Non-idempotent intersection types for the Lambda-Calculus

ANTONIO BUCCIARELLI* and DELIA KESNER**, *Institut de Recherche en Informatique Fondamentale, CNRS and Université Paris Diderot, Paris, France.*

DANIEL VENTURA†, *Universidade Federal de Goiás, Instituto de Informática, Goiânia, Brazil.*

Abstract

This article explores the use of non-idempotent intersection types in the framework of the λ-calculus. Different topics are presented in a uniform framework: head normalization, weak normalization, weak head normalization, strong normalization, inhabitation, exact bounds and principal typings. The reducibility technique, traditionally used when working with idempotent types, is replaced in this framework by trivial combinatorial arguments.

Keywords: Type Systems, intersection types, quantitative semantics, normalization properties, inhabitation problems, principal typing.

1 Introduction

1.1 Intersection types for λ-Calculus

Intersection types, IT for short, were originally introduced for describing and capturing computational/functional properties of λ-terms. For instance, they provided the first characterization of *strongly normalizing* λ-terms [46, 55]. They extend simple types with a new constructor $\cap$ in such a way that $\sigma \cap \tau$ is a type if σ and τ are. Then, a term t typable with $\sigma \cap \tau$ can be considered either of type σ or τ, so that, e.g., if the variable x is typable with $(\sigma \to \sigma) \cap \sigma$, then the term xx has type σ. As a consequence, the strongly-normalizing term $\lambda x.xx$, which is not typable with simple types, turns out to be typable with intersection types.

Since the seminal papers published in the 1980 [15, 17], intersection types have been used to characterize properties of λ-terms in a broader sense, as well as to analyze and synthetize models of the λ-calculus [2]. For example, characterizations of *head normalizing* terms and *weakly normalizing* terms can be found in [19].

Also, IT opened interesting instances of classical type systems related problems. For instance, the *inhabitation problem* for IT was shown to be undecidable in [66] and the *principal typing* property for IT was also investigated in [18], followed by many other works [44, 59, 70].

Concerning more practical applications, IT have been used in the design of programming languages. The first example is the type system of Forsythe in [57], and more recent approach combine IT and other features, such as *refinement* [31], *liquid* [53] and *union* types [28, 29]. Besides type systems, program analysis using IT ranges from *control-flow analysis* [47, 52] to *model-checking*

*E-mail: buccia@irif.fr
**E-mail: kesner@irif.fr
†E-mail: daniel@inf.ufg.br

Vol. 25 No. 4, © The Author 2017. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
doi:10.1093/jigpal/jzx018 Advance Access published 3 July 2017

[56]. Moreover, IT combined with *union* types are used in typing systems for the π-calculus [61], to investigate termination properties [54] and as a semantics for *session types* [14, 49, 50].

An exhaustive survey of the use of idempotent IT is out of the scope of this article. The interested reader may refer to [3, 33, 70] for an overview.

1.2 Non-idempotent intersection types for λ-Calculus

All intersection types mentioned above enjoy associativity ($(\sigma \cap \tau) \cap \rho = \sigma \cap (\tau \cap \rho)$), commutativity ($\sigma \cap \tau = \tau \cap \sigma$), and in particular idempotency ($\sigma \cap \sigma = \sigma$). A tight correspondence between the *non-idempotent* intersection and the multiplicative connective of Linear Logic [35] — the logic of resources — gives a first insight about the rôle of non-idempotent intersection systems in terms of resource management refinement. As a matter of notation, remark that associativity, commutativity and idempotence are granted if intersections are denoted by sets, e.g. $\{\tau, \sigma\}$ stands for $\tau \cap \sigma$. This idea leads naturally to represent non-idempotent intersections by multisets, e.g. $[\tau, \tau, \sigma]$ stands for $\tau \cap \tau \cap \sigma$. We follow this convention throughout this article.

Just like their idempotent precursors, non-idempotent intersection type systems allow for a characterization of strong normalization [5, 23], weak normalization and head normalization [24], but they also grant a substantial improvement: proving that *typable terms are strongly normalizing* becomes much simpler. Let us provide a brief account of this improvement, by highlighting in the way the *quantitative* character of non-idempotent intersection types versus the *qualitative* flavour of the idempotent ones. The proof of the highlighted statement above, in the non-idempotent case, goes roughly as follows: given a typing derivation for a term t, and willing to prove that t is strongly normalizing, take whatever β-reduct t' of t. The subject reduction lemma, in this case, ensures *not only* that t' is typable *but also* that there exists a typing derivation for t' whose size is smaller than the one of the typing derivation for t we started from. Hence any β-reduction sequence starting from t is finite. This shrinking of the size of typing derivations along reduction sequences, in sharp contrast to what happens in the idempotent setting, is essentially due to the fact that a type derivation for a term of the shape $(\lambda x.u)v$ may require as many sub-derivations for v as the number of occurrences of x in u. Let us provide a simple example involving the Church numeral $\underline{n} := \lambda y.\lambda x.y(y(...yx)...)$. Why is the term $u = \lambda x.t(t(...tx)...)$, t being an arbitrarily complex term, 'simpler to type' than its β-expanded form $\underline{n}t$? The point is that the typical non-idempotent intersection type that can be assigned to the Church numeral $\underline{n}$ is, in our notation, $[[\sigma] \to \sigma, ..., [\sigma] \to \sigma] \to [\sigma] \to \sigma$, the leftmost multiset containing n copies of $[\sigma] \to \sigma$. Thus, to assign a type to $\underline{n}t$, n typing derivations assigning $[\sigma] \to \sigma$ to t must be provided, exactly like in a type derivation for u. At the same time, the outermost application $\underline{n}t$ vanishes with the reduction $\underline{n}t \to_\beta u$, so the typing derivation for u is smaller than that for $\underline{n}t$. In the idempotent case, on the other hand, a type for $\underline{n}$ is an instance of $\{\{\sigma\} \to \sigma\} \to \{\sigma\} \to \sigma$, and the typing derivations for u may be hugely bigger than those for $\underline{n}t$, the former requiring n sub-derivations for t, the latter just one. Thus the proof of the result above for idempotent intersection type systems are typically based on *reducibility* arguments [36, 46, 65]. However, a *combinatorial* proof of strong normalization for idempotent intersection types is provided in [45] by considering multisets of redexes, somehow anticipating the use of multisets of types, i.e. of non-idempotent intersection types. This shift of perspective goes beyond lowering the logical complexity of the proof: the quantitative information provided by typing derivations in the non-idempotent setting unveils interesting relations between typings (static) and reductions (dynamic) of λ-terms. For instance, in [24], a correspondence is presented between the size of a typing derivation for t and the number of steps taken by the Krivine machine to reduce t, and in [5] it is shown how to compute the length of the longest β-reduction sequence starting from any typable strongly normalizing λ-term.

A first non-idempotent IT system is defined in [32] to identify needed redexes in the λ-calculus. Independently, non-idempotent IT systems are presented in [42, 43] to study the linearization of the λ-calculus, and in [44], where a typing inference algorithm is presented. Also, a *resource aware semantics* of the λ-calculus is obtained in [6] by means of a non-idempotent IT system. D. de Carvalho [24] established a relation between the size of a typing derivation in a relevant non-idempotent intersection type system for the λ-calculus and the head/weak-normalization execution time of head/weak-normalizing λ-terms, respectively. Termination characterizations are proved by the usual reducibility candidates argument (e.g. [20]) and the correspondence is stated for a modified Krivine machine to compute the corresponding head or β-normal forms.

In [5] a non-idempotent intersection type system was used to prove the exact relation between the size of typing derivations and the number of β-steps in the longest reduction sequence of strong-normalizing terms in the λ-calculus. Also, in [23] intersection types neither idempotent nor associative are used to give an upper bound on the number of steps necessary to reduce a β-normalizing term t to its normal form.

In [32] non-idempotent IT are also related to *needed redexes*: a redex r in a term t is *needed* if and only if r, or some *residual* of r, is reduced in *every* reduction of t to its normal form. A completeness proof for call-by-need with respect to call-by-name is developed in [39] by means of non-idempotent IT.

The inhabitation problem, known to be undecidable in an idempotent setting [66], was proved to be decidable for non-idempotent types [7]. In particular [7] gives a characterization of solvable terms and head-normalizing terms by means of non-idempotent IT.

1.3 Non-idempotent intersection types for other higher-order languages

Besides all the applications of non-idempotent IT systems to the λ-calculus, many different higher-order languages were also studied by means of this technology. Here we give a non-exhaustive list. Solvability is characterized by means of non-idempotent IT in a lambda-calculus with resources [51]. Strong normalization is characterized in [5] for calculi with explicit substitution such as `λs` and `λlxr`. Non-idempotent IT systems were used in [40, 41] to characterize different normalization properties of higher-order calculi. For example, *linear-head*, *head*, *weak* and *strong* normalization are characterized in [40] by means of appropriate non-idempotent types in the framework of the linear substitution calculus [1], a calculus with explicit substitution at a distance. A resource aware computational interpretation for Herberlin's syntax [38] is investigated in [41] for the intuitionistic focused sequent calculus. Inspired by the notion of solvability in the λ-calculus, a notion of observability for a calculus with pattern matching is characterized in [8]. A calculus with constructors and fixpoints is studied in [4], where strong normalization is obtained through an appropriate non-idempotent IT system.

In [30], strong normalization for a call-by-value λ-calculus is characterized by means of intersection types, both idempotent and non-idempotent. It is shown that the idempotent types of a term can be recovered by applying a suitable logical relation, the *extensional collapse*, to its non-idempotent types. Following the same approach, and exploiting in particular the validity of the *Taylor formula* in the *relational model* of the λ-calculus and a *resource call-by-value λ-calculus* presented in [30], [13] provides a characterization of solvability for a call-by-value λ-calculus in terms of non-idempotent IT[1]. Game semantics of a typed λ-calculus has been logically described in [34] using an IT system where intersection is neither commutative nor associative.

[1]More precisely, [13] deals with the interpretation of call-by-value λ-terms in the relational model, rather than with their non-idempotent typings, but these are equivalent notions, as showed in [24] in the case of the usual, call-by-name, λ-calculus.

1.4 Contribution

The goal of this article is to gather together different results obtained by means of non-idempotent IT in the framework of the λ-calculus. Some of them are already published, some are revisited, some are new.

New results include (1) the characterization of strong normalization in Section 8, inspired by the corresponding construction for idempotent IT [3, 67], (2) the characterization of weak normalization in Section 6, inspired by [24], where the same result is proved using reducibility, and (3) some properties concerning the inhabitation problem for system $\mathcal{S}$ in Section 10, that lead us to conjecture its decidability. Revisited results include those of Section 11 on principal typings, where we follow [62, 63], and those of Section 9, on the relation between the size of type derivations in non-idempotent IT systems and the complexity of the normalization process, where we follow [5]. The already published results that we present for the sake of completeness include the characterization of head normalization appearing in [32], presented in Section 5 and the characterization of weak head normalization appearing in [39], presented in Section 7.

1.5 Structure of the paper

In Section 2 we present the syntax and semantics of the λ-calculus together with some key properties that we are going to characterize later on. In Section 3 we introduce our main IT systems $\mathcal{W}$ and $\mathcal{S}$, and establish some of their basic properties. Section 4 shows the weighted subject reduction property and the subject expansion property for both systems $\mathcal{W}$ and $\mathcal{S}$. Section 5 gives the characterization of head β-normalizing terms through system $\mathcal{W}$ by both reducibility and combinatorial arguments. The characterization of weak β-normalization through system $\mathcal{W}$ is established in Section 6, while Section 7 is devoted to the characterization of weakly head β-normalizing terms by means of the system $\mathcal{WH}$, which is a variant of system $\mathcal{W}$. The characterization of strong β-normalization through system $\mathcal{S}$ is established in Section 8, while in Section 9 a modification of system $\mathcal{S}$ is introduced to obtain exact (upper) bounds for the normalization process. Section 10 recalls the decidability of the inhabitation problem for $\mathcal{W}$ and approaches the same problem for $\mathcal{S}$. In Section 11 the principal typing property is shown for system $\mathcal{W}$, for weakly β-normalizing terms.

2 Preliminaries

Given a countable infinite set of symbols $x, y, z, \ldots$ we define the set of **terms**, written Λ, and the set of **contexts** of the λ-calculus by means of the following grammars.

$$\begin{array}{lllll} \textbf{(terms)} & t,u & ::= & x \mid \lambda x.t \mid tu \\ \textbf{(contexts)} & C & ::= & \Box \mid \lambda x.C \mid Ct \mid tC. \end{array}$$

Special λ-terms are $\mathtt{I} = \lambda x.x$, $\Delta = \lambda x.xx$ and $\Omega = \Delta\Delta$. The sets of **free** and **bound** variables of a term t, written respectively $\mathtt{fv}(t)$ and $\mathtt{bv}(t)$, are defined as usual. We work with the standard notion of **α-conversion** i.e. renaming of bound variables for abstractions.

A **head-context** is a context of the form $(\lambda x_1 \ldots x_n.\Box)t_1 \ldots t_m$ for some $n, m \geq 0$. The set of **occurrences** in a term t, written $\circ(t)$, is the finite language over $\{0,1\}$ inductively defined as follows: $\epsilon \in \circ(t)$ for every t; $0w \in \circ(\lambda x.t)$ if $w \in \circ(t)$; $0w \in \circ(tu)$ if $w \in \circ(t)$; $1w \in \circ(tu)$ if $w \in \circ(u)$. The set of **occurrences** in a context C, written $\circ(C)$, is the finite language over $\{0,1\}$ inductively defined as follows: $\epsilon \in \circ(C)$ for every C; $0w \in \circ(\lambda x.C)$ if $w \in \circ(C)$; $0w \in \circ(Cu)$ if $w \in \circ(C)$; $1w \in \circ(tC)$ if

$w \in \circ(C)$. The **subterm of** t **at the occurrence** w is written $t|_w$ and defined as expected. We write $C[t]_w$ (or simply $C[t]$ for short) for the term obtained by replacing the hole $\Box$ at the occurrence w of C by the term t. The term u **has an occurrence** in t iff there is $w \in \circ(t)$ such that $t|_w = u$, i.e. if there is C and w such that $t = C[u]_w$.

To define the operational semantics of the λ-calculus we consider the following rewriting rule:

$$(\lambda x.t)u \mapsto_\beta t\{x/u\}.$$

The **reduction relation** $\to_\beta$ is generated by the closure of the rewriting rule $\mapsto_\beta$ by all the contexts C. A term of the form $(\lambda x.t)u$ is called a β**-redex**; we write $t \xrightarrow{w}_\beta t'$ iff $t \to_\beta t'$ reduces the β-redex at occurrence $w \in \circ(t)$, i.e. if $t = C[(\lambda x.u)v]_w$ and $t' = C[u\{x/v\}]_w$. A reduction step $C[(\lambda x.u)v] \to_\beta C[u\{x/v\}]$ is said to be **non-erasing** iff $x \in \mathtt{fv}(u)$.

We denote by $\to_\beta^*$ (resp. $\to_\beta^+$) the **reflexive-transitive** (resp. **transitive**) closure of $\to_\beta$. We write $\to_\beta^n$ to denote the self-composition of $\to_\beta$ exactly $n \geq 0$ times, i.e. $t \to_\beta^0 t$ and $t_0 \to_\beta^{n+1} t_{n+1}$ if $t_0 \to_\beta^n t_n$ and $t_n \to_\beta t_{n+1}$.

We say that t is a **head** β**-normal form**, written $t \in \mathcal{HNF}(\beta)$, iff t is of the form $\lambda x_1 \ldots x_n.yt_1 \ldots t_m$ for some $n \geq 0$ and some $m \geq 0$; and t **has a head** β**-normal form**, or is **head** β**-normalizing**, written $t \in \mathcal{HN}(\beta)$, iff there is $t' \in \mathcal{HNF}(\beta)$ such that $t \to_\beta^* t'$. A term t is a **weak head** β**-normal form**, written $t \in \mathcal{WHNF}(\beta)$, iff t is an abstraction or $t \in \mathcal{HNF}(\beta)$; and t **has a weak head** β**-normal form**, or is **weakly head** β**-normalizing**, written $t \in \mathcal{WHN}(\beta)$, iff there is $t' \in \mathcal{WHNF}(\beta)$ such that $t \to_\beta^* t'$. Similarly, we say that t is **in** β**-normal form**, written $t \in \mathcal{NF}(\beta)$, if there is no t' such that $t \to_\beta t'$; and t **has a** β**-normal form** iff there is $t' \in \mathcal{NF}(\beta)$ such that $t \to_\beta^* t'$; Remark that $\mathcal{NF}(\beta) \subseteq \mathcal{HNF}(\beta) \subseteq \mathcal{WHNF}(\beta)$ but the converse does not hold: $\lambda x.\Omega \in \mathcal{WHNF}(\beta)$ but $\lambda x.\Omega \notin \mathcal{HNF}(\beta)$. Similarly, $\lambda x.x\Omega \in \mathcal{HNF}(\beta)$ but $\lambda x.x\Omega \notin \mathcal{NF}(\beta)$.

A term t is **weakly** β**-normalizing**, written $t \in \mathcal{WN}(\beta)$, iff t has a β-normal form, i.e. if there is some finite β-reduction sequence starting at t. A term t is **strongly** β**-normalizing**, written $t \in \mathcal{SN}(\beta)$, if there is no infinite β-reduction sequence starting at t. Observe that every t is β**-finitely branching**, i.e. the set $\{o' \mid o \to_\beta o'\}$ is finite. As a consequence, the **depth of** t, written $\eta_\beta(t)$, can be defined as the maximal length of β-reduction sequences starting at t.

3 The type systems

Let $\mathcal{A}$ be a countable infinite set of type variables $\alpha, \beta, \gamma, \ldots$. **Strict types** and **multiset types** are defined by the following grammars:

(strict types)	σ, τ	::=	$\alpha \mid \mathcal{M} \to \sigma$
(multiset types)	$\mathcal{M}, \mathcal{N}$	::=	$[\sigma_i]_{i \in I}$, I finite set.

