Characterisation of Strongly Normalising $\lambda\mu$-Terms

(ITRS'12, EPTCS 121, pp 1-17, 2013)

Steffen van Bakel
Imperial College London
London, UK
svb@doc.ic.ac.uk

Franco Barbanera
Università di Catania
Catania, Italy
barba@dmi.unict.it

Ugo de'Liguoro
Università di Torino
Torino, Italy
deliguoro@di.unito.it

Abstract

We provide a characterisation of strongly normalising terms of the $\lambda\mu$-calculus by means of a type system with intersection and product types. The presence of the latter and a restricted use of the type ω enable us to represent the particular notion of continuation used in the literature for the definition of semantics for the $\lambda\mu$-calculus. This makes it possible to lift the well-known characterisation property for strongly-normalising λ-terms - that uses intersection types - to the $\lambda\mu$-calculus. From this result an alternative proof of strong normalisation for terms typeable in Parigot's propositional logical system follows, by means of an interpretation of that system into ours.

Introduction

Parigot's $\lambda\mu$-calculus [20] is an extension of the λ-calculus [11, 9] that was first introduced in [20] to express a notion of (confluent) computation with classical proofs in Gentzen's *sequent calculus* LK. That calculus was introduced in [15] as a logical system in which the rules only introduce connectives (but on either side of a sequent), in contrast to *natural deduction* (also introduced in [15]) which uses rules that introduce or eliminate connectives in the logical formulae. Natural deduction normally derives statements with a single conclusion, whereas LK allows for multiple conclusions, deriving sequents of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$, where $A_1, \ldots, A_n$ is to be understood as $A_1 \wedge \ldots \wedge A_n$ and $B_1, \ldots, B_m$ is to be understood as $B_1 \vee \ldots \vee B_m$.

With $\lambda\mu$, Parigot created a multi-conclusion typing system that is, in fact, based on a mixture of Gentzen's two approaches: the system is a natural deduction system that has *introduction* and *elimination* rules, but derivable statements have the shape $\Gamma \vdash M : A \mid \Delta$, where A is the main conclusion of the statement, expressed as the *active* conclusion. Here Δ contains the alternative conclusions, consisting of pairs of Greek characters and types; the left-hand context Γ, as usual, contains pairs of Roman characters and types, and represents the types of the free term variables of M. This yields a logic with *focus* where the main conclusion is the focus of the proof; derivable judgements correspond to provable statements in *minimal classical logic* [1]. In addition to the normal λ-calculus reduction rules, Parigot needed to express that the focus of the derivation (proof) changes; he therefore added *structural* rules, where elimination takes place for a type constructor that appears in one of the alternative conclusions (the Greek variable is the name given to a subterm). This is achieved by extending the syntax with two new constructs $[\alpha]M$ and $\mu\alpha.M$ that act as witness to *deactivation* and *activation*, which together move the focus of the derivation. The collection of reduction rules Parigot defined are carefully engineered to yield a confluent reduction system; normally, systems based on classical logic are not confluent, as is the case for example for the Symmetric λ-calculus [8], $\overline{\lambda}\mu\tilde{\mu}$ [14], and $\mathcal{X}$ [7].

In spite of being motivated by classical logic, the $\lambda\mu$-calculus itself is type free. As a consequence there exist more terms than proofs, and properties of pure $\lambda\mu$-terms have been extensively studied (see e.g. [23, 18, 24]). In particular, among them there are perfectly meaningful terms that do not correspond to any proof, like fixed-point constructors for example. The basic idea here to turn non-constructive proofs into algorithms is to add a form of continuation by means of *names* and *μ-abstraction* to capture (a notion of) control. However, continuations introduce a great deal of complexity to the calculus' semantics and inspired by the results proven in [4] we decided to explore the possibility of defining *filter* semantics for $\lambda\mu$. Starting from Streicher and Reus' denotational semantics of $\lambda\mu$ in [25], in [6] we have introduced an intersection type assignment system that induces a filter model. This, essentially, is a logical description of the domain-theoretic model of [25], with the advantage of providing a formal tool to reason about the meaning of terms.

One of the main results for $\lambda\mu$, proved in [21], states that all $\lambda\mu$-terms that correspond to proofs of second-order natural deduction are strongly normalising; the reverse of this property does not hold for Parigot's system, since there, for example, not all terms in normal form are typeable.

The full characterisation of strong normalisation (M is strong normalising if and only if M is typeable) is a property that is shown for various intersection systems for the λ-calculus, and towards the end of [6] we conjectured that in an appropriate subsystem we would be able to type exactly all strongly normalising $\lambda\mu$-terms as well. The first to state the characterisation result was Pottinger [22] for a notion of type assignment similar to the intersection system of [12, 13], but extended in that it is also closed for η-reduction, and is defined without the type constant ω. However, to show that all typeable terms are strongly normalisable, [22] only *suggests* a proof using Tait's computability technique [26]. A detailed proof, using computability, in the context of the ω-free BCD-system [10] is given in [2]; to establish the same result saturated sets are used by Krivine in [19] (chapter 4), in Ghilezan's survey [16], and in [5].

The converse of that result, the property that all strongly normalisable terms are typeable has proven to be more elusive: it has been claimed in many papers but not shown in full (we mention [22, 2, 16]); in particular, the proof for the property that type assignment is closed for subject expansion (the converse of subject reduction) is dubious. Subject expansion can only reliably be shown for *left-most outermost* reduction, which is used for the proofs in [19, 3, 5], and our result follows that approach as well.

In the full system of [6], all terms are typeable with ω and this clearly interferes with the termination property. However, the problem we face is slightly more complex than straightforwardly removing ω, as done in [2, 3]. In the model (for details, see [6]) a *continuation* is an infinite tuple of terms, which is typed in the system by (a finite intersection of) types $\kappa = \delta_1 \times \cdots \delta_k \times \omega$ for some $k > 0$, where the leading $\delta_1, \ldots, \delta_k$ encode the information about the first k terms in the tuple, while the ending ω represents the lack of information about the remaining infinite part. This implies that, for our system for $\lambda\mu$, we cannot remove ω completely. To solve this problem, we first restrict types to those having ω only as the final part of a product type; we then suitably modify the standard interpretation of intersection types, adapting Tait's argument in such a way that the semantics of κ is the *set of all finite tuples* $\vec{L}$ (called *stacks*) of strongly normalising terms that begin with k terms $L_1 \ldots, L_k$ that belong to the interpretations of, respectively, $\delta_1, \ldots, \delta_k$. For this restricted system, we will show that typeability characterises strong normalisability for $\lambda\mu$-terms.

As a consequence of our characterisation result we also obtain an alternative proof of Parigot's termination result [21] (for the propositional fragment), by interpreting ordinary types into our intersection types and proving that the translation preserves derivability from

Parigot's system to ours.

Outline of this paper. In Section 1, we will briefly recall Parigot's untyped $\lambda\mu$-calculus [20]. After defining appropriate sets of types in 2.1, a pre-order over types, and our typeing system in Section 2.2, we will show that typeability implies strong normalisation. The opposite implication, proved in Section 2.3, will complete our main results. The alternative proof of Parigot's theorem for the propositional fragment will be developed in Section 3, and we finish by giving concluding remarks.

1 The $\lambda\mu$-calculus

In this section we present Parigot's pure $\lambda\mu$-calculus as introduced in [20], slightly changing the notation.

Definition 1.1 (Term Syntax [20]) *i*) The sets Trm of *terms* and Cmd of *commands* are defined inductively by the following grammar (where $x \in Var$, a set of *term variables*, and $\alpha \in Name$, a set of *names*, both denumerable):

$$\begin{array}{rcll} M,N & ::= & x \mid \lambda x.M \mid MN \mid \mu\alpha.C & \text{(terms)} \\ C & ::= & [\alpha]M & \text{(commands)} \end{array}$$

ii) We call $\vec{L} \equiv L_1 : \cdots : L_k$ a *stack of terms*; we denote the set of all finite (possibly empty) stacks of terms by Trm^*, and write ϵ for the empty stack. If $M \in \mathsf{Trm}$ and $\vec{L} \equiv L_1 : \cdots : L_k$ then $M : \vec{L} \equiv M : L_1 : \cdots : L_k \in \mathsf{Trm}^*$, while we define $M(P : \vec{L}) \stackrel{\Delta}{=} MP\vec{L}$, so $M\vec{L} \equiv ML_1 \cdots L_k$.

We will often speak of a stack rather than a stack of terms. For convenience of notation, for $\vec{L} = L_1 : \cdots : L_k \in \mathsf{Trm}^*$, we introduce the notation:

$$M[\alpha \Leftarrow \vec{L}] \;\stackrel{\Delta}{=}\; M[\alpha \Leftarrow L_1][\alpha \Leftarrow L_2] \cdots [\alpha \Leftarrow L_n]$$

when each L_i does not contain α. In particular, $M[\alpha \Leftarrow \epsilon] \stackrel{\Delta}{=} M$. Notice that, by definition of structural substitution,

$$[\alpha]M[\alpha \Leftarrow \vec{L}] \;\stackrel{\Delta}{=}\; [\alpha]M[\alpha \Leftarrow L_1][\alpha \Leftarrow L_2] \cdots [\alpha \Leftarrow L_n] \;\stackrel{\Delta}{=}\; [\alpha](M[\alpha \Leftarrow \vec{L}])\vec{L}$$

As usual, we consider λ and μ to be binders; we adopt Barendregt's convention on terms, and will assume that free and bound variables are different; the sets $fv(M)$ and $fn(M)$ of, respectively, *free variables* and *free names* in a term M are defined in the usual way.

