Characterisation of Approximation and (Head) Normalisation for $\lambda\mu$ using Strict Intersection Types

Steffen van Bakel

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK

We study the strict type assignment for $\lambda\mu$ that is presented in [7]. We define a notion of approximants of $\lambda\mu$-terms, show that it generates a semantics, and that for each typeable term there is an approximant that has the same type. We show that this leads to a characterisation via assignable types for all terms that have a head normal form, and to one for all terms that have a normal form, as well as to one for all terms that are strongly normalisable.

Introduction

The Intersection Type Discipline [13] is an extension of the standard, implicative type assignment known as Curry's system [16] for the λ-calculus [15, 12]; the extension made consists of relaxing the requirement that a parameter for a function should have a single type, adding the type constructor $\cap$ next to $\rightarrow$. This simple extension allows for a great leap in complexity: not only can a (filter) model be built for the λ-calculus using intersection types, also strong normalisation (termination) can be characterised via assignable types; however, type assignment becomes undecidable.

A natural question is whether or intersection type assignment yields a semantics also for other calculi, like $\lambda\mu$ [19]. To answer that, in [8, 9, 10] a notion of intersection type assignment was defined for $\lambda\mu$ that is a variant of the union-intersection system defined in [5]. Inspired by Streicher and Reus's domain [23], $\lambda\mu$-terms are separated into terms and *streams*; then $\lambda\mu$'s names act as the destination of streams, the same way variables are the destination of terms. A type theory is defined following the domain construction; the main results for that system are the definition of a filter model, closure under conversion, and that the system is an extension of Parigot's [8]; and that, in a restricted system, the terms that are typeable are exactly the strongly normalising ones [9].

One of the main disadvantages of taking the domain-directed approach to type assignment is that, naturally, intersection becomes a 'top level' type constructor, that lives at the same level as arrow, for example, which induces a contra-variant type inclusion relation '$\leq$' and type assignment rule $(\leq)$ that greatly hinder proofs and gives an intricate generation lemma. This problem is addressed in [7] where a *strict* version of the system of [10] is defined, in the spirit of that of [1, 6] that allows for more easily constructed proofs. The main restriction with respect to the system of [10] is limiting the type inclusion relation to a relation that is no longer contra-variant, and allows only for the selection of a component of an intersection type; this is accompanied by a restriction of the type language, essentially no longer allowing intersection on the right of an arrow. The main results shown in [7] are that the system is closed under conversion (i.e. under reduction and expansion), and that all terms typeable in a system that excludes the type constant ω are strongly normalisable. To that aim it shows that, in this system, cut-elimination is strongly normalisable, using the technique of derivation reduction [3] (see also [4, 6]).

In this paper, we will elaborate further on the strict system. As in [4, 6], in this paper we will show that the fact that derivation reduction is strongly normalisable also here leads to an approximation

N. Kobayashi (Ed.): Eighth Workshop
on Intersection Types and Related Systems (ITRS 2016).
EPTCS 242, 2017, pp. 20–30, doi:10.4204/EPTCS.242.4

result. For that, we define a notion of approximation for $\lambda\mu$, and show that this yields a semantics (Thm. 13). We then show that for every typeable term there exists an approximant of that term that can be assigned exactly the same types (Thm. 17). We then show that this approximation result naturally gives a characterisation of head normalisation (Thm 18), as well as a characterisation of normalisation (Thm 24). We also revisit the proof of characterisation of strong normalisation of terms through the assignable types (Thm 28), which thanks to the approximation result has a more elegant proof.

Because of the restricted available space, most of the (full) proofs are not presented here. A version of this paper with the proofs added in an appendix can be found at `www.doc.ic.ac.uk/~svb/Research/Papers/ITRS16wapp.pdf`.

Note: We will write $\underline{n}$ for the set $\{1,\ldots,n\}$ and use a vector notation for the abbreviation of sequences, so write $\vec{X}_{\underline{n}}$ for $X_1,\ldots,X_n$, and $\vec{X}$ if the number of elements in the sequence is not important.

1 The $\lambda\mu$-calculus

In this section we present Parigot's pure $\lambda\mu$-calculus as introduced in [19]. It is an extension of the untyped λ-calculus obtained by adding *names* and a name-abstraction operator μ and was intended as a proof calculus for a fragment of classical logic. Derivable statements have the shape $\Gamma \vdash M : A \mid \Delta$, where A is the main (*active*) conclusion of the statement, and Δ contains the alternative conclusions, consisting of pairs of names and types; the left-hand context Γ, as usual, is a mapping from term variables to types, and represents the assumptions about free variables of M.

Definition 1 (TERM SYNTAX [19]) Let $x,y,z,\ldots$ range over *term variables*, and $\alpha,\beta,\gamma,\delta,\ldots$ range over *names*. The *terms*, ranged over by $M,N,P,Q,\ldots$ are defined by the grammar:

$$M,N \quad ::= \quad x \mid \lambda y.M \mid MN \mid \mu\alpha.[\beta]M$$

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

Definition 2 (SUBSTITUTION [19]) Substitution takes two forms:

term substitution:	$M[N/x]$	(N is substituted for x in M)
structural substitution:	$M[L{\cdot}\gamma/\alpha]$	(every 'subterm' $[\alpha]N$ of M is replaced by $[\gamma]NL$)

As usual, both substitutions are capture avoiding, using α-conversion when necessary.

Definition 3 (REDUCTION [19]) Reduction in $\lambda\mu$ is based on the following rules:

$$\begin{array}{rrcll}
(\beta): & (\lambda x.M)N & \rightarrow & M[N/x] & & (\textit{logical reduction}) \\
(\mu)^1: & \left\{\begin{array}{l}(\mu\beta.[\beta]P)Q \\ (\mu\beta.[\delta]P)Q\end{array}\right. & \begin{array}{c}\rightarrow \\ \rightarrow\end{array} & \begin{array}{l}\mu\gamma.[\gamma](P[Q{\cdot}\gamma/\beta])Q \\ \mu\gamma.[\delta]P[Q{\cdot}\gamma/\beta],\end{array} & \begin{array}{l} \\ \text{if } \delta\neq\beta\end{array} & (\textit{structural reduction}) \\
(\text{REN}): & \left\{\begin{array}{l}\mu\alpha.[\beta]\mu\gamma.[\gamma]M \\ \mu\alpha.[\beta]\mu\gamma.[\delta]M\end{array}\right. & \begin{array}{c}\rightarrow \\ \rightarrow\end{array} & \begin{array}{l}\mu\alpha.[\beta]M[\beta/\gamma] \\ \mu\alpha.[\delta]M[\beta/\gamma],\end{array} & \begin{array}{l} \\ \text{if } \delta\neq\gamma\end{array} & (\textit{renaming})
\end{array}$$

We write $M \rightarrow_{\beta\mu} N$ for the reduction relation that is the compatible closure of these rules, and $=_{\beta\mu}$ for the equivalence relation generated by it.

