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

(8th International Workshop on Intersection Types and Related Systems, EPTCS 242, 2016)

Steffen van Bakel

Department of Computing, Imperial College London, 180 Queen’s Gate, London SW7 2BZ, UK
s.vanbakel@imperial.ac.uk

Extended Abstract

Abstract

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 $\lambda$-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 $\to$. This simple extension allows for a great leap in complexity: not only can a (filter) model be built for the $\lambda$-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 \( \omega \) 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 result. For that, we define a notion of approximation for \( \lambda \mu \), and show that this yields a semantics (Thm. 3.5). We then show that for every typeable term there exists an approximant of that term that can be assigned exactly the same types (Thm. 4.4). We then show that this approximation result naturally gives a characterisation of head normalisation (Thm 4.5), as well as a characterisation of normalisation (Thm 5.6). We also revisit the proof of characterisation of strong normalisation of terms through the assignable types (Thm 5.10), 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 \( n \) for the set \( \{1, \ldots, n\} \) and use a vector notation for the abbreviation of sequences, so write \( X_n \) 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 \( \lambda \)-calculus obtained by adding names and a name-abstraction operator \( \mu \) and was intended as a proof calculus for a fragment of classical logic. Derivable statements have the shape \( \Gamma \vdash M : A | \Delta \), where \( A \) is the main (active) conclusion of the statement, and \( \Delta \) contains the alternative conclusions, consisting of pairs of names and types; the left-hand context \( \Gamma \), as usual, is a mapping from term variables to types, and represents the assumptions about free variables of \( M \).

**Definition 1.1 (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:

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

As usual, we consider \( \lambda \) and \( \mu \) to be binders; the sets \( \text{fv}(M) \) and \( \text{fn}(M) \) 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 1.2 (Substitution [19])** Substitution takes two forms:

- **term substitution:** \( M[N/x] \) (\( N \) is substituted for \( x \) in \( M \))
- **structural substitution:** \( M[L, \gamma/\alpha] \) (every ‘subterm’ \( [\alpha] N \) of \( M \) is replaced by \( [\gamma] NL \))

As usual, both substitutions are capture avoiding, using \( \alpha \)-conversion when necessary.

**Definition 1.3 (Reduction [19])** Reduction in \( \lambda \mu \) is based on the following rules:
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.

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

$$H ::= xM_1 \cdots M_n \quad (n \geq 0)$$

$$| \lambda x.H$$

$$| \mu x.\beta H \quad (H \neq \mu\gamma.\delta H')$$

## 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 $\delta$ and streams by continuation types $\pi$ that are of the shape $\delta_1 \times \cdots \times \delta_n \times \omega$, so essentially is a sequence of $\delta$s. This later [9] was followed by the proof that, as for the $\lambda$-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 $\Omega$ rather than $\omega$ 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 2.1 (Strict Types [7])** i) Let $\nu$ range over a countable, infinite set of type constants.

We define our strict types by the grammar:

\[(\beta): (\lambda x.M)N \rightarrow M[N/x] \quad \text{(logical reduction)}\]

\[(\mu): \begin{cases} (\mu\beta.\beta P)Q \rightarrow \mu\gamma.\gamma(P[Q/\gamma/\beta])Q \\ (\mu\beta.\beta P)Q \rightarrow \mu\gamma.\gamma[P[Q/\gamma/\beta]], \quad \text{if} \ \delta \neq \beta \end{cases} \quad \text{(structural reduction)}\]

\[(\text{REN}): \begin{cases} \mu x.\beta \mu\gamma.\gamma M \rightarrow \mu x.\beta M[\beta/\gamma] \\ \mu x.\beta \mu\gamma.\gamma M \rightarrow \mu x.\beta M[\beta/\gamma], \quad \text{if} \ \delta \neq \gamma \end{cases} \quad \text{(renaming)}\]

0 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 $\beta$ disappears during reduction, and through $\alpha$-conversion can be picked as name for the newly created applications instead of $\gamma$. But, in fact, this is not the same $\beta$ (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.
\[
\begin{align*}
A, B &::= C \rightarrow v & \text{basic types} \\
R, S, T &::= \omega \mid A_1 \cap \cdots \cap A_n \quad (n \geq 1) & \text{intersection types} \\
C, D &::= \Omega \mid S \times C & \text{continuation types}
\end{align*}
\]

**ii)** On strict types, the type inclusion relation \( \leq_s \) is the smallest partial order satisfying the rules:

\[
\begin{align*}
A_1 \cap \cdots \cap A_n \leq A_j & (j \in \mathbb{N}, n \geq 1) \\
S \leq A_i & (\forall i \in \mathbb{N}) \\
S \leq \Omega & \\
S \leq \Omega & \\
S \leq T & C \leq D \\
S \leq A_1 \cap \cdots \cap A_n & (n \geq 0)
\end{align*}
\]

For convenience, we will write \( \cap_i A_i \) for \( A_{i_1} \cap \cdots \cap A_{i_n} \) where \( I = \{ i_1, \ldots, i_n \} \) and the notions \( \sqcap \) and the notions \( \sqcup \) are defined similarly.