Definition 1.2 (Substitution [20]) Substitution takes two forms:

term substitution:	$M[N/x]$	(N is substituted for x in M, avoiding capture)
structural substitution:	$T[\alpha \Leftarrow L]$	(every subterm $[\alpha]N$ of M is replaced by $[\alpha]NL$)

where $M, N, L \in \mathsf{Trm}$, $C \in \mathsf{Cmd}$ and $T \in \mathsf{Trm} \cup \mathsf{Cmd}$. More precisely, $T[\alpha \Leftarrow L]$ is defined by:

$$\begin{array}{rcll} ([\alpha]M)[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & [\alpha](M[\alpha \Leftarrow L])L & \\ ([\beta]M)[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & [\beta]M[\alpha \Leftarrow L] & \text{if } \alpha \neq \beta \\ (\mu\beta.C)[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & \mu\beta.C[\alpha \Leftarrow L] & \\ x[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & x & \\ (\lambda x.M)[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & \lambda x.M[\alpha \Leftarrow L] & \\ (MN)[\alpha \Leftarrow L] & \stackrel{\Delta}{=} & (M[\alpha \Leftarrow L])(N[\alpha \Leftarrow L]) & \end{array}$$

Definition 1.3 (REDUCTION [20]) The reduction relation $M \to N$, where $M, N \in \mathsf{Trm}$, is defined as the compatible closure of the following rules:

$$\begin{array}{lrcll} (\beta): & (\lambda x.M)N & \to & M[N/x] & (\textit{logical reduction}) \\ (\mu): & (\mu\beta.C)N & \to & \mu\beta.C[\beta \Leftarrow N] & (\textit{structural reduction}) \end{array}$$

2 Characterisation of Strong Normalisation

In this section we will show that we can characterise strong normalisation for pure $\lambda\mu$-terms completely through a notion of intersection typeing which employs product types and a restricted use of the type ω.

2.1 The type system

As mentioned in the introduction, our characterisation can be carried out by means of a precisely tailored version of the type system we presented in [6]. The types of our system will be formed by means of the $\to$, $\times$, and $\wedge$ type constructors over a single base type ν.[1]

Definition 2.1 (TYPES) The sets $\mathcal{T}_D$ of *term types* and $\mathcal{T}_C$ of *continuation-stack types* are defined inductively by the following grammar, where ν is a type constant:

$$\begin{array}{lrcl} \mathcal{T}_D: & \delta & ::= & \nu \mid \omega\to\nu \mid \kappa\to\nu \mid \delta\wedge\delta \\ \mathcal{T}_C: & \kappa & ::= & \delta\times\omega \mid \delta\times\kappa \mid \kappa\wedge\kappa \end{array}$$

(we will call the types $\delta\times\omega$ and $\delta\times\kappa$ also *product types*). We define the set $\mathcal{T}$ of *types* as $\mathcal{T} = \mathcal{T}_D \cup \mathcal{T}_C$ and let σ, τ, ρ, etc. range over $\mathcal{T}$.

Notice that an important feature of our system is the absence of ω as a proper type (and, consequently, the absence of its corresponding typeing rule); notice that we have not removed ω completely, since it always occurs at the very end of any product type in order to represent the (unspecified) last part of a continuation stack.

Definition 2.2 The relations $\leq_D$ and $\leq_C$ are the least pre-orders over $\mathcal{T}_D$ and $\mathcal{T}_C$, respectively, such that:

$$\frac{}{\sigma\wedge\tau \leq_A \sigma} \qquad \frac{}{\sigma\wedge\tau \leq_A \tau} \qquad \frac{}{\nu \leq_D \omega\to\nu} \qquad \frac{}{\omega\to\nu \leq_D \nu} \qquad \frac{}{\delta_1\times\delta_2\times\omega \leq_C \delta_1\times\omega}$$

$$\frac{}{(\delta_1\times\omega)\wedge(\delta_2\times\kappa) \leq_C (\delta_1\wedge\delta_2)\times\kappa} \qquad \frac{}{(\delta_1\times\kappa_1)\wedge(\delta_2\times\kappa_2) \leq_C (\delta_1\wedge\delta_2)\times(\kappa_1\wedge\kappa_2)}\ (\kappa_1,\kappa_2 \not\equiv \omega)$$

$$\frac{\delta_1 \leq_D \delta_2}{\delta_1\times\omega \leq_C \delta_2\times\omega} \qquad \frac{\delta_1 \leq_D \delta_2 \quad \kappa_1 \leq_C \kappa_2}{\delta_1\times\kappa_1 \leq_C \delta_2\times\kappa_2} \qquad \frac{\sigma \leq_A \tau_1 \quad \sigma \leq_A \tau_2}{\sigma \leq_A \tau_1\wedge\tau_2} \qquad \frac{\kappa_2 \leq_C \kappa_1}{\kappa_1\to\nu \leq_D \kappa_2\to\nu}$$

where A is either D or C. As usual, we define $=_A \stackrel{\Delta}{=} \leq_A \cap \geq_A$.

For convenience of notation, in the following the subscripts D and C on $\leq$ are normally omitted.

The pre-orders in Definition 2.2 are a restriction to $\mathcal{T}$ of the pre-orders defined in [6]. We point out that, in the system defined in that paper, the inequality $\delta_1\times\delta_2\times\omega \leq \delta_1\times\omega$ is derivable. In fact, in [6] we had $\omega =_C \omega\times\omega$ and hence $\delta_1\times\delta_2\times\omega \leq \delta_1\times\omega\times\omega = \delta_1\times\omega$. In the present

[1] In [6], more base types are used, but for our present purposes one suffices.

system, instead, $\omega \notin \mathcal{T}_D$ so that $\delta_1 \times \omega \times \omega \notin \mathcal{T}_C$, and therefore this inequality has to be explicitly postulated above.

The notions of *basis* (*variable context*), denoted by Γ, Γ', ..., and *name context*, denoted by Δ, Δ', ..., are defined in the standard way as, respectively, mappings of a finite set of term variables to types in $\mathcal{T}_D$, and of a finite set of names to types in $\mathcal{T}_C$, represented for convenience as sets of statements on variables and names (we call these assumptions). Below we shall write $\Gamma, x{:}\delta$ for $\Gamma \cup \{x{:}\delta\}$ where $x \notin dom(\Gamma)$; similarly for $\alpha{:}\kappa, \Delta$ (note that the order in which variable and name assumptions are listed in the rules is immaterial).

Definition 2.3 (Typeing System) *i*) A *judgement* in our system has the form $\Gamma \vdash M{:}\delta \mid \Delta$, where Γ is a basis, $M \in \mathsf{Trm}$, $\delta \in \mathcal{T}_D$ and Δ is a name context.

ii) We define *typeing* for pure $\lambda\mu$-terms (in Trm) through the following natural deduction system:

$$(ax): \ \frac{}{\Gamma, x{:}\delta \vdash x{:}\delta \mid \Delta} \qquad (\mu): \ \frac{\Gamma \vdash M{:}\kappa' \to \nu \mid \alpha{:}\kappa, \Delta}{\Gamma \vdash \mu\alpha.[\beta]M{:}\kappa \to \nu \mid \beta{:}\kappa', \Delta} \quad \frac{\Gamma \vdash M{:}\kappa \to \nu \mid \alpha{:}\kappa, \Delta}{\Gamma \vdash \mu\alpha.[\alpha]M{:}\kappa \to \nu \mid \Delta}$$

$$(abs): \ \frac{\Gamma, x{:}\delta \vdash M{:}\kappa \to \nu \mid \Delta}{\Gamma \vdash \lambda x.M{:}\delta \times \kappa \to \nu \mid \Delta} \qquad (app): \ \frac{\Gamma \vdash M{:}\delta \times \kappa \to \nu \mid \Delta \quad \Gamma \vdash N{:}\delta \mid \Delta}{\Gamma \vdash MN{:}\kappa \to \nu \mid \Delta}$$

$$(\leq): \ \frac{\Gamma \vdash M{:}\delta \mid \Delta}{\Gamma \vdash M{:}\delta' \mid \Delta}\ (\delta \leq \delta') \qquad (\wedge): \ \frac{\Gamma \vdash M{:}\delta \mid \Delta \quad \Gamma \vdash M{:}\delta' \mid \Delta}{\Gamma \vdash M{:}\delta \wedge \delta' \mid \Delta}$$

where κ in rules (abs) and (app)[2] is either a type in $\mathcal{T}_C$ or ω.

iii) We write $\Gamma \vdash M{:}\delta \mid \Delta$ whenever there exists a derivation built using the above rules that has this judgement in the bottom line, and $\mathcal{D} :: \Gamma \vdash M{:}\delta \mid \Delta$ when we want to name that derivation. We write $\vdash M{:}\delta \mid \Delta$ when the variable context is empty, and $\Gamma \vdash M{:}\delta \mid$ when the name context is.

Note that we use a single name, (μ), for the two rules concerning μ-abstraction; which is the one actually used will always be clear from the context.

We extend Barendregt's convention to judgements $\Gamma \vdash M{:}\delta \mid \Delta$ by seeing the variables that occur in Γ and names in Δ as binding occurrences over M as well; in particular, we can assume that no variable in Γ and no name in Δ is bound in M.

Definition 2.4 *i*) The relation $\leq$ is naturally extended to bases as follows:

$$\Gamma' \leq \Gamma \ \ \textit{iff} \ \ x{:}\delta \in \Gamma \Rightarrow \exists\, x{:}\delta' \in \Gamma' \,[\delta' \leq \delta].$$

The $\leq$ relation on name contexts is defined in the same way.

ii) Given two bases Γ_1 and Γ_2, we define the basis $\Gamma_1 \wedge \Gamma_2$ as follows:

$$\begin{array}{rcl} \Gamma_1 \wedge \Gamma_2 & \triangleq & \{x{:}\delta_1 \wedge \delta_2 \mid x{:}\delta_1 \in \Gamma_1 \ \&\ x{:}\delta_2 \in \Gamma_2\} \ \cup \\ & & \{x{:}\delta \mid x{:}\delta \in \Gamma_1 \ \&\ x \notin dom(\Gamma_2)\} \ \cup \\ & & \{x{:}\delta \mid x{:}\delta \in \Gamma_2 \ \&\ x \notin dom(\Gamma_1)\} \end{array}$$

iii) The name context $\Delta_1 \wedge \Delta_2$ is constructed out of Δ_1 and Δ_2 in a similar way.