[1] A more common notation for the second rule, for example, would be $(\mu\beta.[\delta]M)N \rightarrow \mu\beta.[\delta]M[N/\beta]$. This implicitly uses the fact that β disappears during reduction, and through α-conversion can be picked as name for the newly created applications instead of γ. But, in fact, this is not the same β (and the named term has changed), as reflected in the fact that its type changes during reduction. Moreover, when making the substitution *explicit* as in [11], it becomes clear that this other approach in fact is a short-cut, which our definition does without.

Confluence for this notion of reduction has been shown in [20].

We will need the concept of head-normal form for $\lambda\mu$, which is defined as follows:

Definition 4 (HEAD-NORMAL FORMS) The $\lambda\mu$ *head-normal forms* (with respect to our notion of reduction $\rightarrow_{\beta\mu}$) are defined through the grammar:

$$\begin{array}{rcll} \boldsymbol{H} & ::= & xM_1\cdots M_n & (n \geq 0) \\ & | & \lambda x.\boldsymbol{H} & \\ & | & \mu\alpha.[\beta]\boldsymbol{H} & (\boldsymbol{H} \neq \mu\gamma.[\delta]\boldsymbol{H}') \end{array}$$

2 Strict type assignment

Intersection (and union) type assignment for $\lambda\mu$ was first defined in [5]; this was followed by [8], in which an intersection type theory is developed departing from Streicher and Reus's domain construction [23]. Terms can be typed with functional types δ and streams by continuation types κ that are of the shape $\delta_1\times\cdots\times\delta_n\times\omega$, so essentially is a sequence of δs. This later [9] was followed by the proof that, as for the λ-calculus, the underlying intersection type system for $\lambda\mu$ allows for the full characterisation of strongly normalisable terms; in that paper, renaming is not considered. These papers were later combined (and revised) into [10]. One of the main disadvantages of taking the domain-directed approach to type assignment is that, naturally, intersection becomes a 'top level' type constructor, that lives at the same level as arrow, for example. This in itself is not negative, since it gives readable types and easy-to-understand type assignment rules, but it also induces a contra-variant type inclusion relation '$\leq$' and type assignment rule $(\leq)$ that hinder proofs and give an intricate generation lemma (see [10] for details).

Therefore, in [7], a strict restriction of the system of [10] was presented, where the occurrence of intersections is limited to only appear as components of continuation types (so no intersections of continuation types), and type inclusion is no longer contra-variant and only allows for the selection of a component in an intersection type. It also uses Ω rather than ω to mark the end of a continuation type. But, more importantly, it removed the inference rule $(\leq)$, and changed the type assignment rules to explicitly state when a $\leq$-step is allowed, as in rule (Ax).

This system is defined as follows:

Definition 5 (STRICT TYPES [7]) 1. Let v range over a countable, infinite set of type constants. We define our strict types by the grammar:

$$\begin{array}{rcll} \mathsf{A},\mathsf{B} & ::= & \mathsf{C}\rightarrow v & \textit{basic types} \\ \mathsf{R},\mathsf{S},\mathsf{T} & ::= & \omega \mid \mathsf{A}_1\cap\cdots\cap\mathsf{A}_n \quad (n\geq 1) & \textit{intersection types} \\ \mathsf{C},\mathsf{D} & ::= & \Omega \mid \mathsf{S}\times\mathsf{C} & \textit{continuation types} \end{array}$$

2. On strict types, the type inclusion relation $\leq_{\mathrm{S}}$ is the smallest partial order satisfying the rules:

$$\frac{}{\mathsf{A}_1\cap\cdots\cap\mathsf{A}_n \leq \mathsf{A}_j}\;(j\in\underline{n},\ n\geq 1) \qquad \frac{\mathsf{S}\leq\mathsf{A}_i \quad (\forall i\in\underline{n})}{\mathsf{S}\leq\mathsf{A}_1\cap\cdots\cap\mathsf{A}_n}\;(n\geq 1) \qquad \frac{}{\mathsf{S}\leq\omega} \qquad \frac{}{\mathsf{C}\leq\Omega} \qquad \frac{\mathsf{S}\leq\mathsf{T} \quad \mathsf{C}\leq\mathsf{D}}{\mathsf{S}\times\mathsf{C}\leq\mathsf{T}\times\mathsf{D}}$$

For convenience, we will write $\cap_I\mathsf{A}_i$ for $\mathsf{A}_{i_1}\cap\cdots\cap\mathsf{A}_{i_n}$ where $I=\{i_1,\ldots,i_n\}$, $\cap_\phi\mathsf{A}_i$ for ω, so the second and third rule combine to

$$\frac{\mathsf{S}\leq\mathsf{A}_i \quad (\forall i\in\underline{n})}{\mathsf{S}\leq\mathsf{A}_1\cap\cdots\cap\mathsf{A}_n}\;(n\geq 0)$$

and $\cap_{\underline{n}}\mathsf{A}_i$ for $\mathsf{A}_1\cap\ldots\cap\mathsf{A}_n$. Notice that for any continuation type C there are $n\geq 0$ and S_i $(i\in\underline{n})$ such that $\mathsf{C}=\mathsf{S}_1\times\cdots\mathsf{S}_n\times\Omega$.

Definition 6 (STRICT TYPE ASSIGNMENT [7]) 1. A *variable context* Γ is a mapping from term variables to intersection types, denoted as a finite set of *statements* $x{:}S$, such that the *subject* of the statements (x) are distinct.

2. We write $\Gamma,x{:}S$ for the context defined by:

$$\begin{array}{rcll} \Gamma,x{:}S & \stackrel{\Delta}{=} & \Gamma \cup \{x{:}S\}, & \text{if } \Gamma \text{ is not defined on } x \\ & \stackrel{\Delta}{=} & \Gamma, & \text{if } x{:}S \in \Gamma \end{array}$$

We write $x \notin \Gamma$ if there exists no S such that $x{:}S \in \Gamma$.