Notice that for any continuation type \( C \) there are \( n \geq 0 \) and \( S_i \ (i \in \mathbb{N}) \) such that \( C = S_1 \times \cdots \times S_n \times \Omega \).

**Definition 2.2 (Strict Type Assignment [7])**

- **i)** A variable context \( \Gamma \) 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.

- **ii)** We write \( \Gamma, x : S \) for the context defined by:
  \[
  \Gamma, x : S \equiv \Gamma \cup \{ x : S \}, \quad \text{if } \Gamma \text{ is not defined on } x \\
  \Gamma \equiv \Gamma, \quad \text{if } x : S \in \Gamma
  \]

  We write \( x \not\in \Gamma \) if there exists no \( S \) such that \( x : S \in \Gamma \).

- **iii)** Name contexts \( \Delta \) and the notions \( \alpha : C \), \( \Delta \) and \( \alpha \not\in \Delta \) are defined in a similar way.

- **iv)** We define strict type assignment for \( \lambda \mu \)-terms through the following natural deduction system:

  \[
  \begin{align*}
  (Ax) &::= \frac{\Gamma, x : S \vdash x : A \mid \Delta \quad (S \leq_s A)}{\Gamma \vdash M : A \mid \Delta} \\
  (\cap) &::= \frac{\Gamma \vdash M : A_i \mid \Delta \quad (\forall i \in I)}{\Gamma \vdash M : \sqcap_i A_i \mid \Delta} (I = \emptyset \lor |I| \geq 2) \\
  (Abs) &::= \frac{\Gamma, x : S \vdash M : C \rightarrow v \mid \Delta \quad (x \not\in \Gamma)}{\Gamma \vdash \lambda x.M : S \times C \rightarrow v \mid \Delta} \\
  (\mu) &::= \frac{\Gamma \vdash M : D \rightarrow v \mid \alpha : C, \Delta \quad (\alpha \not\in \Delta, C \leq D)}{\Gamma \vdash \mu \alpha. [\alpha]M : C \rightarrow v \mid \Delta} \\
  (App) &::= \frac{\Gamma \vdash M : S \times C \rightarrow v \mid \Delta \quad \Gamma \vdash N : S \mid \Delta \quad (\mu')} {\Gamma \vdash MN : C \rightarrow v \mid \Delta} \\
  \end{align*}
  \]

  We write \( \Gamma \vdash_s M : S \mid \Delta \) for judgements derivable using these rules, and prefix this with ‘\( \vdash \)’ if we want to name the derivation.

- **v)** The relation \( \leq_s \) is naturally extended to variable contexts as follows:
  \[
  \Gamma \leq_s \Gamma' \equiv \forall x : S \in \Gamma' \exists x : T \in \Gamma \quad [T \leq_s S];
  \]

  \( \Delta \leq_s \Delta' \) is defined similarly.

**Definition 2.3** By abuse of notation, we allow the notation \( S \cap T \), where \( S = \sqcap_i A_i \) and \( T = \sqcap_j B_j \), which stands for \( A_1 \cap \cdots \cap A_n \cap B_1 \cap \cdots \cap B_m \). Given two contexts \( \Gamma_1 \) and \( \Gamma_2 \), we define the context \( \Gamma_1 \cap \Gamma_2 \) as follows:

\[
\Gamma_1 \cap \Gamma_2 \equiv \{ x : S \cap S_2 \mid x : S_1 \in \Gamma_1 \& x : S_2 \in \Gamma_2 \} \cup \{ x : S \mid x : S \in \Gamma_1 \& x \not\in \Gamma_2 \} \cup \{ x : S \mid x : S \in \Gamma_2 \& x \not\in \Gamma_1 \}
\]

and write \( \sqcap \Gamma_i \) for \( \sqcap \Gamma_1 \cap \cdots \cap \Gamma_n \). We will also allow intersection of continuation types as shorthand notation: let \( D = S_1 \times \cdots \times S_n \times \Omega \), and \( C = T_1 \times \cdots \times T_m \times \Omega \) and assume, that \( n < m \);
we define
\[ D \cap C \triangleq S_1 \cap T_1 \times \cdots \times S_n \cap T_n \times T_{n+1} \times \cdots \times T_m \times \Omega. \]
(we need this notion in the proof of Thm. 4.5). 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 \( \rightarrow_{\text{Der}} \), where only those redexes in terms that are typed with a type different from \( \omega \); it shows that this notion is strongly normalisable, which then leads to the proof that all terms typeable in a restriction of \( \vdash \) that eliminates the type constant \( \omega \), are strongly normalisable.