[2] We use (app) and (abs) to name the rules concerning λ-abstraction and application, rather than the more usual $(\to I)$ and $(\to E)$, since in our system there is no introduction or elimination of the $\to$ type constructor.

Trivially, $dom(\Gamma_1 \wedge \Gamma_2) = dom(\Gamma_1) \cup dom(\Gamma_2)$ and $dom(\Delta_1 \wedge \Delta_2) = dom(\Delta_1) \cup dom(\Delta_2)$. Moreover, it is straightforward to show that:

Proposition 2.5 $\Gamma_1 \wedge \Gamma_2 \leq \Gamma_i$ *and* $\Delta_1 \wedge \Delta_2 \leq \Delta_i$ *for* $i = 1,2$.

We can also show that *Weakening* and *Strengthening* rules are implied by the system:

Lemma 2.6 (Weakening and Strengthening) *The following rules are admissible*[3]*:*

$$(W): \ \frac{\Gamma \vdash M:\delta \mid \Delta}{\Gamma' \vdash M:\delta \mid \Delta'} \ (\Gamma' \leq \Gamma, \Delta' \leq \Delta)$$

$$(S): \ \frac{\Gamma \vdash M:\delta \mid \Delta}{\Gamma' \vdash M:\delta \mid \Delta'} \ (\Gamma' = \{x{:}\delta \in \Gamma \mid x \in fv(M)\},\ \Delta' = \{\alpha{:}\kappa \in \Delta \mid \alpha \in fn(M)\})$$

The above lemma and Proposition 2.5 lead immediately to the following:

Corollary 2.7 *If* $\Gamma_1 \vdash M:\delta \mid \Delta_1$ *then for any* Γ_2, Δ_2: $\Gamma_1 \wedge \Gamma_2 \vdash M:\delta \mid \Delta_1 \wedge \Delta_2$.

Notice that, by Barendregt's convention, the variables in Γ_2 and names in Δ_2 are not bound in M.

The following substitution results can be proved along the lines of similar ones in [6]:

Lemma 2.8 (Substitution Lemma) *i)* $\Gamma \vdash M[N/x]:\delta \mid \Delta$ *with* $x \in fv(M)$*, if and only if there exists* δ' *such that* $\Gamma \vdash N:\delta' \mid \Delta$ *and* $\Gamma, x{:}\delta' \vdash M:\delta \mid \Delta$.

ii) $\Gamma \vdash M[\alpha \Leftarrow L]:\delta \mid \alpha{:}\kappa, \Delta$ *with* $\alpha \in fn(M)$*, if and only if there exists* δ' *such that* $\Gamma \vdash L:\delta' \mid \Delta$*, and* $\Gamma \vdash M:\delta \mid \alpha{:}\delta' \times \kappa, \Delta$.

2.2 Typeability implies Strong Normalisation

In this subsection we will show that – as can be expected of a well-defined notion of type assignment that does not type recursion and has no general rule that types all terms – all typeable terms are strongly normalising. Such a property does not hold for the system in [6] where, in fact, by means of types not allowed in the present system, it is possible to type the fixed-point constructor $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ in a non-trivial way, as shown by the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f{:}\omega\times\omega\to\nu, x{:}\omega \vdash f:\omega\times\omega\to\nu \mid}(ax) \quad \dfrac{}{f{:}\omega\times\omega\to\nu, x{:}\omega \vdash xx:\omega \mid}(\omega)}{f{:}\omega\times\omega\to\nu, x{:}\omega \vdash f(xx):\omega\to\nu \mid}(app)}{f{:}\omega\times\omega\to\nu \vdash \lambda x.f(xx):\omega\times\omega\to\nu \mid}(abs) \quad \dfrac{}{f{:}\omega\times\omega\to\nu \vdash \lambda x.f(xx):\omega \mid}(\omega)}{f{:}\omega\times\omega\to\nu \vdash (\lambda x.f(xx))(\lambda x.f(xx)):\omega\to\nu \mid}(app)}{\vdash \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)):(\omega\times\omega\to\nu)\times\omega\to\nu \mid}(abs)$$

[3] We should perhaps point out that Barendregt's convention, extended to judgements as we do here, is essential for the correctness of this result. By writing $\Gamma' \vdash M:\delta \mid \Delta'$, we assume that Γ' and Δ' do not contain statements for variables and names that occur bound in M, so we do not allow contexts to be weakened by statements concerning bound names or variables. As a counter example, take $\vdash \mu\alpha.[\alpha]\lambda x.x:(\kappa\to\nu)\to\kappa\to\nu \mid$ and $\Gamma_2 = x{:}\delta$, $\Delta_2 = \alpha{:}\kappa$; we cannot derive $x{:}\delta \vdash \mu\alpha.[\alpha]\lambda x.x:(\kappa\to\nu)\to\kappa\to\nu \mid \alpha{:}\kappa$.

This is also the case for systems for the λ-calculus; in past papers it has been claimed that, if $\Gamma_1 \vdash_\lambda M : A$ and $\Gamma_2 \vdash_\lambda N : B$ (without any restrictions), then also $\Gamma_1 \wedge \Gamma_2 \vdash M:A$ and $\Gamma_1 \wedge \Gamma_2 \vdash N:B$. This is incorrect for the same reason: take $\vdash_\lambda \lambda y.y : A\to A$ and $y{:}(A\to A)\wedge A\to A \vdash yy:A$; we *cannot* derive $y{:}(A\to A)\wedge A\to A \vdash_\lambda \lambda y.y : A\to A$.

Notice that this term does not have a normal form, so is not strongly normalisable.

Definition 2.9 The set $\mathcal{SN}$ of *strongly normalisable* terms is defined as usual as the set of all terms M such that no infinite reduction sequence out of M exists; we use $\mathcal{SN}(M)$ for $M \in \mathcal{SN}$, and $\mathcal{SN}^*$ for the set of finite stacks of terms in $\mathcal{SN}$.

The following is straightforward:

Proposition 2.10 *i) If $\mathcal{SN}(x\vec{M})$ and $\mathcal{SN}(\vec{N})$, then $\mathcal{SN}(x\vec{M}\vec{N})$.*
ii) If $\mathcal{SN}(M[N/x]\vec{P})$ and $\mathcal{SN}(N)$, then $\mathcal{SN}((\lambda x.M)N\vec{P})$.
iii) If $\mathcal{SN}(M)$, then $\mathcal{SN}(\mu\alpha.[\beta]M)$.
iv) If $\mathcal{SN}(\mu\alpha.[\beta]M[\alpha \Leftarrow \vec{N}]\vec{L})$ and $\mathcal{SN}(\vec{N})$, then $\mathcal{SN}((\mu\alpha.[\beta]M)\vec{N}\vec{L})$.
v) If $\mathcal{SN}(\mu\alpha.[\alpha]M[\alpha \Leftarrow \vec{N}]\vec{N}\vec{L})$, then $\mathcal{SN}((\mu\alpha.[\alpha]M)\vec{N}\vec{L})$.

Definition 2.11 (Type Interpretation) *i*) We define a map

$$[\![\cdot]\!] : \ (\mathcal{T}_D \to \wp(\mathsf{Trm})) + (\mathcal{T}_C \to \wp(\mathsf{Trm}^*))$$

(where $\wp$ represents the powerset constructor) interpreting term types and continuation-stack types as, respectively, sets of terms and sets of stacks, as follows:

$$\begin{array}{rcl} [\![\nu]\!] = [\![\omega{\to}\nu]\!] & = & \mathcal{SN} \\ [\![\kappa{\to}\nu]\!] & = & \{\, M \in \mathsf{Trm} \mid \forall \vec{L} \in [\![\kappa]\!]\, [\, M\vec{L} \in [\![\nu]\!]\,]\,\} \\ [\![\delta{\times}\omega]\!] & = & \{\, N : \vec{L} \mid N \in [\![\delta]\!], \vec{L} \in \mathcal{SN}^*\,\} \\ [\![\delta{\times}\kappa]\!] & = & \{\, N : \vec{L} \mid N \in [\![\delta]\!], \vec{L} \in [\![\kappa]\!]\,\} \\ [\![\sigma{\wedge}\tau]\!] & = & [\![\sigma]\!] \cap [\![\tau]\!] \end{array}$$

ii) We define the *length* of a stack type, $|\cdot| : \mathcal{T}_C \to \mathsf{N}$, as follows:

$$\begin{array}{rcl} |\delta{\times}\omega| & = & 1 \\ |\delta{\times}\kappa| & = & 1 + |\kappa| \\ |\kappa_1{\wedge}\kappa_2| & = & max\ |\kappa_1|\ |\kappa_2| \end{array}$$

By this interpretation, the elements of $[\![\delta_1 \times \cdots \times \delta_n \times \omega]\!]$ are stacks of strongly normalisable terms that have an arbitrary length greater than or equal to n. It is easy to check that $|\kappa|$ returns the minimal length of the stacks in $[\![\kappa]\!]$.

We can show:

Lemma 2.12 *For any $\delta \in \mathcal{T}_D$ and $\kappa \in \mathcal{T}_C$:*
i) $[\![\delta]\!] \subseteq \mathcal{SN}$ and $[\![\kappa]\!] \subseteq \mathcal{SN}^*$.
ii) $x\vec{N} \in \mathcal{SN} \Rightarrow x\vec{N} \in [\![\delta]\!]$.
iii) $\vec{x} = x_1 : \ldots : x_n \in [\![\kappa]\!]$, for all n such that $n \geq |\kappa|$.