3. *Name contexts* Δ and the notions $\alpha{:}\mathrm{C},\Delta$ and $\alpha \notin \Delta$ are defined in a similar way.

4. We define *strict type assignment* for $\lambda\mu$-terms through the following natural deduction system:

$$(Ax): \frac{}{\Gamma,x{:}S \vdash x : A \mid \Delta}\,(S \leq_{\mathrm{S}} A) \qquad (\cap): \frac{\Gamma \vdash M : A_i \mid \Delta \quad (\forall i \in I)}{\Gamma \vdash M : \cap_I A_i \mid \Delta}\,(I = \emptyset \vee |I| \geq 2)$$

$$(Abs): \frac{\Gamma,x{:}S \vdash M : \mathrm{C} \to v \mid \Delta}{\Gamma \vdash \lambda x.M : S \times \mathrm{C} \to v \mid \Delta}\,(x \notin \Gamma) \qquad (\mu): \frac{\Gamma \vdash M : \mathrm{D} \to v \mid \alpha{:}\mathrm{C},\Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : \mathrm{C} \to v \mid \Delta}\,(\alpha \notin \Delta, \mathrm{C} \leq_{\mathrm{S}} \mathrm{D})$$

$$(App): \frac{\Gamma \vdash M : S \times \mathrm{C} \to v \mid \Delta \quad \Gamma \vdash N : S \mid \Delta}{\Gamma \vdash MN : \mathrm{C} \to v \mid \Delta} \qquad (\mu'): \frac{\Gamma \vdash M : \mathrm{D} \to v \mid \alpha{:}\mathrm{C},\beta{:}\mathrm{C}',\Delta}{\Gamma \vdash \mu\alpha.[\beta]M : \mathrm{C} \to v \mid \beta{:}\mathrm{C}',\Delta}\,\begin{array}{l}(\beta \neq \alpha \;\&\; \alpha \notin \Delta, \\ \mathrm{C}' \leq_{\mathrm{S}} \mathrm{D})\end{array}$$

We write $\Gamma \vdash_{\mathrm{S}} M : S \mid \Delta$ for judgements derivable using these rules, and prefix this with '$\mathcal{D}$ :: ' if we want to name the derivation.

5. The relation $\leq_{\mathrm{S}}$ is naturally extended to variable contexts as follows:

$$\Gamma \leq_{\mathrm{S}} \Gamma' \quad \stackrel{\Delta}{=} \quad \forall x{:}S \in \Gamma' \; \exists x{:}T \in \Gamma \; [T \leq_{\mathrm{S}} S];$$

$\Delta \leq_{\mathrm{S}} \Delta'$ is defined similarly.

Definition 7 By abuse of notation, we allow the notation $S \cap T$, where $S = \cap_{\underline{n}} A_i$ and $T = \cap_{\underline{m}} B_j$, which stands for $A_1 \cap \cdots \cap A_n \cap B_1 \cap \cdots \cap B_m$. Given two contexts Γ_1 and Γ_2, we define the context $\Gamma_1 \cap \Gamma_2$ as follows:

$$\begin{array}{rcl} \Gamma_1 \cap \Gamma_2 & \stackrel{\Delta}{=} & \{\, x{:}S_1 \cap S_2 \mid x{:}S_1 \in \Gamma_1 \;\&\; x{:}S_2 \in \Gamma_2 \,\} \cup \\ & & \{\, x{:}S \mid x{:}S \in \Gamma_1 \;\&\; x \notin \Gamma_2 \,\} \cup \{\, x{:}S \mid x{:}S \in \Gamma_2 \;\&\; x \notin \Gamma_1 \,\} \end{array}$$

and write $\cap_{\underline{n}} \Gamma_i$ for $\Gamma_1 \cap \cdots \cap \Gamma_n$. We will also allow intersection of continuation types as short-hand notation: let $\mathrm{D} = S_1 \times \cdots \times S_n \times \Omega$, and $\mathrm{C} = T_1 \times \cdots \times T_m \times \Omega$ and assume, that $n < m$; we define

$$\mathrm{D} \cap \mathrm{C} \quad \stackrel{\Delta}{=} \quad S_1 \cap T_1 \times \cdots \times S_n \cap T_n \times T_{n+1} \times \cdots \times T_m \times \Omega.$$

(we need this notion in the proof of Thm. 18). Then $\Delta_1 \cap \Delta_2$ is defined the same way as $\Gamma_1 \cap \Gamma_2$.

In [7] it is then shown that this notion of type assignment is closed under conversion, so can be used to define a (filter) semantics. That paper also defines a notion of cut-elimination, by defining derivation reduction $\to_{\mathrm{DER}}$, where only those redexes in terms are contracted that are typed with a type different from ω; it shows that this notion is strongly normalisable, which then leads to the proof that all terms typeable in a restriction of $\vdash_{\mathrm{S}}$ that eliminates the type constant ω, are strongly normalisable.

The main results shown in [7] that are relevant to this paper are:

Theorem 8 ([7]) *1. If $\Gamma \vdash_{\mathrm{S}} M : S \mid \Delta$, $\Gamma' \leq_{\mathrm{S}} \Gamma$, $\Delta' \leq_{\mathrm{S}} \Delta$,[2] and $S \leq_{\mathrm{S}} T$, then $\Gamma' \vdash_{\mathrm{S}} M : T \mid \Delta'$.*

[2] The condition $\Delta' \leq_{\mathrm{S}} \Delta$ might seem counterintuitive, since one might expect the inclusion relation to be reversed. To support intuition, we can see types in name contexts as negations, and $\alpha{:}A \times \Omega$ as $\alpha{:}\neg A$. Notice that $A \cap B \times \Omega \leq_{\mathrm{S}} A \times \Omega$; obviously we have $\alpha{:}A \cap B \times \Omega \leq_{\mathrm{S}} \alpha{:}A \times \Omega$ and $\neg A \leq \neg A \cup \neg B$.

2. *If $\Gamma \vdash_{\rm S} M : A \mid \Delta$ and $M =_{\beta\mu} N$, then $\Gamma \vdash_{\rm S} N : A \mid \Delta$.*
3. *Let $\mathcal{D} :: \Gamma \vdash_{\rm S} M : S \mid \Delta$, and $\mathcal{D} \rightarrow^*_{\rm DER} \mathcal{D}' :: \Gamma \vdash_{\rm S} N : S \mid \Delta$, then $M \rightarrow^*_{\beta\mu} N$.*
4. *If $\mathcal{D} :: \Gamma \vdash_{\rm S} M : S \mid \Delta$, then $SN(\mathcal{D})$ ($\mathcal{D}$ is strongly normalisable).*

3 Approximation semantics for $\lambda\mu$

Following the approach of [24], we now define an *approximation semantics* for $\lambda\mu$ with respect to $\rightarrow_{\beta\mu}$.