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

**Theorem 2.4 ([7])**
1. If \( \Gamma \vdash_s M : S \mid \Delta, \Gamma' \leq_s \Gamma, \Delta' \leq_s \Delta,^1 \) and \( S \leq_s T \), then \( \Gamma' \vdash_s M : T \mid \Delta' \).
2. If \( \Gamma \vdash_s M : A \mid \Delta \) and \( M =_{\beta\mu} N \), then \( \Gamma \vdash_s N : A \mid \Delta \).
3. Let \( D :: \Gamma \vdash_s M : S \mid \Delta \), and \( D \rightarrow_{\text{Der}}^s D' :: \Gamma \vdash_s N : S \mid \Delta \), then \( M \rightarrow_{\beta\mu}^s N \).
4. If \( D :: \Gamma \vdash_s M : S \mid \Delta \), then \( SN(D) \) (\( D \) 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 \( \mu \)-binding are separated [17]) has been studied by others as well [22, 18]; weak approximants for \( \lambda\mu \) are studied in [11].

**Definition 3.1 (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 \( A \) with respect to \( \rightarrow_{\beta\mu} \) is defined through the grammar:
   \[ A ::= \perp \mid xA_1 \cdots A_n \quad (n \geq 0) \]
   \[ \mid \lambda x. A \quad (A \neq \perp) \]
   \[ \mid \mu x. [\beta] A \quad (A \neq \mu \gamma [\delta] A', A \neq \perp) \]
3. The relation \( \subseteq \subseteq \lambda\mu \perp^2 \) is the smallest preorder that is the compatible extension of \( \perp \subseteq M \).
4. The set of approximants of \( M, A(M) \), is defined as
   \[ A(M) \triangleq \{ A \in A \mid \exists N \in \lambda\mu \ [M \rightarrow_{\beta\mu}^s N \& A \subseteq N \} \} \]
5. Approximation equivalence between terms is defined through: \( M \sim_A N \triangleq A(M) = A(N) \).

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

**Lemma 3.2**
1. If \( A \subseteq M \) and \( M \rightarrow_{\beta\mu}^s N \), then \( A \subseteq N \).
2. \( H \) is a head-normal form if and only if there exists \( A \in A \) such that \( A \subseteq H \) and \( A \neq \perp \).

---

^1 The condition \( \Delta' \leq_s \Delta \) 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 \( A \cap B \times \Omega \leq A \times \Omega \); obviously we have \( \alpha : A \cap B \times \Omega \leq A \times \Omega \) and \( \neg A \leq \neg A \cup \neg B \).
The following definition introduces an operation of join on $\lambda\mu\bot$-terms.

**Definition 3.3 (Join, Compatible Terms)** i) The partial mapping $\sqcup : \lambda\mu\bot^2 \to \lambda\mu\bot$, is defined by:

\[
\begin{align*}
\bot \sqcup M & \equiv M \sqcup \bot \equiv M \\
\bot \sqcup x & \equiv x \\
(\lambda x.M) \sqcup (\lambda x.N) & \equiv \lambda x.(M \sqcup N) \\
(\mu a.[\beta]M) \sqcup (\mu a.[\beta]N) & \equiv \mu a.[\beta](M \sqcup N) \\
(M_1M_2) \sqcup (N_1N_2) & \equiv (M_1 \sqcup N_1)(M_2 \sqcup N_2)^2
\end{align*}
\]

ii) 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 $\sqcup_M$ for the term $M_1 \sqcup \cdots \sqcup M_n$. Note that $\bot$ can be defined as the empty join, i.e. if $M \equiv \sqcup_M \bot$, then $M \equiv \bot$.

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

**Lemma 3.4** i) If $P \sqsubseteq M$, and $Q \sqsubseteq M$, then $P \sqcup Q$ is defined, and:

\[
P \sqsubseteq P \sqcup Q, \quad Q \sqsubseteq P \sqcup Q, \quad \text{and} \quad P \sqcup Q \sqsubseteq M.
\]

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

We can also define $\mathcal{I}M = \sqcup \{ A \mid A \in \mathcal{A}(M) \}$ (which by the previous lemma is well defined); then $\mathcal{I}$ 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 3.5 (Approximation Semantics for $\lambda\mu$)** If $M =_{\beta\mu} N$, then $M \sim_A N$.

