 1980.
13. M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Mathematical Logic Quarterly*, 27(2-6):45–58, 1981.
14. F. Damiani and P. Giannini. A decidable intersection type system based on relevance. In *TACS*, LNCS 789, pages 707–725. Springer-Verlag, 1994.
15. E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *ITRS*, EPTCS, pages 48–57. Cornell University Library, 2013.
16. D. de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. These de doctorat, Université Aix-Marseille II, 2007.
17. R. Dyckhoff and C. Urban. Strong normalization of herbelin's explicit substitution calculus with substitution propagation. *Journal of Logic and Computation*, 13(5):689–706, 2003.
18. J. Espírito Santo, J. Ivetic, and S. Likavec. Characterising strongly normalising intuitionistic terms. *Fundamenta Informaticae*, 121(1-4):83–120, 2012.
19. J. Espírito Santo. The lambda-calculus and the unity of structural proof theory. *Theory of Computing Systems*, 45(4):963–994, 2009.
20. J. Espírito Santo. Revisiting the correspondence between cut elimination and normalisation. In *ICALP*, *LNCS* 1853, pages 600–611. Springer-Verlag, 2000.
21. G. Gentzen. The collected papers of Gerhard Gentzen, 1969.
22. S. Ghilezan, J. Ivetic, P. Lescanne, and S. Likavec. Intersection types for the resource control lambda calculi. In *ICTAC*, *LNCS* 6916, pages 116–134. Springer-Verlag, 2011.
23. J.-Y. Girard. Proof-nets: The parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124. Marcel Dekker, 1996.
24. J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.
25. H. Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *CSL*, *LNCS* 933, pages 61–75. Springer-Verlag, 1995.

26. D. Kesner and D. Ventura. Quantitative types for intuitionistic calculi. Technical Report hal-00980868, Paris Cité Sorbonne, 2014.
27. D. Kesner and D. Ventura. Quantitative types for the linear substitution calculus. In *TCS*, LNCS 8705, pages 296-310 Springer-Verlag, 2014.
28. A. Kfoury. A linearization of the lambda-calculus and consequences. Technical report, Boston University, 1996.
29. A. Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000.
30. A. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables. *Theoretical Computer Science*, 311(1-3):1–70, 2004.
31. K. Kikuchi. Uniform proofs of normalisation and approximation for intersection types. In *ITRS*, Vienna, Austria, 2014.
32. J. W. Klop. *Combinatory reduction systems.* PhD thesis, Univ. Utrecht, 1980.
33. J.-L. Krivine. *Lambda-calculus, types and models.* Ellis Horwood, 1993.
34. S. Lengrand. Deriving strong normalisation. In *HOR*, pages 84–88, 2004. Available at `http://www-i2.informatik.rwth-aachen.de/HOR04/`.
35. S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Information and Computation*, 189:17–42, 2004.
36. D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
37. R. Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). In *ENTCS* 175(3):65–73, 2007.
38. P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In *ICFP*, pages 138–149. ACM, 2004.
39. L. Ong and S. J. Ramsay. Verifying higher-order functional programs with pattern matching algebraic data types. In *POPL*, pages 587–598. ACM, 2011.
40. M. Pagani and S. Ronchi Della Rocca. Solvability in resource lambda-calculus. In *FOSSACS*, *LNCS* 6014, pages 358–373. Springer-Verlag, 2011.
41. G. Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.
42. G. Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–578. Academic Press, 1980.
43. D. Prawitz. *Natural deduction: a proof-theoretical study.* Phd thesis, Stockholm University, 1965.
44. L. Regnier. Une équivalence sur les lambda-termes. *Theoretical Computer Science*, 2(126):281–292, 1994.
45. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic* , 32(2):198–212, 1967.
46. P. Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, 64(3):1195–1215, 1999.
47. S. Valentini. An elementary proof of strong normalization for intersection types. *Archive of Mathematical Logic*, 40(7):475–488, 2001.
48. S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.
49. J. Zucker. The correspondence between cut-elimination and normalization I. *Annals of Mathematical Logic*, 7:1–112, 1974.