Types are *strict*[2] i.e. multiset types do not occur on the right-hand sides of arrows. A multiset type should be read as the intersection of the strict types it contains. For instance, the multiset $[\tau, \tau, \sigma]$ stands for $\tau \wedge \tau \wedge \sigma$, where the symbol $\wedge$ is associative, commutative and non-idempotent. The empty multiset is denoted by $[]$. Observe however that the commutativity, associativity and non-idempotency[3] of the intersection symbol are granted by the multiset notation: no further equivalence relation on types is needed.

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 $\mathtt{dom}(\Gamma) := \{x \mid \Gamma(x) \neq$

[2]The terminology is due to S. van Bakel [68].

[3]Differently from [18] that explicitly defines idempotency although it has the same notation for IT.

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

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

FIG. 1. The intersection type system $\mathcal{W}$ for the λ-calculus

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

$$\frac{\Gamma \vdash t{:}[\,] \to \tau \qquad \Delta \vdash u{:}\sigma}{\Gamma + \Delta \vdash tu{:}\tau}\ (\to \mathtt{e}_1)$$

$$\frac{I \neq \emptyset \quad \Gamma \vdash t{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash u{:}\sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash tu{:}\tau}\ (\to \mathtt{e}_2)$$

FIG. 2. The intersection type system $\mathcal{S}$ for the λ-calculus

[]}. The **intersection of type assignments**, written $\Gamma + \Delta$, is defined by $(\Gamma + \Delta)(x) := \Gamma(x) + \Delta(x)$, where $+$ denotes multiset union. Hence, $\mathtt{dom}(\Gamma + \Delta) = \mathtt{dom}(\Gamma) \cup \mathtt{dom}(\Delta)$. We write $\Gamma \setminus\setminus x$ for the assignment $(\Gamma \setminus\setminus x)(x) = []$ and $(\Gamma \setminus\setminus x)(y) = \Gamma(y)$ if $y \neq x$. When $\mathtt{dom}(\Gamma)$ and $\mathtt{dom}(\Delta)$ are disjoint we use $\Gamma; \Delta$ instead of $\Gamma + \Delta$, and we write $x : [\sigma_i]_{i\in I}; \Gamma$, even when $I = \emptyset$, for the assignment $(x : [\sigma_i]_{i\in I}; \Gamma)(x) = [\sigma_i]_{i\in I}$ and $(x : [\sigma_i]_{i\in I}; \Gamma)(y) = \Gamma(y)$ if $y \neq x$. We write $x\#\Gamma$ iff $x \notin \mathtt{dom}(\Gamma)$.

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 intersection type systems $\mathcal{W}$ and $\mathcal{S}$ for the λ-calculus are presented in Figures 1 and 2 respectively. 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 are Γ and τ such that $\Gamma \vdash_X t{:}\tau$. 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 the derivation** Φ, denoted by $\mathtt{sz}(\Phi)$, is defined as the number of nodes of the correspoding derivation tree.

In rule $(\to \mathtt{e}_1)$, the argument u is erasable, so that we require exactly one typing derivation for u (the *typing witness*), even if its type σ is ignored in the conclusion of the rule. Sometimes, to save some space in our proofs, we will capture the rules $(\to \mathtt{e}_1)$ and $(\to \mathtt{e}_2)$ by the single generalised rule $(\to \mathtt{e}_g)$:

$$\frac{\Gamma \vdash t{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Delta_j \vdash u{:}\sigma_j)_{j\in J}}{\Gamma +_{j\in J} \Delta_j \vdash tu{:}\tau}\ (\to \mathtt{e}_g)$$
$$\text{where } (I = \emptyset \Rightarrow |J| = 1) \text{ and } (I \neq \emptyset \Rightarrow J = I).$$

Here is an example of type derivation in system $\mathcal{W}$:

$$\dfrac{\dfrac{\dfrac{}{z{:}[\gamma] \vdash z{:}\gamma}}{z{:}[\gamma] \vdash \lambda y.z{:}[] \to \gamma}}{z{:}[\gamma] \vdash (\lambda y.z)(\lambda x.xx){:}\gamma}.$$

Here is an example of type derivation in system $\mathcal{S}$:

$$\dfrac{\dfrac{}{z{:}[\gamma]\vdash z{:}\gamma}}{\dfrac{z{:}[\gamma]\vdash \lambda y.z{:}[\,]\to\gamma}{z{:}[\gamma]\vdash(\lambda y.z)(\lambda x.xx){:}\gamma}}\qquad\dfrac{\dfrac{\dfrac{}{x{:}[[\sigma]\to\tau]\vdash x{:}[\sigma]\to\tau}\qquad\dfrac{}{x{:}[\sigma]\vdash x{:}\sigma}}{x{:}[[\sigma]\to\tau],[\sigma]\to\tau]\vdash xx{:}\tau}}{\vdash\lambda x.xx{:}[[\sigma]\to\tau],[\sigma]\to\tau]\to\tau}$$

Remark that neither $(\lambda y.z)\Omega$ nor $z\Omega$ are typable in $\mathcal{S}$, for $\Omega=(\lambda x.xx)(\lambda x.xx)$.

3.1 Basic static properties

This section presents some basic static properties of the typing systems.

The systems $\mathcal{W}$ and $\mathcal{S}$ are *relevant* in the sense of [22]. Relevant type systems reject by design the weakening rule, hence there are no unnecessary assumptions in the type context. This is a key property used in the study of principal typings for intersection types systems [18] and in the description of the functional character of solvable terms in the λ-calculus [19].

LEMMA 3.1 (Relevance)
(1) If $\Phi\rhd\Gamma\vdash_{\mathcal{W}} t{:}\tau$ then $\mathtt{dom}(\Gamma)\subseteq\mathtt{fv}(t)$.
(2) If $\Phi\rhd\Gamma\vdash_{\mathcal{S}} t{:}\tau$ then $\mathtt{dom}(\Gamma)=\mathtt{fv}(t)$.

By induction on Φ. ■

LEMMA 3.2 (Substitution Lemma)
Let $X\in\{\mathcal{W},\mathcal{S}\}$. If $\Phi\rhd x{:}[\rho_i]_{i\in I};\Gamma\vdash_X t{:}\tau$ and $(\Phi^i_u\rhd\Delta_i\vdash_X u{:}\rho_i)_{i\in I}$ then $\Phi'\rhd\Gamma+_{i\in I}\Delta_i\vdash_X t\{x/u\}{:}\tau$ where $\mathtt{sz}(\Phi')=\mathtt{sz}(\Phi)+_{i\in I}\mathtt{sz}(\Phi^i_u)-|I|$. In particular, if $I=\emptyset$, we have $\mathtt{sz}(\Phi')=\mathtt{sz}(\Phi)$.

The proof is by induction on Φ.

- (ax) $t=x$ and $x\{x/u\}=u$. By construction one has that $x:[\rho_i]_{i\in I};\Gamma=\{x:[\tau]\}$ so that $|I|=1$ (let say $I=\{a\}$) and $\Gamma=\emptyset$ and $[\rho_i]_{i\in I}=[\tau]$ and $\mathtt{sz}(\Phi)=1$. Therefore, $\Phi^a_u\rhd\Delta_a\vdash_X x\{x/u\}{:}\tau$. Moreover, $\mathtt{sz}(\Phi')=\mathtt{sz}(\Phi^a_u)=1+\mathtt{sz}(\Phi^a_u)-1=\mathtt{sz}(\Phi)+\mathtt{sz}(\Phi^a_u)-|I|$.
- (ax) $t=y$, $y\neq x$ and $t\{x/u\}=y$. By construction one has that $x:[\rho_i]_{i\in I};\Gamma=\{y:[\tau]\}$, so that $I=\emptyset$. Hence $\mathtt{sz}(\Phi')=\mathtt{sz}(\Phi)$ and the result thus trivially holds.
- ($\to$ i) $t=\lambda y.v$ and $\tau=\mathcal{M}\to\sigma$ and $\Phi_v\rhd y{:}\mathcal{M};x{:}[\rho_i]_{i\in I};\Gamma\vdash_X v{:}\sigma$ where $\mathtt{sz}(\Phi_v)+1=\mathtt{sz}(\Phi)$. By the *i.h.* $\Phi_{v\{x/u\}}\rhd(y{:}\mathcal{M};\Gamma)+_{i\in I}\Delta_i\vdash_X v\{x/u\}{:}\sigma$ where $\mathtt{sz}(\Phi_{v\{x/u\}})$ is equal to $\mathtt{sz}(\Phi_v)+_{i\in I}\mathtt{sz}(\Phi^i_u)-|I|$. By α-conversion we can assume that $y\notin\mathtt{fv}(u)$ thus $y\notin\mathtt{dom}(\Delta_i)$ for any $i\in I$, which implies $(y{:}\mathcal{M};\Gamma)+_{i\in I}\Delta_i=y{:}\mathcal{M};\Gamma+_{i\in I}\Delta_i$. Hence,

$$\Phi':=\dfrac{\Phi_{v\{x/u\}}:=\dfrac{\vdots}{y{:}\mathcal{M};\Gamma+_{i\in I}\Delta_i\vdash_X v\{x/u\}{:}\sigma}}{\Gamma+_{i\in I}\Delta_i\vdash_X\lambda y.v\{x/u\}{:}\mathcal{M}\to\sigma}.$$

Moreover, $\mathtt{sz}(\Phi')=\mathtt{sz}(\Phi_{v\{x/u\}})+1=\mathtt{sz}(\Phi)+_{i\in I}\mathtt{sz}(\Phi^i_u)-|I|$.

- $(\to \mathtt{e})$ $t = vr$ and $x{:}[\rho_i]_{i\in I}; \Gamma = \Delta + _{j\in J} \Gamma_j$ where $\Phi_v \rhd \Delta \vdash_{\mathcal{W}} v{:}[\sigma_j]_{j\in J} \to \tau$ and $(\Phi^j_r \rhd \Gamma_j \vdash_{\mathcal{W}} r{:}\sigma_j)_{j\in J}$, where $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_v) + _{j\in J} \mathtt{sz}(\Phi^j_r) + 1$. Let $\{I_v\} \cup \{I_j \mid j \in J\}$ be a partition of I such that $\Delta = x{:}[\rho_i]_{i\in I_v}; \Delta'$ and $(\Gamma_j = x{:}[\rho_i]_{i\in I_j}; \Gamma'_j)_{j\in J}$ for some $\Delta', (\Gamma'_j)_{j\in J}$. Observe that $\Gamma = \Delta' + _{j\in J} \Gamma'_j$. By the *i.h.* $\Phi_{v\{x/u\}} \rhd \Delta' + _{i\in I_v} \Delta_i \vdash_{\mathcal{W}} v\{x/u\}{:}[\sigma_j]_{j\in J} \to \tau$ where $\mathtt{sz}(\Phi_{v\{x/u\}}) = \mathtt{sz}(\Phi_v) + _{i\in I_v} \mathtt{sz}(\Phi^i_u) - |I_v|$ and, for each $j \in J$, $\Phi^j_{r\{x/u\}} \rhd \Gamma'_j + _{i\in I_j} \Delta_i \vdash_{\mathcal{W}} r\{x/u\}{:}\sigma_j$ where $\mathtt{sz}(\Phi^j_{r\{x/u\}}) = \mathtt{sz}(\Phi^j_r) + _{i\in I_j} \mathtt{sz}(\Phi^i_u) - |I_j|$.
  Therefore,

$$\Phi' := \frac{\Phi_{v\{x/u\}} \qquad \left(\Phi^j_{r\{x/u\}}\right)_{j\in J}}{\Delta' + _{i\in I_v} \Delta_i + _{j\in J} \Gamma'_j + _{i\in I_j} \Delta_i \vdash v\{x/u\} r\{x/u\}{:}\tau}.$$

  We conclude with the first statement of the lemma since $\Delta' + _{i\in I_v} \Delta_i + _{j\in J} \Gamma'_j + _{i\in I_j} \Delta_i = \Gamma + _{i\in I_v} \Delta_i + _{j\in J} (+ _{i\in I_j} \Delta_i) = \Gamma + _{i\in I} \Delta_i$. Moreover, $\mathtt{sz}(\Phi') = \mathtt{sz}(\Phi_{v\{x/u\}}) + _{j\in J} \mathtt{sz}(\Phi^j_{r\{x/u\}}) + 1 = \mathtt{sz}(\Phi_v) + _{i\in I_v} \mathtt{sz}(\Phi^i_u) - |I_v| + _{j\in J} (\mathtt{sz}(\Phi^j_r) + _{i\in I_j} \mathtt{sz}(\Phi^i_u) - |I_j|) + 1 = \mathtt{sz}(\Phi) + _{i\in I} \mathtt{sz}(\Phi^i_u) - |I|$.
- $(\to \mathtt{e}_g)$ $t = vr$ and $x{:}[\rho_i]_{i\in I}; \Gamma = \Delta + _{k\in K} \Gamma_k$ where $\Phi_v \rhd \Delta \vdash_{\mathcal{S}} v{:}[\sigma_j]_{j\in J} \to \tau$ and $(\Phi^k_r \rhd \Gamma_k \vdash_{\mathcal{S}} r{:}\sigma_k)_{k\in K}$, $J = \emptyset$ implies $|K| = 1$ and $J \neq \emptyset$ implies $J = K$. Moreover, $\mathtt{sz}(\Phi) = \mathtt{sz}(\Phi_v) + _{k\in K} \mathtt{sz}(\Phi^k_r) + 1$. The proof proceeds similarly to the previous case by using the *i.h.*

■

Lemma 3.3 (Reverse Substitution Lemma)
Let $X \in \{\mathcal{W}, \mathcal{S}\}$. If $\Phi \rhd \Gamma \vdash_X t\{x/u\}{:}\tau$. Then $\exists \Gamma_0, \exists I, \exists (\Gamma_i)_{i\in I}, \exists (\sigma_i)_{i\in I}$ such that:

- $\Gamma = \Gamma_0 + _{i\in I} \Gamma_i$, and
- $x{:}[\sigma_i]_{i\in I}; \Gamma_0 \vdash_X t{:}\tau$, and
- $(\Gamma_i \vdash_X u{:}\sigma_i)_{i\in I}$.

By induction on Φ. The first item and the case $u = z$ in the second item are considered the base cases of the induction.

- $(\mathtt{ax})$ $t = y$, $y \neq x$, and $y\{x/u\} = y$. By construction one has that $\Gamma = y{:}[\tau]$. The result thus holds for $I = \emptyset$ and $\Gamma_0 = \Gamma$.
- $(\mathtt{ax})$ $t = x$ and $x\{x/u\} = u$. By $(\mathtt{ax})$ one has that $x{:}[\tau] \vdash_X x{:}\tau$. Therefore, the result holds for $\Gamma_0 = \emptyset$, $|I| = 1$, $\sigma_1 = \tau$ and $\Gamma_1 = \Gamma$.
- $(\to \mathtt{i})$ $t = \lambda y.v$ and $(\lambda y.v)\{x/u\} = \lambda y.v\{x/u\}$. By construction one has $\tau = \mathcal{M} \to \sigma$, for some $\mathcal{M}$ and some σ and $y{:}\mathcal{M}; \Gamma \vdash_X v\{x/u\}{:}\sigma$. By the *i.h.* $y{:}\mathcal{M}; \Gamma = \Gamma_0 + _{i\in I} \Gamma_i$ and $x{:}[\sigma_i]_{i\in I}; \Gamma_0 \vdash_X v{:}\sigma$ and $(\Gamma_i \vdash_X u{:}\sigma_i)_{i\in I}$. By α-conversion we can assume that $y \notin \mathtt{fv}(u)$, so that $y \notin \mathtt{dom}(\Gamma_i)$ and thus $\Gamma_0 = y{:}\mathcal{M}; \Gamma'_0$ and $\Gamma = \Gamma'_0 + _{i\in I} \Gamma_i$. Hence, we obtain $x{:}[\sigma_i]_{i\in I}; \Gamma'_0 \vdash_X \lambda y.v{:}\tau$ by the rule $(\to \mathtt{i})$.
- $(\to \mathtt{e})$ $t = vr$ and $(vr)\{x/u\} = v\{x/u\} r\{x/u\}$. By construction one has that $\Gamma = \Delta + _{j\in J} \Delta_j$ and $\Delta \vdash_{\mathcal{W}} v\{x/u\}{:}[\rho_j]_{j\in J} \to \tau$ and $(\Delta_j \vdash_{\mathcal{W}} r\{x/u\}{:}\rho_j)_{j\in J}$. By the *i.h.* $\Delta = \Delta_0 + _{i\in I_v} \Delta_i$ and $x{:}[\sigma_i]_{i\in I_v}; \Delta_0 \vdash_{\mathcal{W}} v{:}[\rho_j]_{j\in J} \to \tau$ and $(\Delta_i \vdash_{\mathcal{W}} u{:}\sigma_i)_{i\in I_v}$, and, $(\Delta_j = \Delta^j_0 + _{i\in I_j} \Delta^j_i)_{j\in J}$, $(x{:}[\sigma^j_i]_{i\in I_j}; \Delta^j_0 \vdash_{\mathcal{W}} r{:}\rho_j)_{j\in J}$ and $((\Delta^j_i \vdash_{\mathcal{W}} u{:}\sigma^j_i)_{i\in I_j})_{j\in J}$. We can suppose w.l.o.g. that I_v and $(I_j)_{j\in J}$ are pairwise disjoint sets. Let $I = I_v \cup _{j\in J} I_j$. Then $x{:}[\sigma_i]_{i\in I}; \Delta_0 + _{j\in J} \Delta^j_0 \vdash_{\mathcal{W}} vr{:}\tau$ by the rule $(\to \mathtt{e})$. The result holds by taking $\Gamma_0 = \Delta_0 + _{j\in J} \Delta^j_0$, since $(\Delta_0 + _{j\in J} \Delta^j_0) + _{i\in I_v} \Delta_i + _{j\in J} (+ _{i\in I_j} \Delta^j_i) = (\Delta_0 + _{i\in I_v} \Delta_i) + _{j\in J} (\Delta^j_0 + _{i\in I_j} \Delta^j_i) = \Delta + _{j\in J} \Delta_j = \Gamma$.
- $(\to \mathtt{e}_g)$ Similar to the previous case.