Essentially, approximants are partially evaluated expressions in which the locations of incomplete evaluation (*i.e.* where reduction *may* still take place) are explicitly marked by the element $\perp$; thus, they *approximate* the result of computations.

Approximation for $\Lambda\mu$ (a variant of $\lambda\mu$ where naming and μ-binding are separated [17]) has been studied by others as well [22, 18]; *weak* approximants for $\lambda\mu$ are studied in [11].

Definition 9 (APPROXIMATION FOR $\lambda\mu$) 1. We define $\lambda\mu\perp$ as an extension of $\lambda\mu$ by adding the term constant $\perp$.

2. The set of $\lambda\mu$'s *approximants* $\mathcal{A}$ with respect to $\rightarrow_{\beta\mu}$ is defined through the grammar:

$$\begin{array}{rcllll} A & ::= & \perp & \mid & xA_1\cdots A_n & (n \geq 0) \\ & & & \mid & \lambda x.A & (A \neq \perp) \\ & & & \mid & \mu\alpha.[\beta]A & (A \neq \mu\gamma[\delta]A',\ A \neq \perp) \end{array}$$

3. The relation $\sqsubseteq\ \subseteq \lambda\mu\perp^2$ is the smallest preorder that is the compatible extension of $\perp \sqsubseteq M$.
4. The set of *approximants* of M, $\mathcal{A}(M)$, is defined as

$$\mathcal{A}(M) \ \stackrel{\Delta}{=}\ \{A \in \mathcal{A} \mid \exists N \in \lambda\mu\ [M \rightarrow^*_{\beta\mu} N \ \&\ A \sqsubseteq N]\}.$$

5. *Approximation equivalence* between terms is defined through: $M \sim_{\mathcal{A}} N \ \stackrel{\Delta}{=}\ \mathcal{A}(M) = \mathcal{A}(N)$.

The relationship between the approximation relation and reduction is characterised by:

Lemma 10 1. *If $A \sqsubseteq M$ and $M \rightarrow^*_{\beta\mu} N$, then $A \sqsubseteq N$.*

2. ***H** is a head-normal form if and only if there exists $A \in \mathcal{A}$ such that $A \sqsubseteq \boldsymbol{H}$ and $A \neq \perp$.*

The following definition introduces an operation of join on $\lambda\mu\perp$-terms.

Definition 11 (JOIN, COMPATIBLE TERMS) 1. The partial mapping *join*, $\sqcup : \lambda\mu\perp^2 \rightarrow \lambda\mu\perp$, is defined by:

$$\begin{array}{rcrcl} \perp \sqcup M & \equiv & M \sqcup \perp & \equiv & M \\ & & x \sqcup x & \equiv & x \\ & & (\lambda x.M) \sqcup (\lambda x.N) & \equiv & \lambda x.(M \sqcup N) \\ & & (\mu\alpha.[\beta]M) \sqcup (\mu\alpha.[\beta]N) & \equiv & \mu\alpha.[\beta](M \sqcup N) \\ & & (M_1M_2) \sqcup (N_1N_2) & \equiv & (M_1 \sqcup N_1)\,(M_2 \sqcup N_2)^3 \end{array}$$

2. If $M \sqcup N$ is defined, then M and N are called *compatible*.

It is easy to show that $\sqcup$ is associative and commutative; we will use $\sqcup_{\underline{n}} M_i$ for the term $M_1 \sqcup \cdots \sqcup M_n$. Note that $\perp$ can be defined as the empty join, i.e. if $M \equiv \sqcup_{\underline{0}} M_i$, then $M \equiv \perp$.

The following lemma shows that the join acts as least upper bound of compatible terms.

[3]The last alternative in the definition of $\sqcup$ defines the join on applications in a more general way than Scott's, that would state that $(M_1M_2) \sqcup (N_1N_2) \ \sqsubseteq\ (M_1 \sqcup N_1)\,(M_2 \sqcup N_2)$, since it is not always sure if a join of two arbitrary terms exists. Since we will use our more general definition only on terms that are compatible, there is no real conflict.

Lemma 12 1. *If $P \sqsubseteq M$, and $Q \sqsubseteq M$, then $P \sqcup Q$ is defined, and:*

$$P \sqsubseteq P \sqcup Q, \quad Q \sqsubseteq P \sqcup Q, \quad \textit{and} \quad P \sqcup Q \sqsubseteq M.$$

2. *If A_1, $A_2 \in \mathcal{A}(M)$, then A_1 and A_2 are compatible.*

We can also define $\ulcorner M \lrcorner = \sqcup \{ A \mid A \in \mathcal{A}(M) \}$ (which by the previous lemma is well defined); then $\ulcorner \cdot \lrcorner$ corresponds to (a $\lambda\mu$ variant of) Böhm trees [14, 12].

As is standard in other settings, interpreting a $\lambda\mu$-term M through its set of approximants $\mathcal{A}(M)$ gives a semantics.

Theorem 13 (APPROXIMATION SEMANTICS FOR $\lambda\mu$) *If $M =_{\beta\mu} N$, then $M \sim_{\mathcal{A}} N$.*

Proof: By induction on the definition of $=_{\beta\mu}$, of which we only show the case $M \rightarrow^*_{\beta\mu} N$.

$(\mathcal{A}(M) \subseteq \mathcal{A}(N))$: If $A \in \mathcal{A}(M)$, then there exists L such that $M \rightarrow^*_{\beta\mu} L$ and $A \sqsubseteq L$. Since $\rightarrow_{\beta\mu}$ is Church-Rosser, there exists R such that $L \rightarrow^*_{\beta\mu} R$ and $N \rightarrow^*_{\beta\mu} R$, so also $M \rightarrow^*_{\beta\mu} R$. Then by Lem. 10, $A \sqsubseteq R$, and since $N \rightarrow^*_{\beta\mu} R$, we have $A \in \mathcal{A}(N)$.

$(\mathcal{A}(N) \subseteq \mathcal{A}(M))$: If $A \in \mathcal{A}(N)$, then there exists L such that $N \rightarrow^*_{\beta\mu} L$ and $A \sqsubseteq L$. But then also $M \rightarrow^*_{\beta\mu} L$, so $A \in \mathcal{A}(M)$. $\square$

The reverse implication of this result does not hold, since terms without head-normal form (which have only $\perp$ as approximant) are not all related by reduction, so approximation semantics is not fully abstract.