Proof: By simultaneous induction on the structure of types, using Definition 2.11. We show some of the cases.

$$\begin{array}{lll} i)(\kappa{\to}\nu): & M \in [\![\kappa{\to}\nu]\!] & \Rightarrow (IH((ii))) \\ & \vec{x} \in [\![\kappa]\!]\ \&\ M \in [\![\kappa{\to}\nu]\!] & \Rightarrow (2.11) \\ & M\vec{x} \in [\![\nu]\!] & \Rightarrow (2.11) \\ & M\vec{x} \in \mathcal{SN} & \Rightarrow M \in \mathcal{SN}. \end{array}$$

$$
\begin{array}{lll}
(\delta\times\omega): & M\in[\![\delta\times\omega]\!] & \Rightarrow (2.11))\\
& M=N:\vec{L}\ \&\ N\in[\![\delta]\!]\ \&\ \vec{L}\in\mathcal{SN} & \Rightarrow (IH((i)))\\
& N\in\mathcal{SN}\ \&\ \vec{L}\in\mathcal{SN}^* & \Rightarrow N:\vec{L}\in\mathcal{SN}^*.\\
(\delta\times\kappa): & M\in[\![\delta\times\kappa]\!] & \Rightarrow (2.11))\\
& M=N:\vec{L}\ \&\ N\in[\![\delta]\!]\ \&\ \vec{L}\in[\![\kappa]\!] & \Rightarrow (IH((i)))\\
& N\in\mathcal{SN}\ \&\ \vec{L}\in\mathcal{SN}^* & \Rightarrow N:\vec{L}\in\mathcal{SN}^*.\\
ii)(\kappa\to\nu): & x\vec{N}\in\mathcal{SN} & \Rightarrow (2.11\ \&\ IH((i)))\\
& \vec{L}\in[\![\kappa]\!]\Rightarrow x\vec{N}\in\mathcal{SN}\ \&\ \vec{L}\in\mathcal{SN}^* & \Rightarrow (2.10)\\
& \vec{L}\in[\![\kappa]\!]\Rightarrow x\vec{N}\vec{L}\in\mathcal{SN} & \Rightarrow (IH((ii)))\\
& \vec{L}\in[\![\kappa]\!]\Rightarrow x\vec{N}\vec{L}\in[\![\nu]\!] & \Rightarrow (2.11)\qquad x\vec{N}\in[\![\kappa\to\nu]\!].\\
iii)(\delta\times\omega): & \vec{x}=x:\vec{x}' & \Rightarrow (IH((ii)))\\
& x\in[\![\delta]\!]\ \&\ \vec{x}'\in\mathcal{SN}^* & \Rightarrow (2.11)\qquad \vec{x}\in[\![\delta\times\omega]\!].\\
(\delta\times\kappa): & \vec{x}=x:\vec{x}' & \Rightarrow (IH((ii)))\ \&\ IH((iii)))\\
& x\in[\![\delta]\!]\ \&\ \vec{x}'\in[\![\kappa]\!] & \Rightarrow (2.11)\qquad \vec{x}\in[\![\delta\times\kappa]\!].
\end{array}
$$

■

The following result follows immediately from Lemma 2.12 ((*ii*)):

Corollary 2.13 For any $x\in Var$ and any $\delta\in\mathcal{T}_D$: $x\in[\![\delta]\!]$.

The following lemma shows that our type interpretation is closed under the type inclusion relation.

Lemma 2.14 For all $\sigma,\tau\in\mathcal{T}$: if $\sigma\leq\tau$, then $[\![\sigma]\!]\subseteq[\![\tau]\!]$.

Proof: By induction on the definition of $\leq$. We show some of relevant cases.

$$
\begin{array}{lll}
(\delta_1\times\omega)\wedge(\delta_2\times\kappa)\leq(\delta_1\wedge\delta_2)\times\kappa): [\![(\delta_1\times\omega)\wedge(\delta_2\times\kappa)]\!] & = & .\\
\{M:\vec{L}\mid M\in[\![\delta_1]\!],\vec{L}\in\mathcal{SN}^*\}\cap\{M:\vec{L}\mid M\in[\![\delta_2]\!],\vec{L}\in[\![\kappa]\!]\} & = & ([\![\kappa]\!]\subseteq\mathcal{SN}^*\text{ by 2.12 ((}i\text{))})\\
\{M:\vec{L}\mid M\in[\![\delta_1]\!]\cap[\![\delta_2]\!],\vec{L}\in[\![\kappa]\!]\} & = & \\
\{M:\vec{L}\mid M\in[\![\delta_1\wedge\delta_2]\!],\vec{L}\in[\![\kappa]\!]\} & = & \\
[\![(\delta_1\wedge\delta_2)\times\kappa]\!] & &
\end{array}
$$

$$
\begin{array}{lll}
(\kappa_2\leq\kappa_1\Rightarrow\kappa_1\to\nu\leq\kappa_2\to\nu): [\![\kappa_1\to\nu]\!] & = & \\
\{M\in\mathsf{Trm}\mid\forall\vec{L}\in[\![\kappa_1]\!]\,[M\vec{L}\in\mathcal{SN}]\} & \subseteq & ([\![\kappa_2]\!]\subseteq[\![\kappa_1]\!]\text{ by induction})\\
\{M\in\mathsf{Trm}\mid\forall\vec{L}\in[\![\kappa_2]\!]\,[M\vec{L}\in\mathcal{SN}]\} & = & \\
[\![\kappa_2\to\nu]\!] & &
\end{array}
$$

■

Our type interpretation is closed under expansion for the logical and for the structural reduction, with the proviso that the term or stack to be substituted is an element of an interpreted type as well.

Lemma 2.15 For any $\delta,\delta'\in\mathcal{T}_D$ and $\kappa\in\mathcal{T}_C$:

i) If $M[N/x]\vec{P}\in[\![\delta]\!]$ and $N\in[\![\delta']\!]$, then $(\lambda x.M)N\vec{P}\in[\![\delta]\!]$.

ii) If $\mu\alpha.[\beta]M[\alpha\Leftarrow\overrightarrow{N}]\vec{P}\in[\![\delta]\!]$ and $\overrightarrow{N}\in[\![\kappa]\!]$, then $(\mu\alpha.[\beta]M)\overrightarrow{N}\vec{P}\in[\![\delta]\!]$.

iii) If $\mu\alpha.[\alpha]M[\alpha\Leftarrow\overrightarrow{N}]\overrightarrow{N}\vec{P}\in[\![\delta]\!]$, then $(\mu\alpha.[\alpha]M)\overrightarrow{N}\vec{P}\in[\![\delta]\!]$.

Proof: By induction on the structure of types, using 2.10, 2.11 and 2.12. ■

In Theorem 2.18 we will show that all typeable terms are strongly normalisable. In order to achieve that, we first show, in Lemma 2.17, that for any a term M typeable with δ, any full

substitution instance M_ξ (i.e. replacing all free term variables by terms, and feeding stacks to all free names) is an element of the interpretation of δ, which by Lemma 2.12 implies that M_ξ is strongly normalisable. We need these substitutions to be applied all 'in one go', so define a notion of parallel substitution. The main result is then obtained by taking the substitution that replaces term variables by themselves and names by stacks of term variables. The reason we first prove the result for *any* substitution is that, in the proof of Lemma 2.17, in the case for $\lambda x.M$ and $\mu\alpha.Q$ the substitution is extended, by replacing the bound variable or name with a normal term (or stack).

Definition 2.16 *i*) A partial mapping $\xi : (Var \to \mathsf{Trm}) + (Name \to \mathsf{Trm}^*)$ is a *parallel substitution* if, for every $p,q \in dom(\xi)$, if $p \neq q$ then $p \notin fv(\xi q)$ and $p \notin fn(\xi q)$.

ii) Borrowing a notation for valuations, for a parallel substitution ξ we define the application of ξ to a term by:

$$\begin{array}{rcll}
([\alpha]M)_\xi & \triangleq & [\alpha]M_\xi\vec{L} & \text{if } \xi\alpha = \vec{L} \\
([\beta]M)_\xi & \triangleq & [\beta]M_\xi & \text{if } \beta \notin dom(\xi) \\
(\mu\beta.Q)_\xi & \triangleq & \mu\beta.Q_\xi & \\
x_\xi & \triangleq & N & \text{if } \xi x = N \\
y_\xi & \triangleq & y & \text{if } y \notin dom(\xi) \\
(\lambda x.M)_\xi & \triangleq & \lambda x.M_\xi & \\
(MN)_\xi & \triangleq & M_\xi N_\xi &
\end{array}$$

iii) We define $\xi[N/x]$ and $\xi[\alpha \Leftarrow \vec{L}]$ by, respectively,

$$\xi[N/x]\,y \triangleq \begin{cases} N & \text{if } y = x \\ \xi\, y & \text{otherwise} \end{cases}$$

$$\xi[\alpha \Leftarrow \vec{L}]\,\beta \triangleq \begin{cases} \vec{L} & \text{if } \alpha = \beta \\ \xi\,\beta & \text{otherwise} \end{cases}$$

iv) We will say that ξ *extends* Γ *and* Δ, if, for all $x{:}\delta \in \Gamma$ and $\alpha{:}\kappa \in \Delta$, we have, respectively, $\xi(x) \in [\![\delta]\!]$ and $\xi(\alpha) \in [\![\kappa]\!]$.

Notice that we do allow a variable to appear in its own image under ξ. Since x does not appear in $M[N/x]$, this does not violate Barendregt's convention.

Lemma 2.17 (Replacement Lemma) *Let ξ be a parallel substitution that extends Γ and Δ. Then:*

$$\text{if } \ \Gamma \vdash M{:}\delta \mid \Delta \ \text{ then } \ M_\xi \in [\![\delta]\!].$$

Proof: By induction on the structure of derivations. We show some more illustrative cases.