■

4 Weighted subject reduction and subject expansion

This section presents two key properties of the typing systems which are used all along this work. The first one is called weighted subject reduction (WSR), a quantitative version of the subject reduction property which holds in the presence of non-idempotent types. Indeed, types and type assignments in system $\mathcal{W}$ are preserved for any kind of reduction step, but this is only true for *non-erasing* reduction steps in system $\mathcal{S}$. Moreover, the size of derivations decreases for *typed occurrences* of redexes. WSR provides direct proofs of normalization for different reduction relations, as will be explained in the next sections. The second property is subject expansion, which is crucial to guarantee that weakly normalizing terms are typable.

To have appropriate statements for the WSR property, we first introduce the following notion, presented for both systems $\mathcal{S}$ and $\mathcal{W}$. Let $\Phi \triangleright \Gamma \vdash_X t : \tau$, where $X \in \{\mathcal{W}, \mathcal{S}\}$. Then $w \in \mathtt{o}(t)$ is a **typed occurrence** of Φ in system X, written $w \in \mathtt{to}(\Phi)$, if either $w = \epsilon$ or $w = iw'$, $(i = 0, 1)$, and $w' \in \mathtt{o}(t|_i)$ is a typed occurrence of some of their corresponding subderivations in Φ. More precisely,

- If Φ is of the form

$$\frac{}{x : [\tau] \vdash x : \tau}\,(\mathtt{ax}).$$

  Then $\epsilon \in \mathtt{to}(\Phi)$.
- If Φ is of the form

$$\frac{\Phi'}{\Gamma \vdash \lambda x.t : \mathcal{M} \to \tau}\,(\to \mathtt{i}).$$

  Then $\epsilon \in \mathtt{to}(\Phi)$ and $0w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi')$.
- If Φ is of the form

$$\frac{\Phi_t := \dfrac{\vdots}{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau} \qquad \left(\Phi_u^i := \dfrac{\vdots}{\Delta_i \vdash u : \sigma_i}\right)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash tu : \tau}\,(\to \mathtt{e}).$$

  Then $\epsilon \in \mathtt{to}(\Phi)$, $0w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi_t)$, and $1w' \in \mathtt{to}(\Phi)$ iff $I \neq \emptyset$ and $w' \in \mathtt{to}(\Phi_u^i)$, for some $i \in I$.
- If Φ is of the form

$$\frac{\Phi_t := \dfrac{\vdots}{\Gamma \vdash t : [] \to \tau} \qquad \Phi_u := \dfrac{\vdots}{\Delta \vdash u : \sigma}}{\Gamma + \Delta \vdash tu : \tau}\,(\to \mathtt{e}_1).$$

  Then $\epsilon \in \mathtt{to}(\Phi)$, $0w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi_t)$, and $1w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi_u)$.
- If Φ is of the form

$$\frac{I \neq \emptyset \qquad \Phi_t := \dfrac{\vdots}{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau} \qquad \left(\Phi_u^i := \dfrac{\vdots}{\Delta_i \vdash u : \sigma_i}\right)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash tu : \tau}\,(\to \mathtt{e}_2).$$

Then $\epsilon \in \mathtt{to}(\Phi)$, $0w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi_t)$, and $1w' \in \mathtt{to}(\Phi)$ iff $w' \in \mathtt{to}(\Phi^i_u)$, for some $i \in I$.

Given $\Phi \rhd \Gamma \vdash_{\mathcal{S}} t:\tau$, observe that every $w \in \mathtt{o}(t)$ is a typed occurrence of Φ, while this is not the case for typing derivations in system $\mathcal{W}$, as the following derivations Φ_1 and Φ_2 show.

$$\Phi_1 := \dfrac{\dfrac{}{x:[[\tau,\tau]\to\tau]\vdash x:[\tau,\tau]\to\tau} \quad \dfrac{\dfrac{}{y:[[]\to\tau]\vdash y:[]\to\tau} \quad \dfrac{}{z:[\tau]\vdash z:\tau}}{y:[[]\to\tau]\vdash yz:\tau}}{x:[[\tau,\tau]\to\tau],y:[[]\to\tau,[\tau]\to\tau],z:[\tau]\vdash x(yz):\tau}$$

$$\Phi_2 := \dfrac{\dfrac{}{x:[[\tau,\tau]\to\tau]\vdash x:[\tau,\tau]\to\tau} \quad \dfrac{\dfrac{}{y:[[]\to\tau]\vdash y:[]\to\tau} \quad \dfrac{}{y:[[]\to\tau]\vdash y:[]\to\tau}}{y:[[]\to\tau]\vdash yz:\tau}}{x:[[\tau,\tau]\to\tau],y:[[]\to\tau,[]\to\tau]\vdash x(yz):\tau}.$$

Indeed, the occurrencies ϵ, 0, 1 and 10 are typed occurrences of both Φ_1 and Φ_2, while $11 \in \mathtt{to}(\Phi_1)$ but $11 \notin \mathtt{to}(\Phi_2)$.

THEOREM 4.1 (WSR)
Let $\Phi_t \rhd \Gamma \vdash_X t:\tau$, where $X \in \{\mathcal{W},\mathcal{S}\}$.

1. If $X = \mathcal{W}$ and $t \xrightarrow{w}_\beta t'$ and $w \notin \mathtt{to}(\Phi_t)$, then $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{W}} t':\tau$ and $\mathtt{sz}(\Phi_t) = \mathtt{sz}(\Phi_{t'})$.
2. If $X = \mathcal{W}$ and $t \xrightarrow{w}_\beta t'$ and $w \in \mathtt{to}(\Phi_t)$, then $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{W}} t':\tau$ and $\mathtt{sz}(\Phi_t) > \mathtt{sz}(\Phi_{t'})$.
3. If $X = \mathcal{S}$ and $t \xrightarrow{w}_\beta t'$ is non-erasing, then $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{S}} t':\tau$ and $\mathtt{sz}(\Phi_t) > \mathtt{sz}(\Phi_{t'})$.

Let $\Phi \rhd \Gamma \vdash_X t:\tau$. We proceed by induction on $w \in \mathtt{o}(t)$.

- If $w = \epsilon$ then $t = (\lambda x.u)v \to_\beta u\{x/v\} = t'$.
  Let $X = \mathcal{W}$. By construction, $\Gamma = \Delta +_{i\in I} \Delta_i$ and

$$\Phi_t := \dfrac{\dfrac{\Phi_u := \dfrac{\vdots}{x:[\sigma_i]_{i\in I};\Delta\vdash u:\tau}}{\Delta\vdash\lambda x.u:[\sigma_i]_{i\in I}\to\tau} \quad \left(\Phi^i_v := \dfrac{\vdots}{\Delta_i\vdash v:\sigma_i}\right)_{i\in I}}{\Delta +_{i\in I}\Delta_i\vdash(\lambda x.u)v:\tau}.$$

  By application of Lemma 3.2 to Φ_u and $(\Phi^i_v)_{i\in I}$ we obtain $\Phi_{t'} \rhd \Delta +_{i\in I} \Delta_i \vdash_X t':\tau$, where $\mathtt{sz}(\Phi_{t'}) = \mathtt{sz}(\Phi_u) +_{i\in I} \mathtt{sz}(\Phi^i_v) - |I| < \mathtt{sz}(\Phi_u) +_{i\in I} \mathtt{sz}(\Phi^i_v) + 1 = \mathtt{sz}(\Phi)$.
  A similar reasoning allows to conclude if $X = \mathcal{S}$.
- If $t = \lambda x.u \xrightarrow{w}_\beta \lambda x.u' = t'$ where $u \xrightarrow{w'}_\beta u'$, then $w = 0w'$. By construction one has that $\Phi_u \rhd x:\mathcal{M};\Gamma \vdash_X u:\sigma$ where $\tau = \mathcal{M}\to\sigma$ and $\mathtt{sz}(\Phi_t) = \mathtt{sz}(\Phi_u)+1$. Note that $0w' \in \mathtt{to}(\Phi_t)$ iff $w' \in \mathtt{to}(\Phi_u)$.
  By the *i.h.* $\Phi_{u'} \rhd x:\mathcal{M};\Gamma \vdash_X u':\sigma$. We consider the different cases.
  If $X = \mathcal{W}$ and $w' \notin \mathtt{to}(\Phi_u)$, then the *i.h.* gives $\mathtt{sz}(\Phi_{u'}) = \mathtt{sz}(\Phi_u)$, thus by the rule $(\to \mathtt{i})$ one has $\Phi_{\lambda x.u'} \rhd \Gamma \vdash_{\mathcal{W}} \lambda x.u':\tau$ where $\mathtt{sz}(\Phi_{t'}) = \mathtt{sz}(\Phi_{u'}) + 1 = \mathtt{sz}(\Phi_t)$.
  If $X = \mathcal{W}$ and $w' \in \mathtt{to}(\Phi_u)$, or $X = \mathcal{S}$, then the *i.h.* gives $\mathtt{sz}(\Phi_{u'}) < \mathtt{sz}(\Phi_u)$, thus by the rule $(\to \mathtt{i})$ one has $\Phi_{t'} \rhd \Gamma \vdash_X \lambda x.u':\tau$ where $\mathtt{sz}(\Phi_{t'}) = \mathtt{sz}(\Phi_{u'}) + 1 < \mathtt{sz}(\Phi_u) + 1 = \mathtt{sz}(\Phi_t)$.
- If $t = ur \xrightarrow{w}_\beta u'r = t'$ where $u \xrightarrow{w'}_\beta u'$, then $w = 0w'$.

Let $X=\mathcal{W}$. By construction one has that $\Gamma=\Gamma_0+_{i\in I}\Gamma_i$ and $(\Phi^i_r\triangleright\Gamma_i\vdash_{\mathcal{W}} r:\rho_i)_{i\in I}$ and $\Phi_u\triangleright\Gamma_0\vdash_{\mathcal{W}} u:[\rho_i]_{i\in I}\to\tau$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1$. Note that $0w'\in\mathtt{to}(\Phi_t)$ iff $w'\in\mathtt{to}(\Phi_u)$.
By the *i.h.* $\Phi'_u\triangleright\Gamma_0\vdash_{\mathcal{W}} u':[\rho_i]_{i\in I}\to\tau$. Therefore,

$$\Phi_{t'}:=\dfrac{\Phi_{u'}:=\dfrac{\vdots}{\Gamma_0\vdash u':[\rho_i]_{i\in I}\to\tau}\qquad\left(\Phi^i_r:=\dfrac{\vdots}{\Gamma_i\vdash r:\rho_i}\right)_{i\in I}}{\Gamma\vdash u'r:\tau}\ (\to \mathsf{e}).$$

If $w'\notin\mathtt{to}(\Phi_u)$, then the *i.h.* also gives $\mathtt{sz}(\Phi_{u'})=\mathtt{sz}(\Phi_u)$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_{u'})+_{i\in I}\mathtt{sz}(\Phi^i_r)+1=\mathtt{sz}(\Phi)$.
If $w'\in\mathtt{to}(\Phi_u)$, then the *i.h.* also gives $\mathtt{sz}(\Phi_{u'})<\mathtt{sz}(\Phi_u)$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_{u'})+_{i\in I}\mathtt{sz}(\Phi^i_r)+1<\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1=\mathtt{sz}(\Phi)$.
Let $X=\mathcal{S}$. We consider two cases depending on the last typing rule used in the derivation Φ:

– $(\to \mathsf{e}_1)$: by construction $\Gamma=\Gamma_0+\Gamma_1$ and $\Phi_u\triangleright\Gamma_0\vdash_{\mathcal{S}} u:[]\to\tau$ and $\Phi_r\triangleright\Gamma_1\vdash_{\mathcal{S}} r:\rho$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+\mathtt{sz}(\Phi_r)+1$. By the *i.h.* $\Phi'_u\triangleright\Gamma_0\vdash_{\mathcal{S}} u':[]\to\tau$. Therefore,

$$\Phi_{t'}:=\dfrac{\Phi_{u'}:=\dfrac{\vdots}{\Gamma_0\vdash u':[]\to\tau}\qquad\Phi_r:=\dfrac{\vdots}{\Gamma_1\vdash r:\rho}}{\Gamma\vdash u'r:\tau}\ (\to \mathsf{e}_1).$$

The *i.h.* also gives $\mathtt{sz}(\Phi_{u'})<\mathtt{sz}(\Phi_u)$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_{u'})+\mathtt{sz}(\Phi_r)+1<\mathtt{sz}(\Phi_u)+\mathtt{sz}(\Phi_r)+1=\mathtt{sz}(\Phi)$.

– $(\to \mathsf{e}_2)$: by construction $\Gamma=\Gamma_0+_{i\in I}\Gamma_i$ and $\Phi_u\triangleright\Gamma_0\vdash_{\mathcal{S}} u:[\rho_i]_{i\in I}\to\tau$ and $(\Phi^i_r\triangleright\Gamma_i\vdash_{\mathcal{S}} r:\rho_i)_{i\in I}$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1$. By the *i.h.* $\Phi'_u\triangleright\Gamma_0\vdash_{\mathcal{S}} u':[\rho_i]_{i\in I}\to\tau$. Therefore,

$$\Phi_{t'}:=\dfrac{\Phi_{u'}:=\dfrac{\vdots}{\Gamma_0\vdash u':[\rho_i]_{i\in I}\to\tau}\qquad\left(\Phi^i_r:=\dfrac{\vdots}{\Gamma_i\vdash r:\rho_i}\right)_{i\in I}}{\Gamma\vdash u'r:\tau}\ (\to \mathsf{e}).$$

The *i.h.* also gives $\mathtt{sz}(\Phi_{u'})<\mathtt{sz}(\Phi_u)$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_{u'})+_{i\in I}\mathtt{sz}(\Phi^i_r)+1<\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1=\mathtt{sz}(\Phi)$.

- If $ur\xrightarrow{w}_\beta ur'$ where $r\xrightarrow{w'}_\beta r'$, then $w=1w'$.
Let $X=\mathcal{W}$. By construction one has that $\Gamma=\Gamma_0+_{i\in I}\Gamma_i$ and $(\Phi^i_r\triangleright\Gamma_i\vdash_{\mathcal{W}} r:\rho_i)_{i\in I}$ and $\Phi_u\triangleright\Gamma_0\vdash_{\mathcal{W}} u:[\rho_i]_{i\in I}\to\tau$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1$. By the *i.h.* $\Phi^i_{r'}\triangleright\Gamma_i\vdash_{\mathcal{W}} r':\rho_i$. Then,

$$\Phi_{t'}:=\dfrac{\Phi_u\triangleright\dfrac{\vdots}{\Gamma_0\vdash u:[\rho_i]_{i\in I}\to\tau}\qquad\left(\Phi^i_{r'}:=\dfrac{\vdots}{\Gamma_i\vdash r':\rho_i}\right)_{i\in I}}{\Gamma\vdash ur':\tau}\ (\to \mathsf{e}).$$

We consider the two cases.
If $1w' \notin \mathtt{to}(\Phi_t)$ then either $I=\emptyset$ or $(w' \notin \mathtt{to}(\Phi^i_r))_{i\in I}$. If $I=\emptyset$ then r is simply replaced by r' in $\Phi_{t'}$ and $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_u)+1=\mathtt{sz}(\Phi_t)$. Otherwise, by the *i.h.* we have $(\mathtt{sz}(\Phi^i_{r'})=\mathtt{sz}(\Phi^i_r))_{i\in I}$ so that $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_{r'})+1=\mathtt{sz}(\Phi)$.
If $1w' \in \mathtt{to}(\Phi_t)$ then $w' \in \mathtt{to}(\Phi^i_r)$ for some $i\in I$. Let $\emptyset \neq K \subseteq I$ be the set of all i such that $w' \in \mathtt{to}(\Phi^i_r)$ and let $\bar{K}=I\setminus K$ be the set of all i such that $w' \notin \mathtt{to}(\Phi^i_r)$. The *i.h.* gives $(\mathtt{sz}(\Phi^i_{r'})<\mathtt{sz}(\Phi^i_r))_{i\in K}$ and $(\mathtt{sz}(\Phi^i_{r'})=\mathtt{sz}(\Phi^i_r))_{i\in\bar{K}}$. Then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_{r'})+1=\mathtt{sz}(\Phi_u)+_{i\in K}\mathtt{sz}(\Phi^i_{r'})+_{i\in\bar{K}}\mathtt{sz}(\Phi^i_{r'})+1<\mathtt{sz}(\Phi_u)+_{i\in K}\mathtt{sz}(\Phi^i_r)+_{i\in\bar{K}}\mathtt{sz}(\Phi^i_{r'})+1=\mathtt{sz}(\Phi_u)+_{i\in K}\mathtt{sz}(\Phi^i_r)+_{i\in\bar{K}}\mathtt{sz}(\Phi^i_r)+1=\mathtt{sz}(\Phi)$.