4 The approximation and head normalisation results for $\vdash_S$

In this section we will show an approximation result, i.e. for every M, Γ, S, and Δ such that $\Gamma \vdash_S M : S \mid \Delta$, there exists an $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_S A : S \mid \Delta$. From this, the well-known characterisation of (head-)normalisation of $\lambda\mu$-terms using intersection types follows easily, i.e. all terms having a (head) normal form are typeable in $\vdash_S$ (with a type without ω-occurrences). Another result is the well-known characterisation of strong normalisation of typeable $\lambda\mu$-terms, i.e. all terms, typeable in $\vdash_S$ without using the rule $(\cap)$ with $I = \emptyset$, are strongly normalisable.

First we give some auxiliary definitions and results.

The rules of the system $\vdash_S$ are generalised to $\lambda\mu\perp$; therefore, if $\perp$ occurs in a term M and $\mathcal{D} :: \Gamma \vdash_S M : S \mid \Delta$, in that derivation $\perp$ has to appear in a position where the rule $(\cap)$ is used with $I = \emptyset$, i.e., in a sub-term typed with ω. Notice that $\lambda x.\perp$, $\perp M_1 \cdots M_n$, and $\mu\alpha.[\beta]\perp$ are typeable by ω only.

First we show that $\vdash_S$ is closed for $\sqsubseteq$.

Lemma 14 *$\Gamma \vdash_S M : S \mid \Delta$ and $M \sqsubseteq N$ then $\Gamma \vdash_S N : S \mid \Delta$.*

Next we define a notion of type assignment that is similar to that of Def. 6, but differs in that it assigns *ω only to the term $\perp$.*

Definition 15 $\perp$-*type assignment* and $\perp$-*derivations* are defined as $\vdash_S$, with the exception of:

$$(\cap_\perp): \quad \frac{\Gamma \vdash M_i : A_i \mid \Delta \quad (\forall i \in \underline{n})}{\Gamma \vdash \sqcup_{\underline{n}} M_i : \cap_{\underline{n}} A_i \mid \Delta} \; (n = 0 \vee n \geq 2)$$

We write $\Gamma \vdash_\perp M : S \mid \Delta$ if this statement is derivable using a $\perp$-derivation.

Notice that, by rule $(\cap_\perp)$, $\Gamma \vdash_\perp \perp : \omega \mid \Delta$, and that this is the only way to assign ω to a term. Moreover, in that rule, the terms M_j need to be compatible (otherwise their join would not be defined).

Lemma 16 1. *If $\mathcal{D} :: \Gamma \vdash_\perp M : \mathsf{S} \mid \Delta$, then $\mathcal{D} :: \Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta$.*

2. *If $\mathcal{D} :: \Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta$, then there exists $M' \sqsubseteq M$ such that $\mathcal{D} :: \Gamma \vdash_\perp M' : \mathsf{S} \mid \Delta$.*

Notice that, since M' need not be the same as M, the second derivation in part (2) is not exactly the same; however, it has the same structure in terms of applied derivation rules.

Using Thm. 8(4) and Lem. 16, as for the BCD-system (see [21]) and the system of [2], the relation between types assignable to a $\lambda\mu$-term and those assignable to its approximants can be formulated as:

Theorem 17 (APPROXIMATION) $\Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta \iff \exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\mathrm{S}} A : \mathsf{S} \mid \Delta]$.

Proof: $(\Rightarrow)$: If $\mathcal{D} :: \Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta$, then, by Thm. 8(4), $SN(\mathcal{D})$. Let $\mathcal{D}' :: \Gamma \vdash_{\mathrm{S}} N : \mathsf{S} \mid \Delta$ be a normal form of $\mathcal{D}$ with respect to $\rightarrow_{\mathrm{DER}}$, then by Thm. 8(3), $M \rightarrow^*_\beta N$ and, by Lem. 16 (2), there exists $N' \sqsubseteq N$ such that $\mathcal{D}' :: \Gamma \vdash_\perp N' : \mathsf{S} \mid \Delta$. So, in particular, N' contains no redexes (no redexes typed with a type different form ω since $\mathcal{D}'$ is in normal form, and none typed with ω since only $\perp$ can be typed with ω), so $N' \in \mathcal{A}$, and therefore $N' \in \mathcal{A}(M)$.

$(\Leftarrow)$: Let $A \in \mathcal{A}(M)$ be such that $\Gamma \vdash_{\mathrm{S}} A : \mathsf{S} \mid \Delta$. Since $A \in \mathcal{A}(M)$, there exists N such that $M \rightarrow^*_{\beta\mu} N$ and $A \sqsubseteq N$. Then, by Lem. 14, $\Gamma \vdash_{\mathrm{S}} N : \mathsf{S} \mid \Delta$, and, by Thm. 8(2), also $\Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta$. □

Using this last result, the characterisation of head-normalisation becomes easy to show.

Theorem 18 (HEAD-NORMALISATION) *There exists Γ, A, and Δ such that $\Gamma \vdash_{\mathrm{S}} M : \mathsf{A} \mid \Delta$, if and only if M has a head normal form.*

Proof: (*only if*): If $\Gamma \vdash_{\mathrm{S}} M : \mathsf{A} \mid \Delta$, then, by Thm. 17, there exists an $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_\perp A : \mathsf{A} \mid \Delta$. Then, by Def. 9, there exists N such that $M \rightarrow^*_{\beta\mu} N$ and $A \sqsubseteq N$. Since $\mathsf{A} \neq \omega$, $A \not\equiv \perp$, so we know that A is either $xA_1 \cdots A_n$ $(n \geq 0)$, $\lambda x.A'$, or $\mu\alpha.[\beta]A'$ with $A' \neq \mu\gamma.[\delta]A''$. Since $A \sqsubseteq N$, N is either $xM_1 \cdots M_n$ $(n \geq 0)$, $\lambda x.P$, or $\mu\alpha.[\beta]P$ with $P \neq \mu\gamma.[\delta]Q$. Then N is in head-normal from and M has a head-normal form.

(*if*): If M has a head-normal form, then there exists N such that $M \rightarrow^*_{\beta\mu} N$ and either:

$(N \equiv xM_1 \cdots M_n)$: Take $\Gamma = x{:}\omega\times\cdots\times\omega\times\Omega\rightarrow v$ (with n times ω) and $\mathsf{A} = \Omega\rightarrow v$.