(abs) : Then $M = \lambda x.M'$, $\delta = \delta' \times \kappa \to \nu$, and $\Gamma, x{:}\delta' \vdash M' : \kappa \to \nu \mid \Delta$. Take $N \in [\![\delta']\!]$; since x is bound, by Barendregt's convention we can assume that it does not occur free in the image of ξ, so $\xi[N/x]$ is a well-defined parallel substitution that extends $\Gamma, x{:}\delta'$ and Δ. Then by induction, we have $M'_{\xi[N/x]} \in [\![\kappa \to \nu]\!]$. Since x does not occur free in the image of ξ, $M'_{\xi[N/x]} = M'_\xi[N/x]$, so also $M'_\xi[N/x] \in [\![\kappa \to \nu]\!]$. By Lemma 2.15 ((*i*)), also $(\lambda x.M'_\xi)N \in [\![\kappa \to \nu]\!]$. By definition of $[\![\kappa \to \nu]\!]$, for any $\vec{L} \in [\![\kappa]\!]$ we have $(\lambda x.M'_\xi)N\vec{L} \in [\![\nu]\!]$; notice that $N : \vec{L} \in [\![\delta \times \kappa]\!]$, so $(\lambda x.M')_\xi \in [\![\delta' \times \kappa \to \nu]\!]$.

(μ) : Then $M = \mu\alpha.[\beta]M'$, and $\delta = \kappa \to \nu$. We distinguish two different sub-cases.

$\alpha = \beta$: Then $M = \mu\alpha.[\alpha]M'$, $\delta = \kappa \to \nu$, and $\Gamma \vdash M' : \kappa \to \nu \mid \alpha{:}\kappa, \Delta$. Take $\vec{L} \in [\![\kappa]\!]$; since α is bound in M, we can assume it does not occur free in the image of ξ, so $\xi[\alpha \Leftarrow \vec{L}]$ is a

well-defined parallel substitution that extends Γ and $\Delta,\alpha{:}\kappa$, and by induction, $M'_{\xi[\alpha \Leftarrow L]} \in [\![\kappa{\to}\nu]\!]$. Since α does not occur free in the image of ξ, $M'_{\xi[\alpha \Leftarrow L]} = M'_{\xi}[\alpha \Leftarrow \vec{L}]$, so we have $M'_{\xi}[\alpha \Leftarrow \vec{L}] \in [\![\kappa{\to}\nu]\!]$, and therefore $M'_{\xi}[\alpha \Leftarrow \vec{L}]\vec{L} \in [\![\nu]\!]$.

Then by Definition 2.11, $\mathcal{SN}(M'_{\xi}[\alpha \Leftarrow \vec{L}]\vec{L})$, but then also $\mathcal{SN}(\mu\alpha.[\alpha]M'_{\xi}[\alpha \Leftarrow \vec{L}]\vec{L})$, by Lemma 2.10 ((*iii*)). So $\mu\alpha.[\alpha]M'_{\xi}[\alpha \Leftarrow \vec{L}]\vec{L} \in [\![\nu]\!]$. Then by Lemma 2.15 ((*iii*)), $(\mu\alpha.[\alpha]M'_{\xi})\vec{L} \in [\![\nu]\!]$; so $(\mu\alpha.[\alpha]M')_{\xi} \in [\![\kappa{\to}\nu]\!]$.

$\alpha \neq \beta$: Then $\Delta = \beta{:}\kappa',\Delta'$, and $\Gamma \vdash M' : \kappa'{\to}\nu \mid \alpha{:}\kappa,\beta{:}\kappa',\Delta$. Assume $\vec{L} \in [\![\kappa]\!]$, then $\xi[\alpha \Leftarrow \vec{L}]$ extends Γ and $\alpha{:}\kappa,\beta{:}\kappa',\Delta'$. Then, by induction, $M'_{\xi[\alpha \Leftarrow L]} \in [\![\kappa'{\to}\nu]\!]$. Now let $\vec{Q} \in [\![\kappa']\!]$, then $M'_{\xi[\alpha \Leftarrow L]}\vec{Q} \in [\![\nu]\!]$ and then also $(M'\vec{Q})_{\xi[\alpha \Leftarrow L]} \in [\![\nu]\!]$.

Then $\mathcal{SN}((M'\vec{Q})_{\xi[\alpha \Leftarrow L]})$ by Definition 2.11, and $\mathcal{SN}(\mu\alpha.[\beta](M'\vec{Q})_{\xi[\alpha \Leftarrow L]})$ by Lemma 2.10 ((*iii*)), so, again by Definition 2.11, $\mu\alpha.[\beta](M'\vec{Q})_{\xi[\alpha \Leftarrow L]} \in [\![\nu]\!]$. As in the previous part, α is not free in the image of ξ, and therefore also $\mu\alpha.[\beta](M'\vec{Q})_{\xi}[\alpha \Leftarrow \vec{L}] \in [\![\nu]\!]$. Then, by Lemma 2.15 ((*ii*)), $(\mu\alpha.[\beta](M'\vec{Q})_{\xi})\vec{L} \in [\![\nu]\!]$. Notice that $[\beta]M'_{\xi}\vec{Q} = [\beta]M'_{\xi}[\beta \Leftarrow \vec{Q}]$; since $\xi\beta = \vec{Q}$, we can infer that $[\beta]M'_{\xi}\vec{Q} = [\beta]M'_{\xi}$, so $(\mu\alpha.[\beta]M')_{\xi}\vec{L} \in [\![\nu]\!]$. But then $(\mu\alpha.[\beta]M')_{\xi} \in [\![\kappa{\to}\nu]\!]$. ■

We now come to the main result of this section, that states that all terms typeable in our system are strongly normalisable.

Theorem 2.18 (Typeable terms are $\mathcal{SN}$) *If $\Gamma \vdash M : \delta \mid \Delta$ for some Γ, Δ and δ, then $M \in \mathcal{SN}$.*

Proof: Let ξ be a parallel substitution such that

$$\begin{array}{rcll} \xi(x) & = & x & \text{for } x \in dom(\Gamma) \\ \xi(\alpha) & = & \vec{y}_{\alpha} & \text{for } \alpha \in dom(\Delta) \end{array}$$

where the length of the stack $\vec{y}_{\alpha}$ is $|\kappa|$ if $\alpha{:}\kappa \in \Delta$ (notice that ξ is well defined). By Lemma 2.12, ξ extends Γ and Δ. Hence, by Lemma 2.17, $M_{\xi} \in [\![\delta]\!]$, and then $M_{\xi} \in \mathcal{SN}$ by Lemma 2.12 ((*i*)). Now

$$\begin{array}{rcl} M_{\xi} & \equiv & M[x_1/x_1,\ldots,x_n/x_n,\alpha_1 \Leftarrow \vec{y}_{\alpha_1},\ldots,\alpha_m \Leftarrow \vec{y}_{\alpha_m}] \\ & \equiv & M[\alpha_1 \Leftarrow \vec{y}_{\alpha_1},\ldots,\alpha_m \Leftarrow \vec{y}_{\alpha_m}] \end{array}$$

Then, by Proposition 2.10, for any $\vec{\beta}$ also $(\mu\alpha_1.[\beta_1]\cdots\mu\alpha_m.[\beta_m]M)\vec{y}_{\alpha_1}\cdots\vec{y}_{\alpha_m} \in \mathcal{SN}$, and therefore also $M \in \mathcal{SN}$. ■

2.3 Strongly Normalising Terms are Typeable

In this section we will show the counterpart of the previous result, namely that all strongly normalisable terms are typeable in our intersection system.

First we describe the shape of the terms in normal form.

Definition 2.19 (Normal Forms) The set $\mathcal{N} \subseteq \mathsf{Trm}$ of *normal forms* is defined by the grammar:

$$N \ ::= \ xN_1\cdots N_k \mid \lambda x.N \mid \mu\alpha.[\beta]N$$

It is straightforward to verify that the terms in $\mathcal{N}$ are precisely the irreducible ones.

We can show that all terms in $\mathcal{N}$ are typeable.

Lemma 2.20 If $N \in \mathcal{N}$ then there exist Γ, Δ, and a type $\kappa{\to}\nu$ such that $\Gamma \vdash N : \kappa{\to}\nu \mid \Delta$.

Proof: By induction on the definition of $\mathcal{N}$. We show the most relevant cases.

$(N \equiv xN_1 \dots N_k)$: Since $N_1, \dots, N_k \in \mathcal{N}$, by induction we have that, for all $i \leq k$ there exist Γ_i, Δ_i and δ_i such that $\Gamma_i \vdash N_i : \delta_i \mid \Delta_i$ (the structure of each δ_i plays no role in this part). Take

$$\Gamma = \Gamma_1 \wedge \dots \wedge \Gamma_k \wedge x{:}(\delta_1 \times \dots \times \delta_k \times \delta \times \omega) \to \nu, \text{ and } \Delta = \Delta_1 \wedge \dots \wedge \Delta_k.$$

where δ is any element of $\mathcal{T}_D$. Then, by Lemma 2.6, $\Gamma \vdash N_i : \delta_i \mid \Delta$ for all $i \leq n$, and $\Gamma \vdash x : (\delta_1 \times \dots \times \delta_k \times \delta \times \omega) \to \nu \mid \Delta$. By repeated application of (app) we get $\Gamma \vdash xN_1 \cdots N_k : \kappa \to \nu \mid \Delta$ for $\kappa = \delta \times \omega$.