Let $X=\mathcal{S}$. If $ur \xrightarrow{w}_\beta ur'$ where $r \xrightarrow{w'}_\beta r'$, then $w=1w'$. We consider two cases depending on the last typing rule used in the derivation Φ:

– $(\to \mathtt{e}_1)$: by construction $\Gamma=\Gamma_0+\Gamma_1$ and $\Phi_u \rhd \Gamma_0 \vdash_{\mathcal{S}} u:[]\to\tau$ and $\Phi_r \rhd \Gamma_1 \vdash_{\mathcal{S}} r:\rho$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+\mathtt{sz}(\Phi_r)+1$. By the *i.h.* $\Phi_{r'} \rhd \Gamma \vdash_{\mathcal{S}} r':\rho$. Then,

$$\Phi_{t'} := \dfrac{\Phi_u \rhd \dfrac{\vdots}{\Gamma_0 \vdash u:[]\to\tau} \qquad \Phi_{r'} := \dfrac{\vdots}{\Gamma_1 \vdash r':\rho}}{\Gamma \vdash ur':\tau}\ (\to \mathtt{e}_1).$$

The *i.h.* also gives $\mathtt{sz}(\Phi_{r'})<\mathtt{sz}(\Phi_r)$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_u)+\mathtt{sz}(\Phi_{r'})+1<\mathtt{sz}(\Phi_u)+\mathtt{sz}(\Phi_r)+1=\mathtt{sz}(\Phi)$.

– $(\to \mathtt{e}_2)$: By construction $\Gamma=\Gamma_0+_{i\in I}\Gamma_i$ and $\Phi_u \rhd \Gamma_0 \vdash_{\mathcal{S}} u:[\rho_i]_{i\in I}\to\tau$ and $(\Phi^i_r \rhd \Gamma_i \vdash_{\mathcal{S}} r:\rho_i)_{i\in I}$ where $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1$. By the *i.h.* $\Phi^i_{r'} \rhd \Gamma_i \vdash_{\mathcal{S}} r':\rho_i$. Then,

$$\Phi_{t'} := \dfrac{\Phi_u \rhd \dfrac{\vdots}{\Gamma_0 \vdash u:[\rho_i]_{i\in I}\to\tau} \qquad \left(\Phi^i_{r'} := \dfrac{\vdots}{\Gamma_i \vdash r':\rho_i}\right)_{i\in I}}{\Gamma \vdash ur':\tau}\ (\to \mathtt{e}).$$

The *i.h.* gives $(\mathtt{sz}(\Phi^i_{r'})<\mathtt{sz}(\Phi^i_r))_{i\in I}$, then $\mathtt{sz}(\Phi_{t'})=\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_{r'})+1<\mathtt{sz}(\Phi_u)+_{i\in I}\mathtt{sz}(\Phi^i_r)+1=\mathtt{sz}(\Phi)$.

■

COROLLARY 4.2
If t is $\mathcal{W}$-typable then any sequence reducing all and only β-redexes in typed occurrences terminates.

THEOREM 4.3 (Subject Expansion)
Let $\Phi_{t'} \rhd \Gamma \vdash_X t':\tau$, where $X\in\{\mathcal{W},\mathcal{S}\}$.

(1) If $X=\mathcal{W}$ and $t\to_\beta t'$, then $\Phi_t \rhd \Gamma \vdash_{\mathcal{W}} t:\tau$.
(2) If $X=\mathcal{S}$ and $t\to_\beta t'$ is non-erasing, then $\Phi_t \rhd \Gamma \vdash_{\mathcal{S}} t:\tau$.

By induction on $t\to_\beta t'$.

- If $t=(\lambda x.u)v \to_\beta u\{x/v\}=t'$, then by Lemma 3.3 one has $\Gamma=\Gamma_0+_{i\in I}\Gamma_i$ and $x:[\sigma]_{i\in I};\Gamma_0 \vdash_X u:\tau$ and $(\Gamma_i \vdash_X v:\sigma_i)_{i\in I}$. Hence, $\Gamma_0 \vdash_X \lambda x.u:[\sigma_i]_{i\in I}\to\tau$ by the rule $(\to \mathtt{i})$.
If $X=\mathcal{W}$, then we conclude by constructing $\Gamma \vdash_{\mathcal{W}} (\lambda x.u)v:\tau$ by the rule $(\to \mathtt{e})$.

If $X = \mathcal{S}$, then there are two cases. If $x \notin \mathtt{fv}(u)$ the reduction step is erasing, which leads to a contradiction. If $x \in \mathtt{fv}(u)$, then Lemma 3.1-2 guarantees $I \neq \emptyset$ and we can build $\Gamma \vdash_{\mathcal{S}} (\lambda x.u)v{:}\tau$ by the rule $(\to \mathtt{e}_2)$.

- Let $t = \lambda x.u \to_\beta \lambda x.u' = t'$ where $u \to_\beta u'$, then one has that $x{:}\mathcal{M}; \Gamma \vdash_X u'{:}\sigma$ where $\tau = \mathcal{M} \to \sigma$. By the *i.h.* $x{:}\mathcal{M}; \Gamma \vdash_X u{:}\sigma$. Hence, by the rule $(\to \mathtt{i})$ one gets $\Gamma \vdash_X \lambda x.u{:}\tau$.
- $X = \mathcal{W}$. Let $t = ur \to_\beta u'r = t'$ where $u \to_\beta u'$, then one has that $\Gamma = \Gamma_0 +_{i\in I} \Gamma_i$ and $\Gamma_0 \vdash_{\mathcal{W}} u'{:}[\rho_i]_{i\in I} \to \tau$ and $(\Gamma_i \vdash_{\mathcal{W}} r{:}\rho_i)_{i\in I}$. By the *i.h.* $\Gamma_0 \vdash_{\mathcal{W}} u{:}[\rho_i]_{i\in I} \to \tau$ thus, by the rule $(\to \mathtt{e})$, $\Gamma \vdash_{\mathcal{W}} ur{:}\tau$.
- $X = \mathcal{W}$. Let $t = ur \to_\beta ur' = t'$ where $r \to_\beta r'$, then one has that $\Gamma = \Gamma_0 +_{i\in I} \Gamma_i$ and $\Gamma_0 \vdash_{\mathcal{W}} u{:}[\rho_i]_{i\in I} \to \tau$ and $(\Gamma_i \vdash_{\mathcal{W}} r'{:}\rho_i)_{i\in I}$. By the *i.h.* $(\Gamma_i \vdash_{\mathcal{W}} r{:}\rho_i)_{i\in I}$. Hence, by the rule $(\to \mathtt{e})$, $\Gamma \vdash_{\mathcal{W}} ur{:}\tau$.
  Note that if $I = \emptyset$ then the *i.h.* is not necessary to obtain the result.
- The cases $X = \mathcal{S}$ are similar.

■

It is worth noticing that Theorem 4.1 does not necessarily hold in system $\mathcal{S}$ for *erasing steps*: we have $\Phi \rhd z{:}[\tau] \vdash_{\mathcal{S}} \lambda x.(\lambda y.z)x{:}[\sigma] \to \tau$ and $\lambda x.(\lambda y.z)x \to_\beta \lambda x.z$ but it is not difficult to see that there is no type derivation in $\mathcal{S}$ ending in $z{:}[\tau] \vdash \lambda x.z{:}[\sigma] \to \tau$. The same happens with Theorem 4.3: we have $\Phi \rhd z{:}[\tau] \vdash_{\mathcal{S}} \lambda x.z{:}[] \to \tau$ and $\lambda x.(\lambda y.z)x \to_\beta \lambda x.z$ but there is no type derivation in $\mathcal{S}$ ending in $z{:}[\tau] \vdash \lambda x.(\lambda y.z)x{:}[] \to \tau$.

5 Head normalization

The type system $\mathcal{W}$ characterizes head normalization: a λ-term is head normalizing if and only if it is typable in system $\mathcal{W}$. This result may be proved either by using the reducibility method [36, 46, 65], as in [24], or more directly by a combinatorial argument, as in [32]. For the sake of completeness, we provide an overview of both proofs here, in Sections 5.1 and 5.2 respectively. Concerning the other normalization properties considered in this article, only the combinatorial argument will be presented.

5.1 Reducibility argument

In a nutshell, normalization proofs by reducibility go as follows: to prove that typable terms normalize, one has to

- Associate with each type σ a set $[[\sigma]]$ of pure λ-terms, the *realisers* of σ. All the realisers have to normalize.
- Prove that if $\Gamma \vdash t{:}\sigma$ is derivable, then $t \in [[\sigma]]$.

DEFINITION 5.1
Letting $\mathcal{HN}(\beta)$ be the set of head β-normalizable λ-terms, we define $[[\sigma]], [[\mathcal{M}]] \subseteq \Lambda$, by mutual induction on strict and multiset types:

- $[[\alpha]] = \mathcal{HN}(\beta)$ if $\alpha \in \mathcal{A}$
- $[[[\sigma_i]_{i\in I}]] = \begin{cases} \Lambda & \text{if } I = \emptyset \\ \bigcap_{i\in I} [[\sigma_i]] & \text{otherwise} \end{cases}$
- $[[\mathcal{M} \to \sigma]] = [[\mathcal{M}]] \to [[\sigma]]$

where $X \to Y = \{t \in \Lambda \mid \forall u \in X\ tu \in Y\}$, for $X, Y \subseteq \Lambda$.

Observe that, in the definition above, the realisers of a multiset $\mathcal{M}$ are the same as those of its underlying set.

The key property of realisers is that they are all head β-normalizable:

LEMMA 5.2
Let $\mathcal{HN}_0(\beta)=\{xt_1\dots t_n\in\Lambda \mid x \text{ is a variable, } n\geq 0,\ t_1,\dots,t_n\in\Lambda\}$. For all type σ, $\mathcal{HN}_0(\beta)\subseteq[[\sigma]]\subseteq\mathcal{HN}(\beta)$.

Notice in particular that the variables realise all types. The basic lemma is then the following:

LEMMA 5.3
If $\Phi\rhd\Gamma\vdash_{\mathcal{W}} t:\sigma$, where $\Gamma=x_1:\mathcal{M}_1,\dots,x_k:\mathcal{M}_k$, and if $t_1\in[[\mathcal{M}]]_1,\dots,t_k\in[[\mathcal{M}]]_k$ then $t\{x_1/t_1,\dots,x_k/t_k\}\in[[\sigma]]$.

The *only if* part of the theorem below follows from Lemma 5.3 and Lemma 5.2. The *if* part is easier, see Theorem 5.6.

THEOREM 5.4
Let $t\in\Lambda$. Then t is $\mathcal{W}$-typable if and only if $t\in\mathcal{HN}(\beta)$.

5.2 Combinatorial argument

We present here an alternative proof of Theorem 5.4, which is purely combinatorial and exploits the WSR Property (Theorem 4.1).

LEMMA 5.5
If $t\in\mathcal{HNF}(\beta)$, then t is $\mathcal{W}$-typable.

Let $t=\lambda x_1\cdots x_n.xt_1\cdots t_m$ $(n,m\geq 0)$ and $\tau=\mathcal{M}_1\to\cdots\to\mathcal{M}_m\to\alpha$, where $(\mathcal{M}_i=[])_{1\leq i\leq m}$. By the rule (ax), we get $x:[\tau]\vdash_{\mathcal{W}} x:\tau$ and by m applications of the rule $(\to e)$, we get $x:[\tau]\vdash_{\mathcal{W}} xt_1\cdots t_m:\alpha$. Therefore, by n applications of the rule $(\to \mathtt{i})$, we get $\Gamma_n\vdash_{\mathcal{W}} t:\Gamma_{n-1}(x_1)\to\cdots\to\Gamma_0(x_n)\to\alpha$, where $\Gamma_0=x:[\tau]$ and $(\Gamma_{i+1}=\Gamma_i\setminus\setminus x_{n-i})_{0\leq i\leq n-1}$. ■

THEOREM 5.6
If $t\in\mathcal{HN}(\beta)$, then t is $\mathcal{W}$-typable.

Let t be head β-normalizable. Then $t\to_\beta^n t'$ and $t'\in\mathcal{HNF}(\beta)$. We reason by induction on n.

- If $t=t'\in\mathcal{HNF}(\beta)$ then, by Lemma 5.5, t is $\mathcal{W}$-typable.
- Let $t\to_\beta t''\to_\beta^n t'$, where $t'\in\mathcal{HNF}(\beta)$. By the *i.h.* t'' is $\mathcal{W}$-typable thus, by Theorem 4.3, t is $\mathcal{W}$-typable.

■

LEMMA 5.7
If $\Phi\rhd\Gamma\vdash_{\mathcal{W}} t:\tau$ and t has no typed redex occurrence in Φ, then $t\in\mathcal{HNF}(\beta)$.

The proof is by contradiction.

Let $\Phi\rhd\Gamma\vdash_{\mathcal{W}} t:\tau$, where t has no typed redex occurrence in Φ. Suppose that $t\notin\mathcal{HNF}(\beta)$, so that $t=\lambda x_1\dots x_n.(\lambda x.p)rt_1\dots t_m$ for some $n,m\geq 0$. By construction there are multiset types $\mathcal{M}_1,\dots,\mathcal{M}_n$ and a type σ such that $\tau=\mathcal{M}_1\to\cdots\to\mathcal{M}_n\to\sigma$ and $\Phi'\rhd x_n:\mathcal{M}_n;\dots;x_1:\mathcal{M}_1;\Gamma\vdash_{\mathcal{W}}(\lambda x.p)rt_1\dots t_m:\sigma$. Again by construction we have derivations $(\Phi_j\rhd\Delta_j\vdash_{\mathcal{W}}(\lambda x.p)rt_1\cdots t_{j-1}:\mathcal{N}_j\to\cdots\to\mathcal{N}_m\to\sigma)_{1\leq j\leq m}$ for some multiset types $\mathcal{N}_1,\dots,\mathcal{N}_m$ and some type assignments $\Delta_1\dots\Delta_m$. In particular, $\Phi_1\rhd\Delta_1\vdash_{\mathcal{W}}(\lambda x.p)r:\mathcal{N}_1\to\cdots\to\mathcal{N}_m\to\sigma$ so that the redex $(\lambda x.p)r$ is in a typed occurrence of Φ, which leads to a contradiction. ■

THEOREM 5.8
If t is $\mathcal{W}$-typable, then $t \in \mathcal{HN}(\beta)$.

Let $\Phi \rhd \Gamma \vdash_{\mathcal{W}} t : \tau$. By Corollary 4.2, the strategy of contracting only redexes in typed occurrences yields a finite sequence of β-reductions $t \to_\beta^* t'$, where t' has no redex typed occurrence. By Theorem 4.1 the term t' is $\mathcal{W}$-typable. Hence, by Lemma 5.7 $t' \in \mathcal{HNF}(\beta)$, so that t is head β-normalizable. ■

From Theorem 5.8 and Theorem 5.6 we obtain the following characterization:

COROLLARY 5.9
A term t is $\mathcal{W}$-typable if and only if $t \in \mathcal{HN}(\beta)$.

For the λ-calculus, the head normalizable terms happen to be exactly the *solvable* ones (see below). Hence $\mathcal{W}$-typability characterizes solvable terms, as well. Interestingly, for calculi where the notion of head-normal form is problematic, sensible notions of solvability may be defined, and characterized via non-idempotent intersection types, as in [8].

A λ-term t is said to be **solvable** iff there is a *head-context* C such that $C[t]$ β-reduces to the identity function $\mathtt{I} = \lambda x.x$. It is well known thet a λ-term is solvable if and only if it is head normalizable. Hence, we get the following:

COROLLARY 5.10
A term t is solvable iff t is $\mathcal{W}$-typable iff $t \in \mathcal{HN}(\beta)$.

6 Weak normalization

Weakly β-normalizing terms can also be characterized by means of $\mathcal{W}$-typability. As for the characterization of head normalization, the first proof of this result appeared in [24], and is based on the reducibility method. In this section, we present an alternative proof, which is purely combinatorial and exploits the WSR Property (Theorem 4.1).

The set of **positive** (resp. **negative**) strict types and multiset types of a strict type/multiset type/type assignment/typing is the smallest set satisfying the following conditions (cf.[10]), where $\mathcal{T}$ denotes a strict type or a multiset type:

$$\frac{}{\sigma \in \mathtt{P}(\sigma)} \qquad \frac{}{\mathcal{M} \in \mathtt{P}(\mathcal{M})}$$

$$\frac{\mathcal{T} \in \mathtt{P}(\sigma_i) \qquad I \neq \emptyset}{\mathcal{T} \in \mathtt{P}([\sigma_i]_{i \in I})} \qquad \frac{\mathcal{T} \in \mathtt{N}(\mathcal{M})}{\mathcal{T} \in \mathtt{P}(\mathcal{M} \to \tau)} \qquad \frac{\mathcal{T} \in \mathtt{P}(\tau)}{\mathcal{T} \in \mathtt{P}(\mathcal{M} \to \tau)}$$

$$\frac{\mathcal{T} \in \mathtt{N}(\sigma_i) \qquad I \neq \emptyset}{\mathcal{T} \in \mathtt{N}([\sigma_i]_{i \in I})} \qquad \frac{\mathcal{T} \in \mathtt{P}(\mathcal{M})}{\mathcal{T} \in \mathtt{N}(\mathcal{M} \to \tau)} \qquad \frac{\mathcal{T} \in \mathtt{N}(\tau)}{\mathcal{T} \in \mathtt{N}(\mathcal{M} \to \tau)}$$

$$\frac{y \in \mathtt{dom}(\Gamma) \text{ and } \mathcal{T} \in \mathtt{N}(\Gamma(y))}{\mathcal{T} \in \mathtt{P}(\Gamma)} \qquad \frac{\mathcal{T} \in \mathtt{P}(\Gamma)}{\mathcal{T} \in \mathtt{P}(\langle \Gamma, \tau \rangle)} \qquad \frac{\mathcal{T} \in \mathtt{P}(\tau)}{\mathcal{T} \in \mathtt{P}(\langle \Gamma, \tau \rangle)}$$

$$\frac{y \in \mathtt{dom}(\Gamma) \text{ and } \mathcal{T} \in \mathtt{P}(\Gamma(y))}{\mathcal{T} \in \mathtt{N}(\Gamma)} \qquad \frac{\mathcal{T} \in \mathtt{N}(\Gamma)}{\mathcal{T} \in \mathtt{N}(\langle \Gamma, \tau \rangle)} \qquad \frac{\mathcal{T} \in \mathtt{N}(\tau)}{\mathcal{T} \in \mathtt{N}(\langle \Gamma, \tau \rangle)}$$

EXAMPLE 6.1
By the definition above, $[] \in \mathtt{P}([])$, so that $[] \in \mathtt{N}([] \to \sigma)$, $[] \in \mathtt{N}([[] \to \sigma])$, $[] \in \mathtt{P}(x{:}[[] \to \sigma])$ and $[] \in \mathtt{P}(\langle x{:}[[] \to \sigma], \sigma \rangle)$.