$(N \equiv \lambda x.P)$: Since P is in head-normal form, by induction there are Γ', C, v, and Δ' such that $\Gamma' \vdash_{\mathrm{S}} P : \mathsf{C}\rightarrow v \mid \Delta'$. If $x{:}\mathsf{S} \in \Gamma'$, take $\Gamma = \Gamma' \backslash x$, and $\mathsf{A} = \mathsf{S}\times\mathsf{C}\rightarrow v$; otherwise take $\Gamma = \Gamma'$ and $\mathsf{A} = \omega\times\mathsf{C}\rightarrow v$. In either case, by rule (Abs), $\Gamma \vdash_{\mathrm{S}} \lambda x.P : \mathsf{A} \mid \Delta'$

$(N = \mu\alpha.[\alpha]P)$: Since P is in head-normal form, by induction there are Γ', C, D, v, and Δ' such that $\Gamma' \vdash_{\mathrm{S}} P : \mathsf{D}\rightarrow v \mid \alpha{:}\mathsf{C},\Delta'$. Take $\mathsf{C}' = \mathsf{C} \cap \mathsf{D}$, then by Thm. 8 (1) also $\Gamma' \vdash_{\mathrm{S}} P : \mathsf{D}\rightarrow v \mid \alpha{:}\mathsf{C}',\Delta'$, and since $\mathsf{C}' \leq_{\mathrm{S}} \mathsf{D}$, by rule (μ) we get $\Gamma' \vdash_{\mathrm{S}} \mu\alpha.[\alpha]P : \mathsf{C}'\rightarrow v \mid \Delta'$.

$(N = \mu\alpha.[\beta]P$, *with* $\alpha \neq \beta)$: Since P is in head-normal form, by induction there are C, C', D such that $\Gamma' \vdash_{\mathrm{S}} P : \mathsf{D}\rightarrow v \mid \alpha{:}\mathsf{C},\beta{:}\mathsf{C}',\Delta$ and $\mathsf{C}' \leq_{\mathrm{S}} \mathsf{D}$. Take $\mathsf{C}'' = \mathsf{C}' \cap \mathsf{D}$, then by Thm. 8 (1) also $\Gamma' \vdash_{\mathrm{S}} P : \mathsf{D}\rightarrow v \mid \alpha{:}\mathsf{C},\beta{:}\mathsf{C}'',\Delta$, and since $\mathsf{C}'' \leq_{\mathrm{S}} \mathsf{D}$ we get $\Gamma' \vdash_{\mathrm{S}} \mu\alpha.[\beta]P : \mathsf{C}'\rightarrow v \mid \beta{:}\mathsf{C}'',\Delta'$ by (μ').

Notice that in all cases, $\Gamma \vdash_{\mathrm{S}} N : \mathsf{A} \mid \Delta$, for some A, and by Thm. 8(2), $\Gamma \vdash_{\mathrm{S}} M : \mathsf{A} \mid \Delta$. □

5 Type assignment for (strong) normalisation

In this section we show the characterisation of both normalisation and strong normalisation, for which we first define a notion of derivability obtained from $\vdash_{\mathrm{S}}$ by restricting the use of the type assignment rule $(\cap)$ to at least two sub-derivations, thereby eliminating the possibility to assign ω to a term.

Definition 19 (SN TYPE ASSIGNMENT) 1. We define the ω-free types by the grammar:

$$\begin{array}{rcl} A,B & ::= & C \rightarrow v \\ R,S,T & ::= & A_1 \cap \cdots \cap A_n \quad (n \geq 1) \\ C,D & ::= & \Omega \mid S \times C \end{array}$$

2. *SN type assignment* is defined using the natural deduction system of Def. 6, but allowing only ω-free types, so restricting rule $(\cap)$ to:

$$(\cap): \quad \frac{\Gamma \vdash M : A_i \mid \Delta \quad (\forall i \in \underline{n})}{\Gamma \vdash M : \cap_{\underline{n}} A_i \mid \Delta} \; (n \geq 2)$$

We write $\Gamma \vdash_{\mathrm{SN}} M : S \mid \Delta$ if this judgement is derivable using this system.

Notice that the only real change in the system compared to $\vdash_{\mathrm{S}}$ is that ω is no longer an intersection type, so in rule $(\cap)$, the empty intersection ω is excluded.[4]

The following properties hold:

Lemma 20 *1. If $S \leq T$, then $S = \cap_I A_i$, $T = \cap_J B_j$, and for every $j \in J$ there exists $i \in I$ such that $A_i = B_j$.*

2. $\Gamma, x{:}S \vdash_{\mathrm{SN}} x : T \mid \Delta$, if and only if $S \leq_{\mathrm{S}} T$.

3. $\Gamma \vdash_{\mathrm{SN}} M : S \mid \Delta \Rightarrow \{x{:}T \in \Gamma \mid x \in fv(M)\} \vdash_{\mathrm{SN}} M : S \mid \{\alpha{:}C \in \Delta \mid \alpha \in fn(M)\}$.

4. $\Gamma \vdash_{\mathrm{SN}} M : S \mid \Delta \;\&\; \Gamma' \supseteq \Gamma \;\&\; \Delta' \supseteq \Delta \Rightarrow \Gamma' \vdash_{\mathrm{SN}} M : S \mid \Delta'$.

5. $\mathcal{D} :: \Gamma \vdash_{\mathrm{SN}} M : S \mid \Delta \;\Rightarrow\; \mathcal{D} :: \Gamma \vdash_{\mathrm{S}} M : S \mid \Delta$.

As for $\vdash_{\mathrm{S}}$, we can show that $(\leq_{\mathrm{S}})$ is an admissible rule in $\vdash_{\mathrm{SN}}$.

Lemma 21 *If $\Gamma \vdash_{\mathrm{SN}} M : S \mid \Delta$, and Γ', T, and Δ' are all ω-free and satisfy $\Gamma' \leq_{\mathrm{S}} \Gamma$, $\Delta' \leq_{\mathrm{S}} \Delta$, and $S \leq_{\mathrm{S}} T$, then $\Gamma' \vdash_{\mathrm{SN}} M : T \mid \Delta'$.*

Proof: Much the same as the proof for Thm. 8(1) in [7]. □

The following lemma shows a (limited) subject expansion result for $\vdash_{\mathrm{SN}}$: it states that if a contraction of a redex is typeable, then so is the redex, provided that the operand N is typeable in its own right; since N might not appear in the contractum, we need to assume that separately. Notice that we demand that N is typeable in the same contexts as the redex itself; this property would not hold once we consider contextual closure (in particular, when the reduction takes place under an abstraction); it might be that free names or variables in N get bound in the context.