$(N \equiv \mu\alpha.[\beta]N')$: By induction, $\Gamma \vdash N' : \kappa \to \nu \mid \Delta$. We distinguish two cases:

$(\alpha \equiv \beta)$: In case $\alpha \in fn(N')$ and $\Delta = \alpha{:}\kappa', \Delta'$, we can construct:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash N' : \kappa \to \nu \mid \alpha{:}\kappa', \Delta'}{\Gamma \vdash N' : \kappa \to \nu \mid \alpha{:}\kappa \wedge \kappa', \Delta'}\,(W)}{\Gamma \vdash N' : \kappa \wedge \kappa' \to \nu \mid \alpha{:}\kappa \wedge \kappa', \Delta'}\,(\leq)}{\Gamma \vdash \mu\alpha.[\alpha]N' : \kappa \wedge \kappa' \to \nu \mid \Delta'}\,(\mu)$$

In case $\alpha \notin fn(N')$, we can construct

$$\dfrac{\dfrac{\Gamma \vdash N' : \kappa \to \nu \mid \Delta}{\Gamma \vdash N' : \kappa \to \nu \mid \alpha{:}\kappa, \Delta'}\,(W)}{\Gamma \vdash \mu\alpha.[\alpha]N' : \kappa \to \nu \mid \Delta'}\,(\mu)$$

$(\alpha \not\equiv \beta)$: We can proceed as in the previous case, obtaining now $\Gamma \vdash N : \kappa \to \nu \mid \alpha{:}\kappa', \beta{:}\kappa, \Delta'$. So by rule (μ) we get $\Gamma \vdash \mu\alpha.[\beta]N' : \kappa' \to \nu \mid \beta{:}\kappa'', \Delta'$. ■

We will now show that typeing is closed under expansion with respect to both logical and structural reduction, with the proviso that the term (stack) that gets substituted is typeable as well in the same contexts.

Lemma 2.21 (Contractum Expansion) *i) If $\Gamma \vdash M[N/x] : \delta \mid \Delta$ and $\Gamma \vdash N : \delta' \mid \Delta$ then $\Gamma \vdash (\lambda x.M)N : \delta \mid \Delta$.*

ii) If $\Gamma \vdash \mu\alpha.[\beta]M[\alpha \Leftarrow N] : \delta \mid \Delta$ and $\Gamma \vdash N : \delta' \mid \Delta$ then $\Gamma \vdash (\mu\alpha.[\beta]M)N : \delta \mid \Delta$.

Proof: *i*) Much the same as the similar result for the intersection systems for the λ-calculus.

ii) We need to consider two different cases:

$(\alpha \notin fn([\beta]M))$: Then $([\beta]M)[\alpha \Leftarrow N] \equiv [\beta]M$ and $\alpha \not\equiv \beta$. We consider all the n minimal sub-derivations ($n \geq 1$) having $\mu\alpha.[\beta]M$ as subject, from which conclusions we derive $\Gamma \vdash \mu\alpha.[\beta]M[\alpha \Leftarrow N] : \delta \mid \Delta$ by applying any number of $(\leq)$ and $(\wedge)$ rules.
The last step in each of these derivations is of the shape:

$$\dfrac{\Gamma \vdash M : \kappa \to \nu \mid \alpha{:}\kappa_i, \beta{:}\kappa, \Delta'}{\Gamma \vdash \mu\alpha.[\beta]M : \kappa_i \to \nu \mid \beta{:}\kappa, \Delta'}\,(\mu)$$

where $\Delta = \beta{:}\kappa, \Delta'$. Since $\alpha \notin fn([\beta]M)$, by strengthening (Lemma 2.6) we can remove $\alpha{:}\kappa_i$ from the name context, so also $\Gamma \vdash M : \kappa \to \nu \mid \beta{:}\kappa, \Delta'$; then, by weakening (Lemma 2.6), we can add $\alpha{:}\delta' \times \kappa_i,$, so $\Gamma \vdash M : \kappa \to \nu \mid \alpha{:}\delta' \times \kappa_i, \beta{:}\kappa, \Delta'$, and then we can

construct

$$\dfrac{\dfrac{\boxed{}}{\Gamma \vdash \mu\alpha.[\beta]M : \delta' \times \kappa_i \to \nu \mid \beta : \kappa, \Delta'}\,(\mu) \qquad \boxed{}}{\Gamma \vdash (\mu\alpha.[\beta]M)N : \kappa_i \to \nu \mid \Delta}\,(app)$$

from which it is possible to derive $\Gamma \vdash (\mu\alpha.[\beta]M)N : \delta \mid \Delta$ by applying the same $(\leq)$ and $(\wedge)$ rules mentioned above.

$(\alpha \in fn([\beta]M))$: We distinguish two further cases:

$(\alpha = \beta)$: Then $([\beta]M)[\alpha \Leftarrow N] \equiv ([\alpha]M)[\alpha \Leftarrow N] \equiv [\alpha](M[\alpha \Leftarrow N])N$; we can assume, without loss of generality, that $\delta = (\kappa_1 \to \nu) \wedge \cdots \wedge (\kappa_n \to \nu)$, and that for all $i \leq n$ there are sub-derivations constructed like

$$\dfrac{\dfrac{\boxed{} \qquad \boxed{}}{\Gamma \vdash (M[\alpha \Leftarrow N])N : \kappa_i \to \nu \mid \alpha : \kappa_i, \Delta}\,(app)}{\Gamma \vdash \mu\alpha.[\alpha](M[\alpha \Leftarrow N])N : \kappa_i \to \nu \mid \Delta}\,(\mu)$$

Then there exists δ_i' such that $\Gamma \vdash N : \delta_i' \mid \Delta$, and $\Gamma \vdash M : \delta_i \times \kappa_i \to \nu \mid \alpha : \delta_i' \times \kappa_i, \Delta$ by Lemma 2.8 ((*ii*)); so we can build the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\boxed{}}{\Gamma \vdash M : \delta_i \wedge \delta_i' \times \kappa_i \to \nu \mid \alpha : \delta_i' \times \kappa_i, \Delta}\,(\leq)}{\Gamma \vdash M : \delta_i \wedge \delta_i' \times \kappa_i \to \nu \mid \alpha : \delta_i \wedge \delta_i' \times \kappa_i, \Delta}\,(W)}{\Gamma \vdash \mu\alpha.[\alpha]M : \delta_i \wedge \delta_i' \times \kappa_i \to \nu \mid \Delta}\,(\mu) \qquad \dfrac{\boxed{} \qquad \boxed{}}{\Gamma \vdash N : \delta_i \wedge \delta_i' \mid \Delta}\,(\wedge)}{\Gamma \vdash (\mu\alpha.[\alpha]M)N : \kappa_i \to \nu \mid \Delta}\,(app)$$

We derive $\Gamma \vdash (\mu\alpha.[\alpha]M)N : \delta \mid \Delta$ by rule $(\wedge)$.

$(\alpha \neq \beta)$: Then $([\beta]M)[\alpha \Leftarrow N] \equiv [\beta](M[\alpha \Leftarrow N])$; as above $\delta = (\kappa_1 \to \nu) \wedge \cdots \wedge (\kappa_n \to \nu)$, and for all $i \leq n$ there are derivations structured like:

$$\dfrac{\boxed{}}{\Gamma \vdash \mu\alpha.[\beta](M[\alpha \Leftarrow N]) : \kappa_i \to \nu \mid \beta : \kappa_i', \Delta'}\,(\mu)$$

where $\Delta = \beta : \kappa_i', \Delta'$. As above, by Lemma 2.8 ((*ii*)) there exists δ_i such that both $\Gamma \vdash N : \delta_i \mid \beta : \kappa_i', \Delta'$ and $\Gamma \vdash M : \kappa_i' \to \nu \mid \alpha : \delta_i \times \kappa_i, \beta : \kappa_i', \Delta'$. We can then construct:

$$\dfrac{\dfrac{\boxed{}}{\Gamma \vdash \mu\alpha.[\beta]M : \delta_i \times \kappa_i \to \nu \mid \Delta}\,(\mu) \qquad \boxed{}}{\Gamma \vdash (\mu\alpha.[\beta]M)N : \kappa_i \to \nu \mid \Delta}\,(app)$$

As above, we conclude that $\Gamma \vdash (\mu\alpha.[\beta]M)N : \delta \mid \Delta$ by rule $(\wedge)$. ∎

We will now show that all strongly normalisable terms are typeable in our system. The proof of the crucial lemma for this result as presented below (Lemma 2.23) goes by induction

on the left-most outer-most reduction path.

Definition 2.22 An occurrence of a redex $\mathtt{R} = (\lambda x.P)Q$ or $(\mu\alpha.[\beta]P)Q$ in a term M is called the *left-most outer-most redex of* M ($lor(M)$), if and only if:

i) there is no redex $\mathtt{R}'$ in M such that $\mathtt{R}' = \mathtt{C}[\mathtt{R}]$ (*outer-most*);

ii) there is no redex $\mathtt{R}'$ in M such that $M = \mathtt{C}_0[\mathtt{C}_1[\mathtt{R}']\,\mathtt{C}_2[\mathtt{R}]]$ (*left-most*).

$M \to_{lor} N$ is used to indicate that M reduces to N by contracting $lor(M)$.

The following lemma formulates a subject expansion result for our system with respect to left-most outer-most reduction. A proof for this property in the context of strict intersection type assignment for the λ-calculus appeared in [3, 5].

Lemma 2.23 Let $M \to_{lor} N$, $lor(M) = RQ$, $\Gamma_1 \vdash N:\delta_1 \mid \Delta_1$ with δ_1 not an intersection, and $\Gamma_2 \vdash Q:\delta_2 \mid \Delta_2$, then there exist Γ_3, Δ_3 and δ_3 such that $\Gamma_3 \leq \Gamma_1$, $\Delta_3 \leq \Delta_1$, $\delta_1 \leq \delta_3$, and $\Gamma_3 \vdash M:\delta_3 \mid \Delta_3$.

Proof: By induction on the structure of terms.

$M = VP_1 \cdots P_n$: Then either:

a) V is a redex $(\lambda y.P)Q$, so $lor(M) = V$; let $V' \equiv P[Q/y]$; or

b) V is a redex $(\mu\alpha.[\beta]P)Q$, so $lor(M) = V$; let $V' \equiv \mu\alpha.[\beta]P[\alpha \Leftarrow Q]$; or

c) $V \equiv z$ and there is an $i \in \underline{n}$ such that $lor(M) = lor(P_j)$, $N \equiv zP_1 \cdots P'_i \cdots P_n$, and $P_i \to_{lor} P'_i$; let $V' = z$.