Remark that $[] \notin \mathtt{P}(\Gamma)$ iff $([] \notin \mathtt{N}(\Gamma(x)))_{x \in \mathrm{dom}(\Gamma)}$. Moreover, $[] \notin \mathtt{P}(\langle \Gamma, \sigma \rangle)$ iff $[] \notin \mathtt{P}(\Gamma)$ and $[] \notin \mathtt{P}(\sigma)$. Therefore, $[] \notin \mathtt{P}(\langle \Gamma, \sigma \rangle)$ iff $[] \notin \mathtt{P}(\langle \Gamma \setminus\setminus x, \Gamma(x) \to \sigma \rangle)$, where the operation $\Gamma \setminus\setminus x$ is the one defined in Section 3.

LEMMA 6.2
If $t \in \mathcal{NF}(\beta)$ then $\exists\, \Gamma, \tau$ such that $\Gamma \vdash_{\mathcal{W}} t{:}\tau$ and $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$.

By induction on the structure of β-normal forms.

Let $t = \lambda x_1 \ldots x_m.x t_1 \ldots t_n \in \mathcal{NF}(\beta)$. Then $t_i \in \mathcal{NF}(\beta)$ for all $1 \leq i \leq n$, thus by the *i.h.* one has $\Delta_i \vdash_{\mathcal{W}} t_i{:}\sigma_i$ and $[] \notin \mathtt{P}(\langle \Delta_i, \sigma_i \rangle)$ for all $1 \leq i \leq n$. Let $\tau := [\sigma_1] \to \cdots \to [\sigma_n] \to \alpha$, for some base type α, and let $\Gamma := \{x{:}[\tau]\} + \Delta_1 + \cdots + \Delta_n$. Note that $[] \notin \mathtt{N}(\tau)$ so that $[] \notin \mathtt{P}(\Gamma)$. Then, by the rule $(\mathtt{ax})$ and n applications of $(\to \mathtt{e})$, we obtain $\Gamma \vdash_{\mathcal{W}} x t_1 \ldots t_n{:}\alpha$ where $[] \notin \mathtt{P}(\langle \{x{:}[\tau]\}, \tau \rangle)$[4] and $[] \notin \mathtt{P}(\langle \Gamma, \alpha \rangle)$[5]. Therefore, by m applications of $(\to \mathtt{i})$, we get $\Gamma_0 \vdash_{\mathcal{W}} t{:}\Gamma_1(x_1) \to \cdots \to \Gamma_m(x_m) \to \alpha$ where $\Gamma_m = \Gamma$ and $(\Gamma_{i-1} = \Gamma_i \setminus\setminus x_i)_{1 \leq i \leq m}$.

■

THEOREM 6.3
If $t \in \mathcal{WN}(\beta)$ then $\exists\, \Gamma, \tau$ such that $\Gamma \vdash_{\mathcal{W}} t{:}\tau$ and $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$.

Let $t \in \mathcal{WN}(\beta)$ so that $t \to_\beta^n t'$ and $t' \in \mathcal{NF}(\beta)$. We reason by induction on n.

- If $t \in \mathcal{NF}(\beta)$, then the result holds by Lemma 6.2.
- Let $t \to_\beta t'' \to_\beta^n t'$, $t' \in \mathcal{NF}(\beta)$. By the *i.h.* $\Gamma \vdash_{\mathcal{W}} t''{:}\tau$ and $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$. Therefore, by Theorem 4.3, $\Gamma \vdash_{\mathcal{W}} t{:}\tau$ and so the result holds.

■

LEMMA 6.4
Let $\Phi \triangleright \Gamma \vdash t{:}\tau$. If there is no typed redex occurrence in Φ and $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$ then $t \in \mathcal{NF}(\beta)$.

By induction on Φ.

Let $\Phi \triangleright \Gamma \vdash t{:}\tau$. If t has no typed redex occurrence in Φ then, by Lemma 5.7, $t \in \mathcal{HNF}(\beta)$, thus $t = \lambda x_1 \ldots x_m.x t_1 \ldots t_n$. Therefore $\tau = \mathcal{M}_1 \to \cdots \to \mathcal{M}_m \to \sigma$ and Φ is of the form:

$$\Phi' := \dfrac{\dfrac{\vdots}{x_m{:}\mathcal{M}_m; \ldots; x_1{:}\mathcal{M}_1; \Gamma \vdash x t_1 \ldots t_n{:}\sigma}}{\Gamma \vdash \lambda x_1 \ldots x_m.x t_1 \ldots t_n{:}\mathcal{M}_1 \to \cdots \to \mathcal{M}_m \to \sigma}\ \vdots$$

By hypothesis $[] \notin \mathtt{P}(\tau)$ then $[] \notin \mathtt{P}(\sigma)$ and $[] \notin \mathtt{N}(\mathcal{M}_i)$ for all $1 \leq i \leq m$. Hence, $[] \notin \mathtt{P}(\langle x_m{:}\mathcal{M}_m; \ldots; x_1{:}\mathcal{M}_1; \Gamma, \sigma \rangle)$.

Let $\Phi' = \Phi_0^{n+1}$, $\Gamma_0^{n+1} := x_m{:}\mathcal{M}_m; \ldots; x_1{:}\mathcal{M}_1; \Gamma$ and $\sigma_{n+1} := \sigma$. Then, for each $j \in \{1, \ldots, n\}$ one has $\Gamma_0^{(j+1)} := \Gamma_0^j +_{i \in I_j} \Gamma_i^j$, $[] \notin \mathtt{P}(\Gamma_0^j)$ and $([] \notin \mathtt{P}(\Gamma_i^j))_{i \in I_j}$, where for $\mathcal{N}_j := [\rho_i]_{i \in I_j}$ and $\sigma_j := \mathcal{N}_j \to \sigma_{(j+1)}$ one

[4]The obtained typing if $n = 0$, where $\tau = \alpha$.
[5]The obtained typing if $m = 0$.

has a derivation Φ_0^{j+1} of the form

$$\frac{\Phi_0^j := \dfrac{\vdots}{\Gamma_0^j \vdash x t_1 \cdots t_{(j-1)} : [\rho_i]_{i\in I_j} \to \sigma_{(j+1)}} \qquad \left(\Phi_{t_j}^i := \dfrac{\vdots}{\Gamma_i^j \vdash t_j : \rho_i} \right)_{i\in I_j}}{\Gamma_0^j +_{i\in I_j} \Gamma_i^j \vdash x t_1 \cdots t_j : \sigma_{(j+1)}} (\to \mathtt{e}).$$

In particular $\Phi_0^1 \rhd \Gamma_0^1 \vdash x:\sigma_1$, where $\sigma_1 = \mathcal{N}_1 \to \cdots \to \mathcal{N}_n \to \sigma$, thus $\Gamma_0^1 = \{x : [\mathcal{N}_1 \to \cdots \to \mathcal{N}_n \to \sigma]\}$. Since $[] \notin \mathtt{P}(\Gamma_0^1)$ then for each $j = 1 \ldots n$, $I_j \neq \emptyset$, i.e. $\mathcal{N}_j \neq []$, and $[] \notin \mathtt{P}(\rho_i)$ for any $i \in I_j$.

Therefore, for each $j = 1 \ldots n$, $(\Phi_{t_j}^i \rhd \Gamma_i^j \vdash t_j : \rho_i$ such that $[] \notin \mathtt{P}(\langle \Gamma_i^j, \rho_i \rangle))_{i\in I_j}$. Note also that, any redex in $w_j \in \mathtt{to}(\Phi_{t_j}^i)$, for $i \in I_j$, would also be a redex in $w \in \mathtt{to}(\Phi)$ hence there is no such a redex of t_j in any $\Phi_{t_j}^i$. Therefore, by the *i.h.* $t_j \in \mathcal{NF}(\beta)$ for each $j = 1 \ldots n$, thus $t \in \mathcal{NF}(\beta)$. ■

THEOREM 6.5
If $\Phi \rhd \Gamma \vdash t:\tau$ such that $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$ then $t \in \mathcal{WN}(\beta)$.

By Corollary 4.2, the strategy consisting of contracting all and only redexes in typed occurrences terminates in a term t' without any typed redex occurrences. Moreover, by Theorem 4.1-2 we obtain $\Phi' \rhd \Gamma \vdash t' : \tau$. Since $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$ by hypothesis, then $t' \in \mathcal{NF}(\beta)$ holds by Lemma 6.4. ■

From Theorem 6.5 and Theorem 6.3 we obtain the following characterization:

COROLLARY 6.6
$\Phi \rhd \Gamma \vdash t:\tau$ and $[] \notin \mathtt{P}(\langle \Gamma, \tau \rangle)$ iff $t \in \mathcal{WN}(\beta)$.

7 Weak head normalization

This section presents a characterization of weak head normalization by means of intersection types. A proof of such result using reducibility candidates appears in [26], while here we present the combinatorial proof based on non-idempotent types which is based on system $\mathcal{WH}$, a slight modification of system $\mathcal{W}$.

The set of types of the new system includes one special constants #, which is used to type abstractions. Strict types and multiset types are defined by means of the following grammar:

(strict types)	σ, τ	::=	$\# \mid \alpha \mid \mathcal{M} \to \sigma$
(multiset types)	$\mathcal{M}, \mathcal{N}$	::=	$[\sigma_i]_{i\in I}$, I finite set

The typing rules of the new typing system are given in Figure 3.

The notion of **typed occurrence** is modified according to the definition of the rules of the system $\mathcal{WH}$.

As expected, the system enjoys weighted subject reduction and subject expansion.

$$\frac{}{\emptyset \vdash \lambda x.t : \#}\ (\mathtt{abs}) \qquad \frac{}{x : [\tau] \vdash x : \tau}\ (\mathtt{ax}) \qquad \frac{\Gamma \vdash t:\tau}{\Gamma \setminus\setminus x \vdash \lambda x.t : \Gamma(x) \to \tau}\ (\to \mathtt{i})$$

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

FIG. 3. The intersection type system $\mathcal{WH}$ for the λ-calculus

THEOREM 7.1 (WSR)
Let $\Phi_t \rhd \Gamma \vdash_{\mathcal{WH}} t:\tau$.

(1) If $t \xrightarrow{w}_\beta t'$ and $w \notin \mathtt{to}(\Phi_t)$, then $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{WH}} t':\tau$ and $\mathtt{sz}(\Phi_t) = \mathtt{sz}(\Phi_{t'})$.
(2) If $t \xrightarrow{w}_\beta t'$ and $w \in \mathtt{to}(\Phi_t)$, then $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{WH}} t':\tau$ and $\mathtt{sz}(\Phi_t) > \mathtt{sz}(\Phi_{t'})$.

By induction on $w \in \mathtt{o}(t)$ following the same lines of the proof of Theorem 4.1. ■

COROLLARY 7.2
If t is $\mathcal{WH}$-typable then any sequence reducing all and only β-redexes in typed occurrences terminates.

THEOREM 7.3 (Subject Expansion)
If $\Phi_{t'} \rhd \Gamma \vdash_{\mathcal{WH}} t':\tau$ and $t \to_\beta t'$ then $\Phi_t \rhd \Gamma \vdash_{\mathcal{WH}} t:\tau$.

By induction on $t \to_\beta t'$. ■

Using WSR and subject expansion we characterize the set $\mathcal{WHN}(\beta)$ as follows.

LEMMA 7.4
If $t \in \mathcal{WHNF}(\beta)$, then t is $\mathcal{WH}$-typable.

If t is an abstraction, then $\emptyset \vdash_{\mathcal{WH}} t:\#$. Otherwise, we proceed as in Lemma 5.5. ■

THEOREM 7.5
If $t \in \mathcal{WHN}(\beta)$ then t is $\mathcal{WH}$-typable.

Using Lemma 7.4 and Theorem 7.3. ■

LEMMA 7.6
If $\Phi \rhd \Gamma \vdash_{\mathcal{WH}} t:\tau$ and t has no typed redex occurrence in Φ then $t \in \mathcal{WHNF}(\beta)$.

The proof is by contradiction. Let $\Phi \rhd \Gamma \vdash_{\mathcal{W}} t:\tau$, where t has no typed redex occurrence in Φ. Suppose that $t \notin \mathcal{WHNF}(\beta)$, so that t cannot be an abstraction, it is then necessarily of the form $(\lambda x.p)r t_1 \dots t_m$ for some $m \geq 0$. By construction there are derivations $(\Phi_j \rhd \Delta_j \vdash_{\mathcal{WH}} (\lambda x.p)r t_1 \cdots t_{j-1}:\mathcal{N}_j \to \cdots \to \mathcal{N}_m \to \sigma)_{1 \leq j \leq m}$ for some multiset types $\mathcal{N}_1, \dots, \mathcal{N}_m$ and some type assignments $\Delta_1 \dots \Delta_m$. In particular, $\Phi_1 \rhd \Delta_1 \vdash_{\mathcal{W}} (\lambda x.p)r:\mathcal{N}_1 \to \cdots \to \mathcal{N}_m \to \sigma$ so that the redex $(\lambda x.p)r$ is in a typed occurrence of Φ, which leads to a contradiction. ■

THEOREM 7.7
If t is $\mathcal{WH}$-typable then $t \in \mathcal{WHN}(\beta)$.

Let $\Phi \rhd \Gamma \vdash_{\mathcal{WH}} t:\tau$. By Corollary 7.2, the strategy of contracting only redexes in typed occurrences yields a finite sequence of β-reductions $t \to^*_\beta t'$, where t' has no redex typed occurrence. By Theorem 7.1 the term t' is $\mathcal{WH}$-typable. Hence, by Lemma 7.6 $t' \in \mathcal{WHNF}(\beta)$, so that t is weakly head β-normalizable. ■

From Theorem 7.7 and Theorem 7.5 we obtain the following characterization:

COROLLARY 7.8
A term t is $\mathcal{WH}$-typable if and only if $t \in \mathcal{WHN}(\beta)$.

8 Strong normalization

In this section, we characterize the set of strongly β-normalizing terms by means of $\mathcal{S}$-typability, i.e. we show that a term is $\mathcal{S}$-typable if and only if it is β-strongly normalizing. Our proof does not

use any reducibility/computability argument [36, 46, 65], but is based on the properties presented in Section 4, namely, WSR (Theorem 4.1-3) and Subject Expansion (Theorem 4.3-2) for non-erasing reductions with respect to the system $\mathcal{S}$.

Besides several characterizations of strongly β-normalizing terms via *idempotent* intersection types (a survey can be found for example in [3]), others, based on *non-idempotent* types, appear in the literature. In [5], a subtyping relation is used to get the subject reduction property, but the system types unnecessary instances of arguments, and turns out to be non-relevant, i.e. some sort of weakening is allowed. In [23] the typing rule for term variables is weakened, thus the system is non-relevant, too. The characterization of strong normalization, more precisely the 'strong normalization implies typability' property, is reported as following from an adaptation of the perpetuality proof in [48]. In [44] the characterization is proved by establishing the relation between the non-idempotent system to an idempotent one. In [9] we define a memory calculus K together with a non-idempotent intersection type system $\mathcal{K}$, and we show that a K-term t is typable in systen $\mathcal{K}$ if and only if $t \in \mathcal{SN}(K)$. We then show that β-strong normalization is equivalent to K-strong normalization. We conclude since λ-terms are strictly included in K-terms. In [25], a characterization of strongly normalizing MELL Proof-Nets is given by means of a multiset based relational model, which can also be naturally understood as a non-idempotent IT system characterizing β-strongly normalizing λ-terms.

In this article we propose a much simpler argument which is achieved by replacing the predicate $\mathcal{SN}(\beta)$ (cf. Section 2) by an inductive predicate $\mathcal{ISN}(\beta)$, which is known [72] to be equivalent to the first one. Indeed, the inductive set $\mathcal{ISN}(\beta)$ is the smallest set of terms satisfying the following closure properties:

- If $t_1, \ldots, t_n$ $(n \geq 0) \in \mathcal{ISN}(\beta)$, then $xt_1 \ldots t_n \in \mathcal{ISN}(\beta)$.
- If $t \in \mathcal{ISN}(\beta)$, then $\lambda x.t \in \mathcal{ISN}(\beta)$.
- If $t\{x/u\}t_1 \ldots t_n, u \in \mathcal{ISN}(\beta)$, then $(\lambda x.t)ut_1 \ldots t_n \in \mathcal{ISN}(\beta)$.

This set is indeed equal to $\mathcal{SN}(\beta)$, as expressed by the following Lemma:

LEMMA 8.1 ([72])
$\mathcal{SN}(\beta) = \mathcal{ISN}(\beta)$.

The main result of this section is then simply obtained as follows:

- Prove that typable λ-terms are strongly normalizing (Theorem 8.2), by induction on typing derivations, using the inductive characterization of strong normalization $\mathcal{ISN}(\beta)$, and the fact that weighted subject reduction holds for *non-erasing* β-reductions.
- Prove that strongly normalizing λ-terms are typable (Theorem 8.3), by induction on $\mathcal{ISN}(\beta)$, using the fact that subject expansion holds for *non-erasing* β-reductions.

THEOREM 8.2
If t is $\mathcal{S}$-typable, then $t \in \mathcal{SN}(\beta)$.