Lemma 22 *If $\Gamma \vdash_{\mathrm{SN}} M[N{\cdot}\gamma/\alpha] : T \mid \gamma{:}C, \Delta$ and $\Gamma \vdash_{\mathrm{SN}} N : B \mid \Delta$, then there exists S such that $\Gamma \vdash_{\mathrm{SN}} M : T \mid \alpha{:}S \times C, \Delta$ and $\Gamma \vdash_{\mathrm{SN}} N : S \mid \Delta$.*

Proof: By nested induction; the outermost is on the structure of types, and the innermost on the structure of terms. We only show:

$(M \equiv x)$: Then $x[N{\cdot}\gamma/\alpha] = x$. Take $S = B$, then by Lem. 20, also $\Gamma \vdash_{\mathrm{SN}} x : C' \rightarrow v \mid \alpha{:}S \times C, \Delta$.

All other cases follow by induction. □

[4]With the aim of the characterisation of strong normalisation, it would have sufficed to only restrict rule $(\cap)$; we restrict the set of types as well in order to be able to characterise normalisation as well.

To prepare the characterisation of terms by their assignable types, we first prove that a term in $\lambda\mu\bot$-normal form is typeable without ω, if and only if it does not contain $\bot$. This forms the basis for the result that all normalisable terms are typeable without ω. Notice that the first result is stated for $\vdash_{\mathrm{S}}$.

Lemma 23 1. *If $\Gamma \vdash_{\mathrm{S}} A : \mathsf{A} \mid \Delta$, and Γ, A, and Δ are ω-free, then A is $\bot$-free.*

2. *If A is $\bot$-free, then there are Γ, A, and Δ, such that $\Gamma \vdash_{\mathrm{SN}} A : \mathsf{A} \mid \Delta$.*

Now, as also shown in [1], it is possible to characterise normalisable terms.

Theorem 24 (CHARACTERISATION OF NORMALISATION) *There exists ω-free Γ, Δ, and A such that $\Gamma \vdash_{\mathrm{S}} M : \mathsf{A} \mid \Delta$, if and only if M has a normal form.*

Proof: $(\Rightarrow)$: If $\Gamma \vdash_{\mathrm{S}} M : \mathsf{A} \mid \Delta$, by Thm. 17 there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\mathrm{S}} A : \mathsf{A} \mid \Delta$. Since Γ, A, and Δ are ω-free, by Lem. 23(1), this A is $\bot$-free. By Def. 9 there exists N such that $M \to^*_{\beta\mu} N$ and $A \sqsubseteq N$. Since A contains no $\bot$, $A \equiv N$, so N is a normal form, so M has a normal form.

$(\Leftarrow)$: If N is the normal form of M, then it is a $\bot$-free approximate normal form. By Lem. 23(2) there are Γ, A, and Δ such that $\Gamma \vdash_{\mathrm{SN}} N : \mathsf{S} \mid \Delta$. By Lem. 20(5) also $\Gamma \vdash_{\mathrm{S}} N : \mathsf{S} \mid \Delta$, and by Thm. 8(2), $\Gamma \vdash_{\mathrm{S}} M : \mathsf{S} \mid \Delta$, and Γ, S, and Δ are ω-free. □

In [7] it is shown that it is possible to characterise the set of all terms that are strongly normalisable with respect to $\to_{\beta\mu}$, using Thm. 8(4), and the proof for the property that all terms in normal form can be typed in $\vdash_{\mathrm{SN}}$, a property that follows here from Lem. 23 (see the proof of the previous result). Other than that, the proof is identical.

The following lemma shows that $\vdash_{\mathrm{SN}}$ is closed under the expansion of redexes (notice that the result is not stated for arbitrary reduction steps, but only for terms that are proper redexes).

Lemma 25 1. *If $\Gamma \vdash_{\mathrm{SN}} M[N/x] : \mathsf{A} \mid \Delta$ and $\Gamma \vdash_{\mathrm{SN}} N : \mathsf{B} \mid \Delta$, then $\Gamma \vdash_{\mathrm{SN}} (\lambda x.M)N : \mathsf{A} \mid \Delta$.*

2. *If $\Gamma \vdash_{\mathrm{SN}} \mu\gamma.[\gamma]P[Q{\cdot}\gamma/\beta]Q : \mathsf{A} \mid \Delta$ and $\Gamma \vdash_{\mathrm{SN}} Q : \mathsf{B} \mid \Delta$, then $\Gamma \vdash_{\mathrm{SN}} (\mu\beta.[\beta]P)Q : \mathsf{A} \mid \Delta$.*

3. *If $\Gamma \vdash_{\mathrm{SN}} \mu\gamma.[\delta]P[Q{\cdot}\gamma/\beta] : \mathsf{A} \mid \Delta$ (with $\beta \neq \delta$) and $\Gamma \vdash_{\mathrm{SN}} Q : \mathsf{B} \mid \Delta$, then $\Gamma \vdash_{\mathrm{SN}} (\mu\beta.[\delta]P)Q : \mathsf{A} \mid \Delta$.*

4. *If $\Gamma \vdash_{\mathrm{SN}} \mu\alpha.([\delta]P)[\beta/\gamma] : \mathsf{A} \mid \Delta$, then $\Gamma \vdash_{\mathrm{SN}} \mu\alpha.[\beta]\mu\gamma.[\delta]P : \mathsf{A} \mid \Delta$.*

Thm. 28 below shows that the set of strongly normalisable terms is exactly the set of terms typeable in the intersection system without using the type constant ω. The proof goes by induction on the leftmost outermost reduction path. First we introduce the notion of leftmost, outer-most reduction.

Definition 26 An occurrence of a redex R in a term M is called the *leftmost, outermost redex of M* (*lor*(M)), if:

1. There is no redex R' in M such that $\mathsf{R}' = \mathsf{C}[\mathsf{R}]$ with $\mathsf{C}[-] \neq [-]$ (*outer-most*);
2. There is no redex R' in M such that $M = \mathsf{C}_0[\mathsf{C}_1[\mathsf{R}']\mathsf{C}_2[\mathsf{R}]]$ (*leftmost*).