By assumption $\delta_1 = \kappa_1 \to \nu$. Then there are δ_j ($j \in \underline{n}$), such that $\Gamma_1 \vdash V':\delta'_1 \times \cdots \times \delta'_n \times \kappa_1 \to \nu \mid \Delta_1$ and $\Gamma_1 \vdash P_i:\delta'_i \mid \Delta_1$, for all $i \in \underline{n}$.

We distinguish:

a) $V' \equiv P[Q/y]$, where the substitution is capture avoiding, so all free variables in Q are free in $P[Q/y]$ when $y \in fv(P)$, and we can assume that Γ_2 and Δ_2 do not have types for bound variables and names in P. Let $\Gamma_3 = \Gamma_1 \wedge \Gamma_2$ and $\Delta_3 = \Delta_1 \wedge \Delta_2$, then by Corollay 2.7 and Lemma 2.21, $\Gamma_3 \vdash (\lambda y.P)Q:\delta'_1 \times \cdots \times \delta'_n \times \kappa_1 \to \nu \mid \Delta_3$.

b) $V' \equiv \mu\alpha.[\beta]P[\alpha \Leftarrow Q]$; we can assume that Γ_2 and Δ_2 do not have types for bound variables and names in $\mu\alpha.[\beta]P$. Let $\Gamma_3 = \Gamma_1 \wedge \Gamma_2$ and $\Delta_3 = \Delta_1 \wedge \Delta_2$, then by Corollay 2.7 and Lemma 2.21, $\Gamma_3 \vdash (\mu\alpha.[\beta]P)Q : \delta'_1 \times \cdots \times \delta'_n \times \kappa_1 \to \nu \mid \Delta_3$.

c) $V' \equiv z$. Then, by induction, there are Γ', Δ', δ''_j such that $\delta''_j \leq \delta'_j$, and $\Gamma' \vdash P_j:\psi'_j \mid \Delta'$. Take $\Gamma_3 = \Gamma_1 \wedge \Gamma', z:\delta'_1 \times \cdots \times \delta''_j \times \cdots \times \delta'_n \times \kappa'_1 \to \nu$, and $\Delta_3 = \Delta_1 \wedge \Delta'$, then

$$\Gamma_3 \vdash z:\delta'_1 \times \cdots \times \delta''_j \times \cdots \times \delta'_n \times \kappa'_1 \to \nu | \Delta_3.$$

In all cases, $\Gamma_3 \leq \Gamma_1$, $\Delta_3 \leq \Delta_1$, and $\Gamma_3 \vdash VP_1 \cdots P_n:\delta \mid \Delta_3$.

$M = \lambda y.M'$: If $M \to_{lor} N$, then $N = \lambda y.N'$ and $M' \to_{lor} N'$. Then there exists δ and κ such that $\delta_1 = \delta \times \kappa \to \nu$, and $\Gamma_1, y:\delta \vdash N':\kappa \to \nu \mid \Delta_1$. By induction, there exists $\Gamma' \leq \Gamma_1$, $\Delta' \leq \Delta_1$, $\delta' \leq \delta$, and $\kappa' \leq \kappa$ such that $\Gamma', y:\delta' \vdash M':\kappa' \to \nu \mid \Delta'$. Then, by rule (abs), $\Gamma' \vdash \lambda y.M':\delta' \times \kappa' \to \nu \mid \Delta'$. Notice that $\delta \times \kappa \to \nu \leq \delta' \times \kappa' \to \nu$; take $\Gamma_3 = \Gamma'$, $\Delta_3 = \Delta'$, and $\delta_3 = \delta' \times \kappa' \to \nu$.

$M = \mu\alpha.[\beta]M'$: If $M \to_{lor} N$, then $N = \mu\alpha.[\beta]N'$ and $M' \to_{lor} N'$. Then there exists κ_1 and κ_2 such that $\delta_1 = \kappa_1 \to \nu$, $\Delta_1 = \alpha:\kappa_2, \Delta'_1$ and $\Gamma_1 \vdash N':\kappa_2 \to \nu \mid \beta:\kappa_1, \Delta'_1$. By induction, there exists $\Gamma' \leq \Gamma_1$, $\Delta' \leq \Delta_1$, $\kappa' \leq \kappa_1$ and $\kappa'' \leq \kappa_2$ such that $\Gamma' \vdash M':\kappa'' \to \nu \mid \alpha:\kappa', \Delta'$. Then, by rule (μ), $\Gamma' \vdash \mu\alpha.[\beta]M':\kappa' \to \nu \mid \beta:\kappa'', \Delta'$. Notice that $\kappa_1 \to \nu \leq \kappa' \to \nu$, and $\beta:\kappa'', \Delta' \leq \beta:\kappa_2, \Delta'_1$; take $\Gamma_3 = \Gamma'$, $\Delta_3 = \beta:\kappa'', \Delta'$, and $\delta_3 = \kappa' \to \nu$.

$M = \mu\alpha.[\alpha]M'$: If $M \to_{lor} N$, then $N = \mu\alpha.[\alpha]N'$ and $M' \to_{lor} N'$. Then there exists κ such that $\delta_1 = \kappa \to \nu$, $\Delta_1 = \alpha:\kappa, \Delta'_1$ and $\Gamma_1 \vdash N':\kappa \to \nu \mid \alpha:\kappa, \Delta'_1$. By induction, there exists $\Gamma' \leq \Gamma_1$, $\Delta' \leq$

Δ_1, and $\kappa_1 \leq \kappa$, $\kappa_2 \leq \kappa$ such that $\Gamma' \vdash M' : \kappa_2 \to \nu \mid \alpha:\kappa_1, \Delta'$. Take $\kappa' = \kappa_1 \wedge \kappa_2$, then by weakening and rule $(\leq)$, also $\Gamma' \vdash M' : \kappa' \to \nu \mid \alpha:\kappa', \Delta'$. Then, by rule (μ), $\Gamma' \vdash \mu\alpha.[\alpha]M' : \kappa' \to \nu \mid \Delta'$. Notice that $\kappa \to \nu \leq \kappa' \to \nu$; take $\Gamma_3 = \Gamma'$, $\Delta_3 = \Delta'$, and $\delta_3 = \kappa' \to \nu$. ■

We can now show that all strongly normalisable terms are typeable in our system.

Theorem 2.24 (Typeability of $\mathcal{SN}$-Terms) *For all $M \in \mathcal{SN}$ there exist Γ and Δ and a type δ such that $\Gamma \vdash M : \delta \mid \Delta$.*

Proof: By induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by $\#(M)$).

i) If $\#(M) = 0$, then M is in normal form, and by Lemma 2.20, there exist Γ and δ such that $\Gamma \vdash M : \delta \mid \Delta$.

ii) If $\#(M) \geq 1$, so M contains a redex, then let $M \to_{lor} N$ by contracting PQ. Then $\#(N) < \#(M)$, and $\#(Q) < \#(M)$ (since Q is a proper subterm of a redex in M), so by induction $\Gamma \vdash N : \delta_1 \mid \Delta$ and $\Gamma' \vdash Q : \delta_2 \mid \Delta$, for some Γ, Γ', δ_1, and δ_2. Then, by Lemma 2.23, there exist Γ_1, Δ_1 and δ' such that $\Gamma_1 \vdash M : \delta' \mid \Delta_1$. ■

In the following section we will prove strong normalisation for terms typeable in the propositional fragment of Parigot's logical system [20] via an interpretation in our system.

3 Interpretation of Parigot's Logical System

We use a version of Parigot's logical system (as presented in [20] which is equivalent to the original one if only terms (so not also proper commands, i.e. elements of Cmd) are typed. This implies that the rule for $\bot$ does not need to be taken into account.[4] We call this propositional fragment of Parigot's original system the *simply-typed $\lambda\mu$-calculus*.

Definition 3.1 (Simply Typed $\lambda\mu$-calculus) *i*) The set **LF** of *Logical Formulas* is defined by

$$A, B ::= \varphi \mid A \to B$$

where φ ranges over an infinite set of *Proposition (Type) Variables*.

ii) The inference rules of this system are:

$$(ax): \frac{}{\Gamma, x:A \vdash x : A \mid \Delta} \quad (\mu_1): \frac{\Gamma \vdash M : A \mid \alpha:A, \Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta} \quad (\mu_2): \frac{\Gamma \vdash M : B \mid \alpha:A, \beta:B, \Delta}{\Gamma \vdash \mu\alpha[\beta]M : A \mid \beta:B, \Delta}$$

$$(\to I): \frac{\Gamma, x:A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : A \to B \mid \Delta} \quad (\to E): \frac{\Gamma \vdash M : A \to B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta}$$

We write $\Gamma \vdash_{\mathrm{P}} M : A \mid \Delta$ to denote that this judgement is derivable in this system.

We can interpret formulas into types of our system as follows.

Definition 3.2 The translation functions $(\cdot)^D : \mathbf{LF} \to \mathcal{T}_D$ and $(\cdot)^C : \mathbf{LF} \to \mathcal{T}_C$ are defined by (re-

[4] The system we consider here does not include rules $(\forall I)$ and $(\forall E)$, since they have no effect on the subject in Parigot's first-order type assignment system.

member that ν is the (only) base type):

$$\begin{array}{rcl} \varphi^C & = & \nu\times\omega \\ (A\to B)^C & = & (A^C\to\nu)\times B^C \\ A^D & = & A^C\to\nu \end{array}$$

For example, $(\varphi_1\to\varphi_2\to\varphi_3)^C = (\nu\times\omega\to\nu)\times(\nu\times\omega\to\nu)\times(\nu\times\omega\to\nu)$.

It is straightforward to show that the above translations are well defined. We extend them to bases and name contexts as follows: $\Gamma^D = \{\, x{:}A^D \mid x{:}A \in \Gamma \,\}$ and $\Delta^C = \{\, \alpha{:}A^C \mid \alpha{:}A \in \Delta \}$.