Let $\Phi \triangleright \Gamma \vdash_{\mathcal{S}} t : \sigma$. We use Lemma 8.1 to replace in the statement $\mathcal{SN}(\beta)$ by $\mathcal{ISN}(\beta)$. We proceed by induction on $\mathtt{sz}(\Phi)$. When Φ ends with the rule $(\mathtt{ax})$ or $(\to \mathtt{i})$ the proof is straightforward. Let Φ be a derivation ending with $(\to \mathtt{e}_g)$, so that $t = xt_1 \ldots t_n$ or $t = (\lambda x.u)t_1 \ldots t_n$ for some $n \geq 1$.

If $t = xt_1 \ldots t_n$, then for all $(1 \leq i \leq n)$ there is some type derivation Φ_{t_i} for t_i such that $\mathtt{sz}(\Phi_{t_i}) < \mathtt{sz}(\Phi)$. The *i.h.* then gives $t_i \in \mathcal{ISN}(\beta)$ $(1 \leq i \leq n)$ and we thus conclude $xt_1 \ldots t_n \in \mathcal{ISN}(\beta)$.

If $t=(\lambda x.u)t_1\dots t_n$, then $\Gamma=\Gamma_{\lambda x.u}+_{i=1\dots n}(+_{j\in J_i}\Gamma_j)$ and

$$\Phi:=\dfrac{\dfrac{\dfrac{\vdots}{\Phi_u:=\Gamma_u\vdash u:\mathcal{M}_2\to\dots\to\mathcal{M}_n\to\sigma}}{\Gamma_{\lambda x.u}\vdash\lambda x.u:\mathcal{M}_1\to\dots\to\mathcal{M}_n\to\sigma}\qquad(\Phi^j_{t_1})_{j\in J_1}}{\Gamma_{\lambda x.u}+_{i=1\dots n}(+_{j\in J_i}\Gamma_j)\vdash(\lambda x.u)t_1\dots t_n:\sigma}\dfrac{\vdots}{}\,(\Phi^j_{t_n})_{j\in J_n},$$

where $\Phi^j_{t_k}\rhd\Delta^k_j\vdash t_k:\sigma^k_j$ $(j\in J_k)$ and $\mathcal{M}_k=[\sigma^k_i]_{i\in I_k}$ $(1\le k\le n)$, and $(|J_i|=1$ if $I_i=\emptyset$ and $J_i=I_i$ otherwise$)_{i=1\dots n}$. Moreover we have $\mathtt{sz}(\Phi_u)<\mathtt{sz}(\Phi)$ and $(\mathtt{sz}(\Phi^j_{t_i})<\mathtt{sz}(\Phi))_{i=1\dots n}$ so that the *i.h.* gives $u\in\mathcal{ISN}(\beta)$ and $(t_i\in\mathcal{ISN}(\beta))_{i=1\dots n}$. We consider two cases:

- $x\in\mathtt{fv}(u)$. Using Theorem 4.1 one shows that there is $\Phi'\rhd\Gamma\vdash u\{x/t_1\}t_2\dots t_n:\sigma$ such that $\mathtt{sz}(\Phi')<\mathtt{sz}(\Phi)$. Then by the *i.h.* we get $u\{x/t_1\}t_2\dots t_n\in\mathcal{ISN}(\beta)$. This together with $t_1\in\mathcal{ISN}(\beta)$ gives $(\lambda x.u)t_1\dots t_n\in\mathcal{ISN}(\beta)$.
- $x\notin\mathtt{fv}(u)$. Then it is easy to build a type derivation $\Phi'\rhd\Gamma_u+_{i=2\dots n}(+_{j\in J_i}\Gamma_j)\vdash ut_2\dots t_n:\sigma$ such that $\mathtt{sz}(\Phi')<\mathtt{sz}(\Phi)$ so that $ut_2\dots t_n=u\{x/t_1\}t_2\dots t_n\in\mathcal{ISN}(\beta)$ holds by the *i.h.* This, together with $t_1\in\mathcal{ISN}(\beta)$ gives $(\lambda x.u)t_1\dots t_n\in\mathcal{ISN}(\beta)$.

■

THEOREM 8.3
If $t\in\mathcal{SN}(\beta)$, then t is $\mathcal{S}$-typable.

We use Lemma 8.1 to replace in the statement $\mathcal{SN}(\beta)$ by $\mathcal{ISN}(\beta)$. We reason by induction $t\in\mathcal{ISN}(\beta)$. The two first cases are straightforward. Let $t=(\lambda x.u)vt_1\dots t_n$ in $\mathcal{ISN}(\beta)$ coming from $u\{x/v\}t_1\dots t_n, v\in\mathcal{ISN}(\beta)$. By the *i.h.* $u\{x/v\}t_1\dots t_n$ and v are both $\mathcal{S}$-typable. We consider two cases. If $x\in\mathtt{fv}(u)$, then $(\lambda x.u)vt_1\dots t_n$ is $\mathcal{S}$-typable by Theorem 4.3. Otherwise $u\{x/v\}=u$. Let $\Phi\rhd\Gamma\vdash_{\mathcal{S}}ut_1\dots t_n:\tau$ and $\Phi_v\rhd\Delta\vdash_{\mathcal{S}}v:\sigma$. By construction, $\Gamma=\Gamma_u+_{i=1\dots n}(+_{j\in J_i}\Gamma_j)$ and Φ has the following form:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Phi_u:=\Gamma_u\vdash u:\mathcal{M}_1\to\dots\to\mathcal{M}_n\to\tau}\qquad\left(\Phi^j_{t_1}:=\dfrac{\vdots}{\Gamma_j\vdash t_1:\sigma_j}\right)_{j\in J_1}}{\Gamma_u+_{j\in J_1}\Gamma_j\vdash ut_1:\mathcal{M}_2\to\dots\to\mathcal{M}_n\to\tau}}{\dfrac{\vdots}{\Gamma_u+_{i=1\dots n-1}(+_{j\in J_i}\Gamma_j)\vdash ut_1\dots t_{(n-1)}:\mathcal{M}_n\to\tau}\qquad\left(\Phi^j_{t_n}:=\dfrac{\vdots}{\Gamma_j\vdash t_n:\sigma_j}\right)_{j\in J_n}}\,,$$

$$\Gamma_u+_{i=1\dots n}(+_{j\in J_i}\Gamma_j)\vdash ut_1\dots t_n:\tau$$

where $\mathcal{M}_k = [\sigma_i]_{i\in I_k}$ and ($|J_k| = 1$ if $I_k = \emptyset$ and $J_k = I_k$ otherwise $)_{k=1\ldots n}$. Therefore, we can construct a derivation like

$$\Phi_{(\lambda x.u)v} := \dfrac{\dfrac{\Phi_u := \dfrac{\vdots}{\Gamma_u \vdash u : \mathcal{M}_1 \to \cdots \to \mathcal{M}_n \to \tau}}{\Gamma_u \vdash \lambda x.u : [] \to \mathcal{M}_1 \to \cdots \to \mathcal{M}_n \to \tau} \qquad \Phi_v := \dfrac{\vdots}{\Gamma_v \vdash v : \sigma}}{\Gamma_u + \Gamma_v \vdash (\lambda x.u)v : \mathcal{M}_1 \to \cdots \to \mathcal{M}_n \to \tau}$$

and another one like

$$\dfrac{\dfrac{\dfrac{\Phi_{(\lambda x.u)v} \qquad \left(\Phi^j_{t_1} := \dfrac{\vdots}{\Gamma_j \vdash t_1 : \sigma_j}\right)_{j\in J_1}}{\Gamma_u + \Gamma_v + {}_{j\in J_1}\Gamma_j \vdash (\lambda x.u)vt_1 : \mathcal{M}_2 \to \cdots \to \mathcal{M}_n \to \tau}}{\vdots}}{\Gamma_u + \Gamma_v + {}_{i=1\ldots n-1}(+_{j\in J_i}\Gamma_j) \vdash (\lambda x.u)vt_1 \ldots t_{(n-1)} : \mathcal{M}_n \to \tau} \qquad \left(\Phi^j_{t_n} := \dfrac{\vdots}{\Gamma_j \vdash t_n : \sigma_j}\right)_{j\in J_n}}{\Gamma_u + \Gamma_v + {}_{i=1\ldots n}(+_{j\in J_i}\Gamma_j) \vdash (\lambda x.u)vt_1 \ldots t_n : \tau}.$$

We thus conclude that the term $(\lambda x.u)vt_1 \ldots t_n$ is $\mathcal{S}$-typable.

■

COROLLARY 8.4
A term t is $\mathcal{S}$-typable iff $t \in \mathcal{SN}(\beta)$.

9 Exact bounds

This section surveys a major result in [5], where a typing system with non-idempotent IT is used to give exact bounds of the longest β-reduction sequences of *strongly* β-normalizing terms. The presentation of (non-idempotent) IT we adopt in this section uses multisets, as in the previous sections, in contrast to the development in [5], which is based on the introduction of associative and commutative axioms for intersection types denoted by the symbol $\cap$.

The set of types of the new system includes two special constants $\triangledown$ and #, which are used to type, the application and abstraction constructors that remain in the β-normal form of terms, respectively. Strict types and multiset types are defined by means of the following grammar:

(output types)	o	::=	$\triangledown \mid \#$
(strict types)	σ, τ	::=	$o \mid \mathcal{M} \to \sigma$
(multiset types)	$\mathcal{M}, \mathcal{N}$	::=	$[\sigma_i]_{i\in I}$, I finite set

An **input multiset type**, denoted by $\mathcal{I}$, is a (finite) multiset type containing only the output type $\triangledown$. An **input type assignment** is a type assignment Γ such that for all $x \in \mathtt{dom}(\Gamma)$, $\Gamma(x)$ is an input multiset type. By writing $\Gamma_{\mathcal{I}}$ we mean that Γ is an input type assignment. A **type judgment** has the form $\Gamma \vdash^n t : \tau$, where $n \geq 0$, t is a term, Γ is a type assignment and τ is a strict type. The typing rules of the new typing system are given in Figure 4.

Despite of the presence of three typing rules for the abstraction and two for the application, the system is syntax directed. Indeed, the input type # is used for abstraction constructors that remain

$$\frac{}{x : [\sigma] \vdash^0 x{:}\sigma}\ (\mathtt{v})$$

$$\frac{x : \mathcal{M}; \Gamma \vdash^n t{:}\sigma \qquad \mathcal{M} \neq [\,]}{\Gamma \vdash^n \lambda x.t{:}\mathcal{M} \to \sigma}\ (\lambda 1) \qquad \frac{\Gamma \vdash^n t{:}\sigma \qquad x \notin \mathtt{dom}(\Gamma)}{\Gamma \vdash^n \lambda x.t{:}[o] \to \sigma}\ (\lambda 2)$$

$$\frac{x : \mathcal{I}; \Gamma \vdash^n t{:}o}{\Gamma \vdash^n \lambda x.t{:}\#}\ (\lambda 3) \qquad \frac{\Gamma \vdash^n t{:}\triangledown \qquad \Delta \vdash^m u{:}o}{\Gamma + \Delta \vdash^{n+m} tu{:}\triangledown}\ (\mathtt{a1})$$

$$\frac{\Gamma \vdash^n t{:}[\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash^{m_i} u{:}\sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash^{n+_{i\in I} m_i + 1} tu{:}\tau}\ (\mathtt{a2})$$

FIG. 4. A typing system which characterizes longest β-reduction sequences

in the normal form of terms, they are typed with rule (λ3); the arrow type, on the other hand, is used for the remaining abstractions, typed with rules (λ1) and (λ2). In particular, rule (λ1) types non-erasing abstractions ($\mathcal{M} \neq [\,] \Leftrightarrow x \in \mathtt{dom}(\Gamma) \Leftrightarrow x \in \mathtt{fv}(t)$) while rule ($\lambda$2) types the erasing ones ($x \notin \mathtt{dom}(\Gamma) \Leftrightarrow x \notin \mathtt{fv}(t)$). The input type $\triangledown$ is used to type application constructors –by means of rule (a1)– that remain in the normal form of terms; while the other applications are typed with rule (a2). An **optimal** typing is a typing of the form $\Gamma_{\mathcal{I}} \vdash^n t{:}o$, i.e. one that uses an input type assignment and gives an output type. We write $\Gamma \vdash^n_{opt} t{:}\sigma$ iff $\Gamma \vdash^n t{:}\sigma$ is an optimal typing. Here is an example of optimal typing:

$$\dfrac{\Phi \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x{:}[\triangledown]\vdash^0 x{:}\triangledown}\,(\mathtt{v})}{x{:}[\triangledown,\triangledown]\vdash^1 x(\mathtt{I}x){:}\triangledown}\,(\mathtt{a1})}{\vdash^1 \lambda x.x(\mathtt{I}x){:}\#}\,(\lambda 3)}{\vdash^2 \mathtt{I}(\lambda x.x(\mathtt{I}x)){:}\#}\,(\mathtt{a2}) \qquad \dfrac{\dfrac{\dfrac{}{z{:}[\triangledown]\vdash^0 z{:}\triangledown}\,(\mathtt{v})}{\vdash^0 \mathtt{I}{:}[\triangledown]\to\triangledown}\,(\lambda 1) \qquad \dfrac{}{x{:}[\triangledown]\vdash^0 x{:}\triangledown}\,(\mathtt{v})}{x{:}[\triangledown]\vdash^1 \mathtt{I}x{:}\triangledown}\,(\mathtt{a2})}{}$$

where

$$\Phi := \frac{\dfrac{}{z{:}[\#]\vdash^0 z{:}\#}\,(\mathtt{v})}{\vdash^0 \mathtt{I}{:}[\#]\to\#}\,(\lambda 1)$$

The previous system enjoys relevance, i.e.

LEMMA 9.1
$\Gamma \vdash^n t{:}\sigma$ iff $\mathtt{dom}(\Gamma) = \mathtt{fv}(t)$.

We write $t \in \mathcal{SN}_n(\beta)$ iff $t \in \mathcal{SN}(\beta)$ and n is the length of the longest β-reduction sequence starting at t.

The main result in [5] is a characterization of the set $\mathcal{SN}(n)$ by means of the previous typing system. The proof, omitted, uses a perpetual strategy for the λ-calculus.

THEOREM 9.2 ([5])
$\Gamma \vdash^n_{opt} t{:}\sigma$ iff $t \in \mathcal{SN}_n(\beta)$.

Indeed, coming back to our previous example, we have $\vdash^{2}_{opt} \mathtt{I}(\lambda x.x(\mathtt{I}x)):\#$ and the longest reduction sequence starting at $\mathtt{I}(\lambda x.x(\mathtt{I}x))$ has length 2.

10 Inhabitation

When dealing with the problem of type inhabitation for intersection type systems, the non-idempotent case is peculiar. In this section, we provide an informal account of type inhabitation for systems $\mathcal{W}$ and $\mathcal{S}$.

Given a type system X, a type assignment Γ and a type σ, the problem of deciding whether there exists a term t such that the judgment $\Gamma \vdash t{:}\sigma$ is provable in X, is known in the literature both as *emptiness problem* and as *inhabitation problem* for X. Intersection types were introduced as an extension of the 'basic functionality theory' of Curry and Feys [21]. Types being more expressive, the inhabitation problem, which for simple types corresponds to deciding tautologies of minimal implicative logic, becomes harder for intersection type systems.

The inhabitation problem for the original intersection type system presented in [16] has been proved to be undecidable in [66].

The model theoretic counterpart of inhabitation problems are *definability problems*: given a model $\mathcal{M}$ and one of its elements f, is there a term t such that $[[t]]=f$? In both cases, one looks for a term implementing a given specification. Interestingly, the inhabitation problem for the original intersection type system has been recently proved to be equi-decidable to the definability problem for the simplest, set-theoretic model of the simply typed λ-calculus [60].

The inhabitation problem for non-idempotent intersection types is simpler, as shown in [7]:

THEOREM 10.1
The inhabitation problem for system $\mathcal{W}$ is decidable.

Before discussing this decidability result, let us show that the inhabitation problems for systems $\mathcal{W}$ and $\mathcal{S}$ are not equivalent.

EXAMPLE 10.2
Let us consider the following instance of the inhabitation problem: $(x:[[]\to\tau,\sigma],\tau)$. In system $\mathcal{S}$, it is straightforward to construct a term having type τ in the type assignment giving to x both the type $[]\to\tau$ and the type σ:

$$\dfrac{\dfrac{}{x:[[]\to\tau]\vdash x:[]\to\tau} \qquad \dfrac{}{x:[\sigma]\vdash x:\sigma}}{x:[[]\to\tau,\sigma]\vdash xx:\tau}.$$

On the other hand, it is easy to see that this instance has no solution in system $\mathcal{W}$, where the argument of a term having a type of the form $[]\to\tau$ does not get any type.

Remark that $\mathcal{S}$ is at least as powerful as $\mathcal{W}$, for solving inhabitation problems:

FACT 10.3
Let (Γ,τ) be an instance of the inhabitation problem having a solution in system $\mathcal{W}$. Then (Γ,τ) has a solution in $\mathcal{S}$.

It is sufficient to apply the following transformation to typing derivations:

$$\dfrac{\dfrac{\vdots}{\Gamma\vdash t:[]\to\tau}}{\Gamma\vdash tu:\tau} \quad\Rightarrow\quad \dfrac{\dfrac{\vdots}{\Gamma\vdash t:[]\to\tau} \qquad \dfrac{\overline{x:[\sigma]\vdash x:\sigma}}{\vdash \mathtt{I}:[\sigma]\to\sigma}}{\Gamma\vdash t\mathtt{I}:\tau}$$

whenever the arrow elimination rule of system $\mathcal{W}$ concerns a functional type whose left-hand side is []. ■

Another remark is that in system $\mathcal{S}$ it is possible to make type assignments grow *ad libitum* when solving instances of the inhabitation problem:

FACT 10.4
Let (Γ,τ) be an instance of the inhabitation problem having a solution in system $\mathcal{S}$. Then $(\Gamma+x:[\sigma],\tau)$ has a solution in $\mathcal{S}$, for each variable x and type σ.