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

The following lemma formulates a subject expansion result for $\vdash_{\mathrm{SN}}$ with respect to left-most outer-most reduction.

Lemma 27 *Assume $M \to_{lor} N$, and $\Gamma \vdash_{\mathrm{SN}} N : \mathsf{C}{\to}v \mid \Delta$; if lor$(M) = PQ$ also assume that $\Gamma_0 \vdash_{\mathrm{SN}} Q : \mathsf{B} \mid \Delta_0$. Then there exists Γ',Δ',C' such that $\Gamma' \vdash_{\mathrm{SN}} M : \mathsf{C}'{\to}v \mid \Delta'$.*

We can now show that all strongly normalisable terms are exactly those typeable in $\vdash_{\mathrm{SN}}$.

Theorem 28 $\exists \Gamma, \Delta, A \; [\Gamma \vdash_{\mathrm{SN}} M : A \mid \Delta] \iff$ *M is strongly normalisable with respect to* $\rightarrow_{\beta\mu}$.

Proof: $(\Rightarrow)$: If $\mathcal{D} :: \Gamma \vdash_{\mathrm{SN}} M : A \mid \Delta$, then by Lem. 20(5), also $\mathcal{D} :: \Gamma \vdash_{\mathrm{S}} M : A \mid \Delta$. Then, by Thm. 8(4), $\mathcal{D}$ is strongly normalisable with respect to $\rightarrow_{\mathrm{DER}}$. Since $\mathcal{D}$ contains no ω, all redexes in M correspond to redexes in $\mathcal{D}$, a property that is preserved by derivation reduction (it does not introduce ω). So also M is strongly normalisable with respect to $\rightarrow_{\beta\mu}$.

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

a. If $\#M = 0$, then M is in normal form, and by Lem. 23(2), there exist Γ, Δ and A such that $\Gamma \vdash_{\mathrm{SN}} M : A \mid \Delta$.

b. If $\#M \geq 1$, so M contains a redex, then let $M \rightarrow_{lor} N$ by contracting the redex PQ. Then $\#N < \#M$, and $\#Q < \#M$ (since Q is a proper sub-term of a redex in M), so by induction, for some Γ, Γ', Δ, Δ', A, and B, we have $\Gamma \vdash_{\mathrm{SN}} M : A \mid \Delta$ and $\Gamma' \vdash_{\mathrm{SN}} Q : B \mid \Delta'$. Then, by Lem. 27, there exist Γ_1, Δ_1, C such that $\Gamma_1 \vdash_{\mathrm{SN}} M : C \mid \Delta_1$. If the redex is $\mu\alpha.[\beta]\mu\gamma.[\delta]P$, then $\#\mu\alpha.[\beta]\mu\gamma.[\delta]P > \#\mu\alpha.([\delta]P)[\beta/\gamma]$, so the result follows by induction. $\square$

Conclusions

We have studied a strict version of the intersection type system for $\lambda\mu$ of [10]. Using the fact that derivation reduction (a kind of cut-elimination) is strongly normalisable, we have shown an approximation theorem, and from that given a characterisation of head normalisation. We have also shown that the system without the type constant ω characterises the strongly normalisable terms and that we can characterise normalisation as well.

References

[1] 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`.

[2] S. van Bakel (1995): *Intersection Type Assignment Systems*. *Theoretical Computer Science* 151(2), pp. 385–435, doi:`10.1016/0304-3975(95)00073-6`.

[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 (2008): *The Heart of Intersection Type Assignment; Normalisation proofs revisited*. *Theoretical Computer Science* 398, pp. 82–94, doi:`10.1016/j.tcs.2008.01.020`.

[5] 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`.

[6] 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`.

[7] S. van Bakel (2016): *Approximation and (Head) Normalisation for $\lambda\mu$ using Strict Intersection Types*. Available at `http://www.doc.ic.ac.uk/~svb/Research/Papers/Lmu-Strict.pdf`.

[8] 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`.

[9] S. van Bakel, F. Barbanera & U. de'Liguoro (2012): *Characterisation of Strongly Normalising $\lambda\mu$-Terms*. In: *Proceedings of 6th International Workshop* Intersection Types and Related Systems *(ITRS'12), Dubrovnik, Croatia, June 29th*, *Electronic Proceedings in Theoretical Computer Science* 121, pp. 31–44, doi:`10.4204/EPTCS.121.1`.

[10] S. van Bakel, F. Barbanera & U. de'Liguoro (2015): *Intersection types for $\lambda\mu$*. *Logical Methods in Computer Science*. To appear.

[11] S. van Bakel & M.G. Vigliotti (2014): *A fully abstract semantics of $\lambda\mu$ in the π-calculus*. In: *Proceedings of Sixth International Workshop on Classical Logic and Computation 2014* (CL&C'14)*, Vienna, Austria*, *Electronic Proceedings in Theoretical Computer Science* 164, pp. 33–47, doi:`10.4204/EPTCS.164.3`.

[12] H. Barendregt (1984): *The Lambda Calculus: its Syntax and Semantics*, revised edition. North-Holland, Amsterdam, doi:`10.2307/2274112`.

[13] 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`.

[14] C. Böhm (1968): *Alcune propietá delle forme $\beta\eta$-normali nel λk-calcolo*. *Pubblicazioni 696, Instituto Nazionale per le Applicazioni del Calcolo. Roma.*

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

[16] H.B. Curry & R. Feys (1958): *Combinatory Logic*. 1, North-Holland, Amsterdam.

[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] U. de'Liguoro (2016): *The Approximation Theorem for the $\Lambda\mu$-Calculus*. *Mathematical Structures in Computer Science* FirstView, pp. 1–21, doi:`10.1017/S0960129515000286`.

[19] 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`.

[20] W. Py (1998): *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie.

[21] S. Ronchi Della Rocca & B. Venneri (1984): *Principal type schemes for an extended type theory*. *Theoretical Computer Science* 28, pp. 151–169, doi:`10.1016/0304-3975(83)90069-5`.

[22] A. Saurin (2010): *Standardization and Böhm Trees for $\lambda\mu$-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`.

[23] Th. Streicher & B. Reus (1998): *Classical logic: Continuation Semantics and Abstract Machines*. *Journal of Functional Programming* 11(6), pp. 543–572, doi:`10.1007/BFb0026995`.

[24] C.P. Wadsworth (1976): *The Relation Between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus*. *SIAM Journal on Computing* 5(3), pp. 488–521, doi:`10.1137/0205036`.