Theorem 3.3 (Derivability preservation) *If $\Gamma \vdash_{\mathrm{P}} M{:}A \mid \Delta$, then $\Gamma^D \vdash M{:}A^D \mid \Delta^C$.*

Proof: By induction on the structure of derivations. Each rule of the simply-typed $\lambda\mu$-calculus has a corresponding one in our intersection type system (allowing for the fact that rule $(\to I)$ gets mapped unto (abs) and $(\to E)$ gets mapped unto (app)); hence it suffices to show that rules are preserved when translating formulas into types. We show just the cases for the μ-abstraction.

$$\dfrac{\begin{array}{c}\\ \Gamma^D \vdash M{:}A^D \mid \alpha{:}A^C, \Delta^C \end{array}}{\Gamma^D \vdash \mu\alpha.[\alpha]M{:}A^D \mid \Delta^C}\,(\mu_1) \qquad \text{becomes} \qquad \dfrac{\begin{array}{c}\\ \Gamma^D \vdash M{:}A^C\to\nu \mid \alpha{:}A^C, \Delta^C \end{array}}{\Gamma^D \vdash \mu\alpha.[\alpha]M{:}A^C\to\nu \mid \Delta^C}\,(\mu)$$

$$\dfrac{\begin{array}{c}\\ \Gamma^D \vdash M{:}B^D \mid \alpha{:}A^C, \beta{:}B^C, \Delta^C \end{array}}{\Gamma^D \vdash \mu\alpha.[\beta]M{:}A^D \mid \beta{:}B^C, \Delta^C}\,(\mu_2) \qquad \text{becomes} \qquad \dfrac{\begin{array}{c}\\ \Gamma^D \vdash M{:}B^C\to\nu \mid \alpha{:}A^C, \beta{:}B^C, \Delta^C \end{array}}{\Gamma^D \vdash \mu\alpha.[\beta]M{:}A^C\to\nu \mid \beta{:}B^C, \Delta^C}\,(\mu)$$

notice that the applications of rule (μ) are valid instances of that rule. ∎

Strong normalisation of typeable terms in Parigot's simply typed $\lambda\mu$-calculus now follows as a consequence of our characterisation result.

Theorem 3.4 (Strong Normalisability of Parigot's Simply Typed $\lambda\mu$-calculus) *If $\Gamma \vdash_{\mathrm{P}} M{:}A \mid \Delta$, then $M \in \mathcal{SN}$.*

Proof: By Theorem 3.3, if $\Gamma \vdash_{\mathrm{P}} M{:}A \mid \Delta$ then $\Gamma^D \vdash M{:}A^D \mid \Delta^C$ is derivable in the intersection type system. Hence $M \in \mathcal{SN}$ by Theorem 2.18. ∎

Conclusion

We have defined an intersection type system which characterises strongly normalising $\lambda\mu$-terms, extending the strong normalisation result for the λ-calculus to the pure $\lambda\mu$-calculus.

We have also provided a translation of propositional types of Parigot's system into types of the system proposed in this paper (a restriction of the one presented in [6]) and proved that derivability is preserved. We are confident that such a result can be extended to the full first-order type assignment system, to obtain an alternative proof of Parigot's strong normalisation theorem.

As we have observed in [6], our intersection-type assignment system can be adapted to de Groote's variant of the $\lambda\mu$-calculus (see e.g. [17]) (called $\Lambda\mu$ by Saurin [23]), that satisfies stronger properties than Parigot's original calculus, such as Böhm's theorem. We leave the question whether the present characterisation result extends to those cases to future work.

Acknowledgements The authors wish to thank Mariangiola Dezani-Ciancaglini for her unceasing support.

References

[1] Z.M. Ariola & H. Herbelin (2003): *Minimal Classical Logic and Control Operators*. In J.C.M. Baeten, J.K. Lenstra, J. Parrow & G.J. Woeginger, editors: *Proceedings of Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003*, *Lecture Notes in Computer Science* 2719, Springer Verlag, pp. 871–885, doi:10.1007/3-540-45061-0_68.

[2] S. van Bakel (1992): *Complete restrictions of the Intersection Type Discipline*. *Theoretical Computer Science* 102(1), pp. 135–163, doi:10.1016/0304-3975(92)90297-S.

[3] S. van Bakel (2004): *Cut-Elimination in the Strict Intersection Type Assignment System is Strongly Normalising*. *Notre Dame journal of Formal Logic* 45(1), pp. 35–63, doi:10.1305/ndjfl/1094155278.

[4] S. van Bakel (2010): *Sound and Complete Typing for $\lambda\mu$*. In: *Proceedings of 5th International Workshop* Intersection Types and Related Systems *(ITRS'10), Edinburgh, Scotland*, *Electronic Proceedings in Theoretical Computer Science* 45, pp. 31–44, doi:10.4204/EPTCS.45.3.

[5] S. van Bakel (2011): *Strict intersection types for the Lambda Calculus*. *ACM Computing Surveys* 43, pp. 20:1–20:49, doi:10.1145/1922649.1922657.

[6] S. van Bakel, F. Barbanera & U. de'Liguoro (2011): *A Filter Model for $\lambda\mu$*. In L. Ong, editor: *Proceedings of* 10th International Conference on Typed Lambda Calculi and Applications *(TLCA'11)*, *Lecture Notes in Computer Science* 6690, Springer Verlag, pp. 213–228, doi:10.1007/978-3-642-21691-6_18.

[7] S. van Bakel & P. Lescanne (2008): *Computation with Classical Sequents*. *Mathematical Structures in Computer Science* 18, pp. 555–609, doi:10.1017/S0960129508006762.

[8] F. Barbanera & S. Berardi (1996): *A Symmetric Lambda Calculus for Classical Program Extraction*. *Information and Computation* 125(2), pp. 103–117, doi:10.1006/inco.1996.0025.

[9] H. Barendregt (1984): *The Lambda Calculus: its Syntax and Semantics*, revised edition. North-Holland, Amsterdam.

[10] H. Barendregt, M. Coppo & M. Dezani-Ciancaglini (1983): *A filter lambda model and the completeness of type assignment*. *Journal of Symbolic Logic* 48(4), pp. 931–940, doi:10.2307/2273659.

[11] A. Church (1936): *A Note on the Entscheidungsproblem*. *Journal of Symbolic Logic* 1(1), pp. 40–41, doi:10.2307/2269326.

[12] M. Coppo & M. Dezani-Ciancaglini (1978): *A New Type Assignment for λ-Terms*. *Archiv für Mathematische Logic und Grundlagen Forschung* 19, pp. 139–156, doi:10.1007/BF02011875.

[13] M. Coppo, M. Dezani-Ciancaglini & B. Venneri (1980): *Principal type schemes and λ-calculus semantics*. In J.R. Hindley & J.P. Seldin, editors: *To H.B. Curry, Essays in combinatory logic, lambda-calculus and formalism*, Academic press, New York, pp. 535–560.

[14] P.-L. Curien & H. Herbelin (2000): *The Duality of Computation*. In: *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, *ACM Sigplan Notices* 35.9, ACM, pp. 233–243, doi:10.1145/351240.351262.

[15] G. Gentzen (1935): *Investigations into logical deduction*. In: *The Collected Papers of Gerhard Gentzen*, Ed M. E. Szabo, North Holland, 68ff (1969).

[16] S. Ghilezan (1996): *Strong Normalization and Typability with Intersection Types*. *Notre Dame journal of Formal Logic* 37(1), pp. 44–52, doi:10.1305/ndjfl/1040067315.

[17] Ph. de Groote (1994): *On the Relation between the $\lambda\mu$-Calculus and the Syntactic Theory of Sequential Control*. In: *Proceedings of 5th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'94)*, *Lecture Notes in Computer Science* 822, Springer Verlag, pp. 31–43, doi:10.1007/3-540-58216-9_27.

[18] H. Herbelin & A. Saurin (2010): *$\lambda\mu$-calculus and $\lambda\mu$-calculus: a Capital Difference*. Manuscript.

[19] J.-L Krivine (1993): *Lambda calculus, types and models*. Ellis Horwood.

[20] M. Parigot (1992): *An algorithmic interpretation of classical natural deduction*. In: *Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'92)*, *Lecture Notes in Computer Science* 624, Springer Verlag, pp. 190–201, doi:10.1007/BFb0013061.

[21] M. Parigot (1997): *Proofs of Strong Normalisation for Second Order Classical Natural Deduction*. *Journal of Symbolic Logic* 62(4), pp. 1461–1479, doi:10.2307/2275652.

[22] G. Pottinger (1980): *A Type Assignment for the Strongly Normalizable λ-terms*. In J.P. Seldin & J.R. Hindley, editors: *To H. B. Curry, Essays in Combinatory Logic, Lambda-Calculus and Formalism*, Academic press, New York, pp. 561–577.

[23] A. Saurin (2008): *On the Relations between the Syntactic Theories of λμ-Calculi*. In M. Kaminski & S. Martini, editors: *Computer Science Logic, 22nd International Workshop (CSL'08), Bertinoro, Italy*, *Lecture Notes in Computer Science* 5213, Springer Verlag, pp. 154–168, doi:10.1016/j.entcs.2005.11.072.

[24] A. Saurin (2010): *Standardization and Böhm Trees for λμ-calculus*. In M. Blume, N. Kobayashi & G. Vidal, editors: *Functional and Logic Programming, 10th International Symposium, (FLOPS'10), Sendai, Japan*, *Lecture Notes in Computer Science* 6009, Springer Verlag, pp. 134–149, doi:10.1007/978-3-642-12251-4_11.

[25] Th. Streicher & B. Reus (1998): *Classical logic: Continuation Semantics and Abstract Machines*. *Journal of Functional Programming* 11(6), pp. 543–572, doi:10.1017/S0956796898003141.

[26] W. Tait (1967): *Intensional Interpretations of Functionals of Finite Type I*. *Journal of Symbolic Logic* 32(2), pp. 198–212, doi:10.2307/2271658.