A suitable type derivation is the following:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma\vdash t:\tau}}{\Gamma\vdash\lambda z.t:[]\to\tau} \qquad \overline{x:[\sigma]\vdash x:\sigma}}{\Gamma+x:[\sigma]\vdash(\lambda z.t)x:\tau}$$

z being any fresh variable.

■

We have just seen that the set of inhabitation problems admitting a solution in $\mathcal{W}$ is a proper subset of those admitting a solution in $\mathcal{S}$. We conjecture that type inhabitation in $\mathcal{S}$ is decidable. Given an inhabitation problem (Γ,σ) for $\mathcal{S}$, the idea is to define a *finite* set of inhabitation problems $\{(\Gamma_i,\sigma_i)\}$ for $\mathcal{W}$ in such a way that (Γ,σ) has a solution if and only if *at least* one of the (Γ_i,σ_i) has a solution. The decidability of type inhabitation for $\mathcal{W}$ allows to conclude. The examples above suggest that (Γ,σ) has to be suitably cut down to define the (Γ_i,σ_i)'s. Without entering into the details of this construction, that is work in progress, let us provide a brief account of the algorithm deciding the type inhabitation for $\mathcal{W}$ [7]. Given an instance (Γ,σ), the algorithm produces all the *approximate normal forms* $\mathbf{a}$ such that there exists a typing derivation $\Phi\triangleright\Gamma\vdash_{\mathcal{W}}\mathbf{a}:\tau$. Again, let us avoid formal definitions and provide an example motivating the use of approximate normal forms:

EXAMPLE 10.5
Consider the inhabitation problem $(\emptyset,[[]\to\alpha]\to\alpha)$. It has infinitely many solutions, of the form

$$\dfrac{\dfrac{x:[[]\to\alpha]\vdash x:[]\to\alpha}{x:[[]\to\alpha]\vdash xt:\alpha}}{\vdash\lambda x.xt:[[]\to\alpha]\to\alpha},$$

where t is an arbitrary term. The approximate normal form produced by the algorithm is $\lambda x.x\Omega$, in this case.

The key property of non-idempotent types that makes the inhabitation problem for $\mathcal{W}$ decidable is that in this case types keep track faithfully of the different uses of variables in terms. The size of an approximate normal form **a** inhabiting a given type σ cannot grow indefinitely, since each occurrence of variable in **a** contributes to σ (remark that this is not the case for terms, because of 'useless' subterms like t in the example above).

11 Principal typings

The following notion of principal typing — which is system independent — is given in [73]. Let $\Theta = \langle \Gamma, \sigma \rangle$ be a typing in some typing system X and let

$$\mathtt{Terms}_X(\Theta) = \{t \text{ is a term } \mid \Gamma \vdash_X t{:}\sigma\}.$$

A **partial order** $\precsim_X$ on typings in system X is then defined by

$$\Theta \precsim_X \Theta' \Longleftrightarrow \mathtt{Terms}_X(\Theta) \subseteq \mathtt{Terms}_X(\Theta').$$

The pair $\Theta = \langle \Gamma, \sigma \rangle$ is a **principal typing (PT)** of t in X if $\Gamma \vdash_X t{:}\sigma$ and, for any typing Θ' of t in X, $\Theta \precsim_X \Theta'$. For instance, if we take the λ-calculus with simple types, then $\langle \{y{:}\beta \to \alpha\}, \beta \to \alpha \rangle$ is a principal typing of $\lambda x.yx$ while $\langle \{y{:}\alpha \to \alpha\}, \alpha \to \alpha \rangle$ is a principal typing of $\lambda x.y(yx)$[6]. In both cases, we can add irrelevant information in the type assignment since weakening is an admissible rule in the system.

On the other hand, the principal typing Θ of a term t is the *most general* typing of t, i.e. we can obtain any other typing of t by applying some syntactic operations on Θ. For *simple types*, just type substitutions — that replace type variables by types — and weakenings are enough. For instance, in the example above, one can obtain the typing $\langle \{y{:}\alpha \to \alpha\}, \alpha \to \alpha \rangle$ from $\langle \{y{:}\beta \to \alpha\}, \beta \to \alpha \rangle$ by unifying β and α. For *intersection types*, however, another operation on typings called *expansion* [11, 12, 18] is needed to build PT compositionally. For instance, the terms $t_0 = \lambda x.xy$ and $t_1 = \lambda f.f(fz)$ have $\langle \{y{:}[\alpha]\}, [[\alpha] \to \beta] \to \beta \rangle$ and $\langle \{z{:}[\alpha']\}, [[\alpha''] \to \beta', [\alpha'] \to \alpha''] \to \beta' \rangle$[7] as PT in system $\mathcal{W}$, respectively. To get a PT of $t_0 t_1$ from those of t_0 and t_1, one needs to identify the type $[\alpha] \to \beta$ with $[[\alpha''] \to \beta', [\alpha'] \to \alpha''] \to \beta'$, an operation which cannot be performed by using type substitution only, thus motivating the introduction of expansions [18].

A type system X is said to have the **principal typing property**, written PTP, if every typable term in system X has PT. In [18] the relation between PT of a term and its β-nf is pointed out, while in [62, 63] PTs are used to reconstruct β-nf from them.

This section reformulates with non-idempotent types the approach of [62, 63] based on idempotent types: first, we restrict system $\mathcal{W}$ introduced in Section 3 to another system $\mathcal{W}_r$ such that (1) all β-nfs have PT in $\mathcal{W}_r$, and (2) type substitutions are the only syntatic operations needed to obtain other typings of β-nfs in $\mathcal{W}_r$ starting from their PT[8]. Secondly, we explain the steps that allow to extend PT for β-nfs from system $\mathcal{W}_r$ to system $\mathcal{W}$, as developed in [63, 64]. In this case, a new syntactic operation called expansion is needed. Instead of developing formal definitions to introduce the expansion operation, we illustrate the main notions behind this concept by means of an example. Finally, as a consequence of the Subject Expansion property (cf. Theorem 4.3-1), any weakly β-normalizing term turns out to have PT in system $\mathcal{W}$.

[6] Note that the last typing is also a typing for the first term but the converse does not hold.

[7] Observe that, in a IT system, each occurrence of a term variable has a distinc type in a PT.

[8] Since $\mathcal{W}$ is a relevant system, no weakening is allowed.

Let us start by formalizing some of the needed notions. **Type substitutions** are functions that map type variables to strict types. Given a type substitution S, we extend S to strict types, intersection types and contexts as follows:

$$\begin{array}{lcl} S(\mathcal{M}\to\tau) & := & S(\mathcal{M})\to S(\tau) \\ S([\sigma_i]_{i\in I}) & := & [S(\sigma_i)]_{i\in I} \\ S(\Gamma)(x) & := & S(\Gamma(x)). \end{array}$$

The **domain** of a type substitution S is defined by $\mathtt{dom}(S)=\{\alpha \mid S(\alpha)\neq\alpha\}$. We write $\mathtt{TypeVar}(\tau)$ to denote the **set of type variables** occurring in τ. The extension of $\mathtt{TypeVar}$ to type assignments is defined as expected. A **renaming** is an injective type substitution S that maps type variables to fresh type variables, i.e. for any $\alpha\in\mathtt{dom}(S)$, $S(\alpha)$ is a variable and $S(\alpha)\notin\mathtt{dom}(S)$, and for any $\alpha,\beta\in\mathtt{dom}(S)$, if $\alpha\neq\beta$ then $S(\alpha)\neq S(\beta)$. Given a type τ, a **renaming for** τ is a renaming S such that for any $\alpha\in\mathtt{dom}(S)$, $S(\alpha)\notin\mathtt{TypeVar}(\tau)$. Principal typings are considered modulo renamings.

The following property about type substitutions holds for both systems $\mathcal{S}$ and $\mathcal{W}$.

PROPOSITION 11.1
Let S be a type substitution and $X\in\{\mathcal{W},\mathcal{S}\}$. If $\Phi\triangleright\Gamma\vdash_X t{:}\sigma$ then $\Phi'\triangleright S(\Gamma)\vdash_X t{:}S(\sigma)$ and $\mathtt{sz}(\Phi)=\mathtt{sz}(\Phi')$.

By induction on Φ. ■

We now define the **restricted type system** $\mathcal{W}_r$ by replacing the typing rule ($\mathtt{ax}$) in system $\mathcal{W}$ by the following typing rule:

$$\frac{}{x{:}[[\sigma_1]\to\cdots\to[\sigma_n]\to\alpha]\vdash x{:}[\sigma_1]\to\cdots\to[\sigma_n]\to\alpha}\,(\mathtt{ax}_r),$$

where $n\geq 0$. Remark that any typing in $\mathcal{W}_r$ is also a typing in $\mathcal{W}$.

The restricted system $\mathcal{W}_r$ allows us to define PTs by using type substitutions only, in contrast to [18, 44, 58, 59, 69–71] which also use other sophisticated operations such as expansions.

We then reformulate the type inference algorithm for normal forms given in [62, 63] which is defined as a function $\mathtt{Infer}$ from $\mathcal{NF}(\beta)$ to pairs $\langle\Gamma,\sigma\rangle$, where Γ is a type assignment and σ is a strict type.

$\mathtt{Infer}(t)=$

Case $t=\lambda x.t'$
  let $\langle\Gamma',\sigma\rangle=\mathtt{Infer}(t')$
  return $\langle\Gamma'\setminus\setminus x,\Gamma'(x)\to\sigma\rangle$
Case $t=xt_1\cdots t_m$
  let $\langle\Gamma_1,\sigma_1\rangle=\mathtt{Infer}(t_1)$
  $\vdots$
  $\langle\Gamma_m,\sigma_m\rangle=\mathtt{Infer}(t_m)$
  α be a fresh type variable
  return $\langle\{x{:}[[\sigma_1]\to\cdots\to[\sigma_m]\to\alpha]\}+_{j=1\ldots m}\Gamma_j,\alpha\rangle$

The **freshness** notion for type variables used above is necessary to prove completeness, since it guarantees that non-overlapping calls to $\mathtt{Infer}$ return pairs with disjoints sets of type variables.

THEOREM 11.2 (Soundness)
If $t\in\mathcal{NF}(\beta)$ and $\mathtt{Infer}(t)=\langle\Gamma,\sigma\rangle$, then $\Gamma\vdash_{\mathcal{W}_r} t{:}\sigma$.

By structural induction on t. ■

COROLLARY 11.3
If $t \in \mathcal{NF}(\beta)$ then t is $\mathcal{W}_r$-typable.

THEOREM 11.4 (Completeness for β-nfs in system $\mathcal{W}_r$)
Let $t \in \mathcal{NF}(\beta)$ and $\langle \Gamma', \sigma' \rangle = \texttt{Infer}(t)$. If $\Gamma \vdash_{\mathcal{W}_r} t : \sigma$ then there exists a type substitution S such that $S(\Gamma') = \Gamma$ and $S(\sigma') = \sigma$.

By structural induction on t. ■

COROLLARY 11.5
If $t \in \mathcal{NF}(\beta)$ then $\langle \Gamma, \sigma \rangle = \texttt{Infer}(t)$ is a principal typing of t in $\mathcal{W}_r$.

So far we have presented results only for β-nfs and for the restricted version of $\mathcal{W}$, the system $\mathcal{W}_r$. In the next section we analyse the shape of typings in $\mathcal{W}_r$ as a step to achieve PT for the full system $\mathcal{W}$.

11.1 Principal typings for weakly normalizing terms

In this section, we start by establishing the general form of the $\mathcal{W}_r$-typings built by the algorithm `Infer`. Since this form is not enough to describe also $\mathcal{W}$-typings, we then extend this notion in order to characterize PT for β-nfs in system $\mathcal{W}$. The result can then be trivially extended to any weakly normalizing term.

Let us start by defining a set of strict (resp. multiset) types representing the kind of strict (resp. multiset) types returned by the algorithm `Infer` introduced above. The sets of types assigned to terms variables, written $\mathcal{T}_V$, and the set of types assigned to β-normal forms, denoted by $\mathcal{T}_{NF}$, are defined by the following grammars:

$$(\mathcal{T}_V) \quad \rho ::= \alpha \mid [\varphi] \to \rho \qquad (\mathcal{T}_{NF}) \quad \begin{array}{lll} \varphi & ::= & \alpha \mid \mathcal{M}_V \to \varphi \\ \mathcal{M}_V & ::= & [\rho_i]_{i \in I} \end{array}$$

with the additional constraint that $[\varphi] \to \rho \in \mathcal{T}_V$ implies $\texttt{TypeVar}(\varphi) \cap \texttt{TypeVar}(\rho) = \emptyset$.

For instance, $[\alpha] \to \beta$ is in $\mathcal{T}_V$ while $[\alpha_1, \alpha_2] \to \beta$, $[\beta] \to \beta$ and $[] \to \beta$ are not.

We denote by $\mathcal{C}$ the set of type assignments containing only multiset types $\mathcal{M}_V$, i.e. such that every strict type in the multiset $\Gamma(x)$, for $\Gamma \in \mathcal{C}$, belongs to $\mathcal{T}_V$. Remark that $\mathcal{C}$ is closed under $+$.

The intuition about those sets is that types in $\mathcal{T}_V$ are the types assigned to term variables in a PT. For instance, the PT of the term xt will contain an assignment like $x : [\tau] \to \beta$ but not like $x : [\tau, \tau] \to \beta$. The multiple occurrences of term variables in a term would then get an intersection type $\mathcal{M}_V$. For instance, $\texttt{Infer}(\lambda x.y(yx)) = \langle \{y : [[\alpha] \to \beta', [\beta] \to \alpha]\}, [\beta] \to \beta' \rangle$. Formally,

LEMMA 11.6
Let $\mathcal{A}$ be the image of the algorithm `Infer`. Then, $\mathcal{A} \subseteq \mathcal{C} \times \mathcal{T}_{NF}$.

By structural induction on $t \in \mathcal{NF}(\beta)$. ■

For the full $\mathcal{W}$ system, type substitutions only are not enough to obtain arbitrary typings for β-nfs. For instance, $\langle \{y : [[\alpha_1, \alpha_2] \to \beta', [\beta] \to \alpha_1, [\beta] \to \alpha_2]\}, [\beta, \beta] \to \beta' \rangle$ is a $\mathcal{W}$-typing of $\lambda x.y(yx)$ but it cannot be obtained from $\texttt{Infer}(\lambda x.y(yx))$ through a type substitution. An expansion operation will then be necessary to achieve completness for $\mathcal{W}$.

The expansion operation allows to introduce copies of strict types on the left-hand side of arrows, corresponding to the duplication of typing derivations. The first important issue in defining a sound expansion operation on typings is to identify the set of types to be expanded (duplicated). A multiset of marked types, called *nucleus* in [18], identifies which occurrences of types have to be expanded, i.e. duplicated. Once the expansion operation is applied to the typing of some term t according to the nucleus, the new pair obtained is still a typing for t.

For instance, let $\Theta = \mathtt{Infer}(\lambda x.y(yx)) = \langle \{y{:}[[\alpha]\to\beta', [\beta]\to\alpha]\}, [\beta]\to\beta'\rangle$, corresponding to the following typing derivation:

$$\Phi := \dfrac{\dfrac{\overline{y{:}[[\alpha]\to\beta']\vdash y{:}[\alpha]\to\beta'} \qquad \Phi_{(yx)} \rhd y{:}[[\beta]\to\alpha]; x{:}[\beta]\vdash yx{:}\alpha}{y{:}[[\alpha]\to\beta', [\beta]\to\alpha]; x{:}[\beta]\vdash y(yx){:}\beta'}}{\overline{y{:}[[\alpha]\to\beta', [\beta]\to\alpha]\vdash \lambda x.y(yx){:}[\beta]\to\beta'}},$$

where

$$\Phi_{(yx)} := \dfrac{\overline{y{:}[[\beta]\to\alpha]\vdash y{:}[\beta]\to\alpha} \qquad \Phi_x := \overline{x{:}[\beta]\vdash x{:}\beta}}{y{:}[[\beta]\to\alpha]; x{:}[\beta]\vdash yx{:}\alpha}.$$

Then, considering $[[\alpha]\to\beta', [\underline{\beta}]\to\alpha, [\underline{\beta}]\to\beta']$ as a nucleus for Θ gives the alternative/expanded typing $\langle \{y{:}[[\alpha]\to\beta', [\beta_1,\beta_2]\to\alpha]\}, [\beta_1,\beta_2]\to\beta'\rangle$, together with the following derivation:

$$\Phi' := \dfrac{\dfrac{\overline{y{:}[[\alpha]\to\beta']\vdash y{:}[\alpha]\to\beta'} \qquad \Phi'_{(yx)} \rhd y{:}[[\beta_1,\beta_2]\to\alpha]; x{:}[\beta_1,\beta_2]\vdash yx{:}\alpha}{y{:}[[\alpha]\to\beta', [\beta_1,\beta_2]\to\alpha]; x{:}[\beta_1,\beta_2]\vdash y(yx){:}\beta'}}{\overline{y{:}[[\alpha]\to\beta', [\beta_1,\beta_2]\to\alpha]\vdash \lambda x.y(yx){:}[\beta_1,\beta_2]\to\beta'}},$$

where

$$\Phi'_{(yx)} := \dfrac{\overline{y{:}[[\beta_1,\beta_2]\to\alpha]\vdash y{:}[\beta_1,\beta_2]\to\alpha} \qquad \Phi^1_x := \overline{x{:}[\beta_1]\vdash x{:}\beta_1} \qquad \Phi^2_x := \overline{x{:}[\beta_2]\vdash x{:}\beta_2}}{y{:}[[\beta_1,\beta_2]\to\alpha]; x{:}[\beta_1,\beta_2]\vdash yx{:}\alpha}.$$

Remark that we have only duplicated the (unique) typing derivation Φ_x in Φ but, in general, there could be any number of derivations of x after expansion. Then, the expansion operation needs also to specify the number of copies of Φ_x to be created in the process, denoted in our example by $\mathtt{Exp}_{(2,\beta)}(\Theta)$, which means, *expand β twice in the derivation of typing* Θ. Remark also that the duplicating process generates distinct type variable names for the new derivations of x. If we wanted to obtain a derivation as Φ' above but with both type derivations for x assigning the *same* type, then we would simply need to apply a type substitution unifying β_1 and β_2.

Another possible nucleus for the same typing is $\{[\underline{\alpha}]\to\beta', [\underline{\beta}]\to\alpha, [\underline{\beta}]\to\beta'\}$, corresponding to the duplication of the derivation of the subterm yx in Φ, denoted by $\mathtt{Exp}_{(2,\alpha)}(\Theta) = \langle \{y{:}[[\alpha_1,\alpha_2]\to\beta', [\beta_1]\to\alpha_1, [\beta_2]\to\alpha_2]\}, [\beta_1,\beta_2]\to\beta'\rangle$ with a derivation Φ'' of the form:

$$\Phi := \dfrac{\dfrac{\overline{y{:}[[\alpha_1,\alpha_2]\to\beta']\vdash y{:}[\alpha_1,\alpha_2]\to\beta'} \qquad (\Phi^i_{(yx)} \rhd y{:}[[\beta_i]\to\alpha_i]; x{:}[\beta_i]\vdash yx{:}\alpha_i)_{i\in I}}{y{:}[[\alpha_1,\alpha_2]\to\beta', [\beta_1]\to\alpha_1, [\beta_2]\to\alpha_2]; x{:}[\beta_1,\beta_2]\vdash y(yx){:}\beta'}}{\overline{y{:}[[\alpha_1,\alpha_2]\to\beta', [\beta_1]\to\alpha_1, [\beta_2]\to\alpha_2]\vdash \lambda x.y(yx){:}[\beta_1,\beta_2]\to\beta'}}.$$

for $I = \{1, 2\}$ where:

$$\Phi^i_{(yx)} := \frac{\overline{y{:}[[\beta_i] \to \alpha_i] \vdash y{:}[\beta_i] \to \alpha_i} \qquad \Phi^i_x := \overline{x{:}[\beta_i] \vdash x{:}\beta_i}}{y{:}[[\beta_i] \to \alpha_i]; x{:}[\beta_i] \vdash yx{:}\alpha_i}.$$

In the typing Φ above, the type variable β' cannot be duplicated[9].

To introduce the expansion operation, a set of *ground* typings is defined, having $C \times \mathcal{T}_{NF}$ as a proper subset and being closed for the expansion operation described above. Given a typing returned by `Infer`, one can apply expansions, possibly interleaved with renamings whenever necessary, and eventually type substitutions. Hence, type variables are kept distinct when expansions are applied.

Completeness in the sense of Theorem 11.4 can then be proved with two kinds of operations: type substitutions (including renamings) and expansions. To combine these operations, as indicated above, we define the **composition** of operations O_1 and O_2, denoted by $O_2 \circ O_1$, as the sequence of applications $O_2(O_1(\_))$. Let $S_1, \ldots, S_n$ be type substitutions and $O_1, \ldots, O_m$ be either expansions or renamings. The composition $S_n \circ \cdots \circ S_1 \circ O_m \circ \cdots \circ O_1$ is called a **chain**.

THEOREM 11.7 (Completeness for β-nfs in system $\mathcal{W}$)
Let $t \in \mathcal{NF}(\beta)$ and $\texttt{Infer}(t) = \langle \Gamma_p, \mu_p \rangle$. If $\Gamma \vdash_{\mathcal{W}} t{:}\tau$ then there exists a chain C such that $C(\langle \Gamma_p, \mu_p \rangle) = \langle \Gamma, \tau \rangle$.

COROLLARY 11.8
Let $t \in \mathcal{WN}(\beta)$, t' its β-nf s.t $\texttt{Infer}(t') = \langle \Gamma_p, \mu_p \rangle$. If $\Gamma \vdash_{\mathcal{W}} t{:}\tau$ then there exists a chain C such that $C(\langle \Gamma_p, \mu_p \rangle) = \langle \Gamma, \tau \rangle$.

12 Conclusion

We presented a survey of results stemming from the application of non-idempotent intersection type systems to the λ-calculus, encompassing both classical characterizations of classes of meaningful programs and new connections with complexity and resource consumption issues. To unveil the wide scope and potential of this approach, in a unified framework, is the aim of this work. By the way, we have presented some new material. Here are some directions for future work:

- We conjecture that the decidability of the inhabitation problem for system $\mathcal{S}$ can be reduced to that of system $\mathcal{W}$.
- We would like to use the ideas in [32] to relate, by means of intersection types, the call-by-need calculus to the standard notion of needed reduction.
- Moreover, the completeness proof of *weak* call-by-need in [39] could be extended to *strong* call-by-need, i.e. to a strategy which lazily reduces terms until strong normal forms are reached.
- A notion of observability for higher-order languages with pair patterns is characterized in [8] by means of non-idempotent IT. It could be interesting to explore other notions of observability, as well as the application of these techniques to a more general framework of functional languages with dynamic pattern matching.
- It could be useful to investigate the relation between principal typing for the $\mathcal{S}$ system and the exact bounds given in [5], obtained through a non-standard notion of *principal* (and *optimal*) derivations.
- Also, the notions of principal typings in $\mathcal{W}$ could be relevant for the development of type inference algorithms based on *β-unification* [11, 44, 58].

[9] Remark that β' does not occur as a type in the multisets representing the nuclei above.

Acknowledgements

We would like to thank Beniamino Accattoli, Alexis Bernadet, Daniel de Carvalho, Stéphane Lengrand and Simona Ronchi Della Rocca for interesting discussions.

References

[1] B. Accattoli, E. Bonelli, D. Kesner and C. Lombardi. A nonstandard standardization theorem. In *the 41st Annual ACM Symposium on Principles of Programming Languages (POPL)*, P. Sewell, ed., pp. 659–670. ACM, 2014.

[2] 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.

[3] H. Barendregt, W. Dekkers and R. Statman. *Lambda Calculus with Types*. Perspectives in Logic. Cambridge University Press, 2013.

[4] A. Bernadet. *Types Intersections Non-idempotents Pour Rafiner la Normalization Forte Avec des Informations Quantitatives*. PhD Thesis, École Polytechnique, 2014.

[5] A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalization. *Logical Methods in Computer Science*, **9**(4), 2013.

[6] G. Boudol, P.-L. Curien and C. Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, **9**, 437–482, 1999.

[7] A. Bucciarelli, D. Kesner and S. Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *the 8th International Conference on Theoretical Computer Science (TCS)*, Vol. 8705 of *LNCS*, J. Díaz, I. Lanese and D. Sangiorgi, eds, pp. 341–354. Springer, 2014.

[8] A. Bucciarelli, D. Kesner and S. Ronchi Della Rocca. Observability for pair pattern calculi. In *the 13th International Conference on Typed Lambda Calculus and Applications (TLCA)*, Vol. 38 of *LIPIcs*, T. Altenkirch, ed., pp. 123–137. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[9] A. Bucciarelli, D. Kesner and D. Ventura. Strong normalization through intersection types and memory. In *the 10th International Workshop on Logical and Semantical Frameworks, with Applications (LSFA)*, Electronic Notes in Theoretical Computer Science. M. Benevides and R. Thiemann, eds, Elsevier Science B. V., 2016.

[10] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In *Logic and Computer Science*, Vol. 31 of *APIC Series*, P. Odifreddi, ed., pp. 19–75. Academic Press, 1990.

[11] S. Carlier and J. B. Wells. Expansion: the crucial mechanism for type inference with intersection types: a survey and explanation. *Electronic Notes in Theoretical Computer Science*, **136**, 173–202, 2005.

[12] S. Carlier and J. B. Wells. The algebra of expansion. *Fundamenta Informaticae*, **121**, 43–82, 2012.

[13] A. Carraro and G. Guerrieri. A semantical and operational account of call-by-value solvability. In *the 17th International Conference Foundations of Software Science and Computation Structures (FOSSACS)*, Vol. 8412 of *LNCS*, A. Muscholl, ed., pp. 103–118. Springer-Verlag, 2014.

[14] G. Castagna, M. Dezani-Ciancaglini, E. Giachino and L. Padovani. Foundations of session types. In *Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, September 7–9, 2009, Coimbra, Portugal*, A. Porto and F. J. López-Fraguas, eds, pp. 219–230. ACM, 2009.

[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 λ-calculus. *Notre Dame Journal of Formal Logic*, **21**, 685–693, 1980.
[17] 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.
[18] M. Coppo, M. Dezani-Ciancaglini and B. Venneri. Principal type schemes and λ-calculus semantics. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. Seldin and J. Hindley, eds, pp. 536–560. Academic Press, 1980.
[19] M. Coppo, M. Dezani-Ciancaglini and B. Venneri. Functional characters of solvable terms. *Zeitschrift für mathematiche Logik und Grundlagen der Mathematik*, **27**, 45–58, 1981.
[20] T. Coquand and A. Spiwack. A proof of strong normalization using domain theory. *Logical Methods in Computer Science*, **3**(4), 2007.
[21] H. Curry and R. Feys. *Combinatory Logic volume I*. North-Holland Co., 1958.
[22] F. Damiani and P. Giannini. A decidable intersection type system based on relevance. In *International Conference on Theoretical Aspects of Computer Software (TACS)*, Vol. 789 of *LNCS*, M. Hagiya and J. C. Mitchell, eds, pp. 707–725. Springer-Verlag, 1994.
[23] E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *the 6th Workshop on Intersection Types and Related Systems (ITRS)*, Vol. 121 of *EPTCS*, S. Graham-Lengrand and L. Paolini, eds, pp. 48–57, 2013.
[24] D. de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. These de doctorat, Université Aix-Marseille II, 2007.
[25] D. de Carvalho and L. Tortora de Falco. A semantic account of strong normalization in linear logic. *Information and Computation*, **248**, 104–129, 2016.
[26] M. Dezani-Ciancaglini, S. Ghilezan and S. Likavec. Behavioural inverse limit lambda-models. *Theoretical Computer Science*, **316**, 49–74, 2004.
[27] J. Díaz, I. Lanese and D. Sangiorgi, editors. *the 8th International Conference on Theoretical Computer Science (TCS)*, Vol. 8705 of *LNCS*. Springer, 2014.
[28] J. Dunfield. Elaborating intersection and union types. *Journals of Functional Programming*, **24**, 133–165, 2014.
[29] J. Dunfield and F. Pfenning. Type assignment for intersections and unions in call-by-value languages. In *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7–11, 2003, Proceedings*, Vol. 2620 of *Lecture Notes in Computer Science*, A. D. Gordon, ed., pp. 250–266. Springer, 2003.
[30] T. Ehrhard. Collapsing non-idempotent intersection types. In *the 26th Annual Conference on Computer Science Logic (CSL)*, Vol. 16 of *LIPIcs* P. Cégielski and A. Durand, eds, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
[31] T. Freeman and F. Pfenning. Refinement types for ML. In *Proceedings of the ACM SIGPLAN'91 Conference on Programming Language Design and Implementation (PLDI), Toronto, Ontario, Canada, June 26–28, 1991*, D. S. Wise, ed., pp. 268–277. ACM, 1991.
[32] P. Gardner. Discovering needed reductions using type theory. In *International Conference on Theoretical Aspects of Computer Software (TACS)*, Vol. 789 of *LNCS*, M. Hagiya and J. C. Mitchell, eds, pp. 555–574. Springer, 1994.
[33] S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame Journal of Formal Logic*, **37**, 44–52, 1996.

[34] P. D. Gianantonio, F. Honsell and M. Lenisa. A type assignment system for game semantics. *Theoretical Computer Science*, **398**, 150–169, 2008.
[35] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, **50**, 1–102, 1987.
[36] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types*, Vol. 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.
[37] M. Hagiya and J. C. Mitchell, eds. *International Conference on Theoretical Aspects of Computer Software (TACS)*, Vol. 789 of *LNCS*. Springer, 1994.
[38] H. Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *the 8th International Workshop on Computer Science Logic (CSL)*, Vol. 933 of *LNCS*, L. Pacholski and J. Tiuryn, eds, pp. 61–75. Springer, 1995.
[39] D. Kesner. Reasoning about call-by-need by means of types. In *the 19th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, Vol. 9634 of *LNCS*, B. Jacobs and C. Löding, eds, Springer, 2016.
[40] D. Kesner and D. Ventura. Quantitative types for the linear substitution calculus. In *the 8th International Conference on Theoretical Computer Science (TCS)*, Vol. 8705 of *LNCS*, J. Díaz, I. Lanese and D. Sangiorgi, eds, pp. 296–310. Springer, 2014.
[41] D. Kesner and D. Ventura. A resource aware semantics for a focused intuitionistic calculus. *Mathematical Structures in Computer Science*, 1–34, 2017.
[42] A. Kfoury. A linearization of the lambda-calculus and consequences. *Technical Report*, Boston University, 1996.
[43] A. Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, **10**, 411–436, 2000.
[44] A. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables. *Theoretical Computer Science*, **311**, 1–70, 2004.
[45] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of beta-strong normalization in typed lambda-calculi. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pp. 311–321. IEEE Computer Society, 1995.
[46] J.-L. Krivine. *Lambda-calculus, Types and Models*. Masson, 1990.
[47] C. Mossin. Exact flow analysis. *Mathematical Structures in Computer Science*, **13**, 125–156, 2003.
[48] P. M. Neergard. Theoretical pearls: a bargain for intersection types: a simple strong normalization proof. *Journal of Functional Programming*, **15**, 669–677, 2005.
[49] L. Padovani. Session types = intersection types + union types. In *Proceedings Fifth Workshop on Intersection Types and Related Systems, ITRS 2010, Edinburgh, U.K., 9th July 2010.*, Vol. 45 of *EPTCS*, E. Pimentel, B. Venneri and J. B. Wells, eds, pp. 71–89, 2010.
[50] L. Padovani. On projecting processes into session types. *Mathematical Structures in Computer Science*, **22**, 237–289, 2012.
[51] M. Pagani and S. Ronchi Della Rocca. Solvability in resource lambda-calculus. In *the 13rd International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, Vol. 6014 of *LNCS*, L. Ong, ed., pp. 358–373. Springer, 2011.
[52] J. Palsberg and C. Pavlopoulou. From polyvariant flow information to intersection and union types. *Journal of Functional Programming*, **11**, 263–317, 2001.
[53] M. Pereira, S. Alves and M. Florido. Liquid intersection types. In *Proceedings Seventh Workshop on Intersection Types and Related Systems, ITRS 2014, Vienna, Austria, 18 July 2014*, Vol. 177 of *EPTCS*, J. Rehof, ed., pp. 24–42, 2015.

[54] M. Piccolo. Strong normalization in the π-calculus with intersection and union types. *Fundamental Information*, **121**, 227–252, 2012.
[55] G. Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. Seldin and J. Hindley, eds, pp. 561–578. Academic Press, 1980.
[56] S. J. Ramsay. Exact intersection type abstractions for safety checking of recursion schemes. In *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014*, pp. 175–186, 2014.
[57] J. C. Reynolds. Design of the programming language forsythe. In *Algol-like Languages*, Progress in Theoretical Computer Science, P. O. Hearn and R. Tennent, eds, pp. 173–233. Birkhuser Boston, 1997.
[58] S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, **59**, 1–29, 1988.
[59] S. Ronchi Della Rocca and B. Venneri. Principal type scheme for an extended type theory. *Theoretical Computer Science*, **28**, 151–169, 1984.
[60] S. Salvati, G. Manzonetto, M. Gehrke and H. Barendregt. Loader and Urzyczyn are logically related. In *the 39th International Colloquium on Automata, Languages, and Programming (ICALP)*, Vol. 7392 of *Lecture Notes in Computer Science*, A. Czumaj, K. Mehlhorn, A. M. Pitts and R. Wattenhofer, eds, pp. 364–376. Springer, 2012.
[61] D. Sangiorgi and D. Walker. *The Pi-Calculus - A Theory of mobile Processes*. Cambridge University Press, 2001.
[62] E. Sayag and M. Mauny. Characterization of principal types of normal forms in an intersection type system. In *the 16th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Vol. 1180 of *LNCS*, V. Chandru and V. Vinay, eds, pp. 335–346. Springer, 1996.
[63] E. Sayag and M. Mauny. A presentation of the intersection type discipline through principal typings of normal forms. Technical Report RR-2998, INRIA, 1996.
[64] E. Sayag and M. Mauny. Structural properties of intersection types. In *Proceedings of the 8th International Conference on Logic and Computer Science – Theoretical Foundations of Computing (LIRA)*, pp. 167–175, Novi Sad, 1997.
[65] W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, **32**, 198–212, 1967.
[66] P. Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, **64**, 1195–1215, 1999.
[67] S. Valentini. An elementary proof of strong normalization for intersection types. *Archive of Mathematical Logic*, **40**, 475–488, 2001.
[68] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, **102**, 135–163, 1992.
[69] S. van Bakel. Principal type schemes for the strict type assignment system. *Journal of Logic and Computation*, **3**, 643–670, 1993.
[70] S. van Bakel. Intersection type assignment systems. *Theoretical Computer Science*, **151**, 385–435, 1995.
[71] S. van Bakel. Strict intersection types for the lambda calculus. *ACM Computing Surveys*, **43**, 20, 2011.
[72] F. van Raamsdonk. *Confluence and Normalization for Higher-Order Rewriting*. PhD Thesis, Amsterdam University, Netherlands, 1996.

[73] J. B. Wells. The essence of principal typings. In *the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, Vol. 2380 of *LNCS*, P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz and R. Conejo, eds, pp. 913–925. Springer, 2002.

Received 31 January 2016