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

\[
(\mathcal{A}(M) \subseteq \mathcal{A}(N)) : \text{If } A \in \mathcal{A}(M), \text{ then there exists } L \text{ such that } M \to_{\beta\mu}^* L \text{ and } A \sqsubseteq L. \text{ Since } \to_{\beta\mu}
\]

is Church-Rosser, there exists $R$ such that $L \to_{\beta\mu}^* R$ and $N \to_{\beta\mu}^* R$, so also $M \to_{\beta\mu}^* R$. Then

by Lem. 3.2, $A \sqsubseteq R$, and since $N \to_{\beta\mu}^* R$, we have $A \in \mathcal{A}(N)$.

\[
(\mathcal{A}(N) \subseteq \mathcal{A}(M)) : \text{If } A \in \mathcal{A}(N), \text{ then there exists } L \text{ such that } N \to_{\beta\mu}^* L \text{ and } A \sqsubseteq L. \text{ But then also } M \to_{\beta\mu}^* L, \text{ so } A \in \mathcal{A}(M).
\]

The reverse implication of this result does not hold, since terms without head-normal form (which have only $\bot$ 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 $M$, $\Gamma$, $S$, and $\Delta$ 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 $\omega$-occurrences). Another result is the well-known characterisation of strong normalisation of

---

2 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.
typeable $\lambda\mu$-terms, i.e., all terms, typeable in $\vdash_\beta$ without using the rule $(\cap)$ with $I = \emptyset$, are strongly normalisable.

First we give some auxiliary definitions and results.

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

First we show that $\vdash_\beta$ is closed for $\sqsubseteq$.

**Lemma 4.1** $\Gamma \vdash_\beta M : S \mid \Delta$ and $M \sqsubseteq N$ then $\Gamma \vdash_\beta N : S \mid \Delta$.

Next we define a notion of type assignment that is similar to that of Def. 2.2, but differs in that it assigns $\omega$ only to the term $\bot$.

**Definition 4.2** $\bot$-type assignment and $\bot$-derivations are defined as $\vdash_\beta$, with the exception of: 

$$
(\cap_\bot) : \frac{\Gamma \vdash M_i : A \mid \Delta \quad (\forall i \in \mathbf{n})}{\Gamma \vdash \bigsqcup_{i=1}^n M_i : \bigsqcup_{i=1}^n A \mid \Delta} (n = 0 \lor n \geq 2)
$$

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

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

**Lemma 4.3**

i) If $D :: \Gamma \vdash_\bot M : S \mid \Delta$, then $D :: \Gamma \vdash_\beta M : S \mid \Delta$.

ii) If $D :: \Gamma \vdash_\beta M : S \mid \Delta$, then there exists $M' \sqsubseteq M$ such that $D :: \Gamma \vdash_\bot M' : S \mid \Delta$.

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

Using Thm. 2.4(iv) and Lem. 4.3, 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 4.4** (Approximation) $\Gamma \vdash_\beta M : S \mid \Delta \iff \exists A \in \mathcal{A}(M) \; [\Gamma \vdash_\beta A : S \mid \Delta]$.

Proof: $(\Rightarrow)$: If $D :: \Gamma \vdash_\beta M : S \mid \Delta$, then, by Thm. 2.4(iv), $SN(D)$. Let $D' :: \Gamma \vdash_\bot N : S \mid \Delta$ be a normal form of $D$ with respect to $\rightarrow_{Der}$, then by Thm. 2.4(iii), $M \rightarrow^*_\beta N$ and, by Lem. 4.3(ii), there exists $N' \sqsubseteq N$ such that $D' :: \Gamma \vdash_\bot N' : S \mid \Delta$. So, in particular, $N'$ contains no redexes (no redexes typed with a type different form $\omega$ since $D'$ is in normal form, and none typed with $\omega$ since only $\bot$ can be typed with $\omega$), so $N' \in \mathcal{A}$, and therefore $N' \in \mathcal{A}(M)$.

$(\Leftarrow)$: Let $A \in \mathcal{A}(M)$ be such that $\Gamma \vdash_\beta A : S \mid \Delta$. Since $A \in \mathcal{A}(M)$, there exists $N$ such that $M \rightarrow^*_\beta N$ and $A \sqsubseteq N$. Then, by Lem. 4.1, $\Gamma \vdash_\beta N : S \mid \Delta$, and, by Thm. 2.4(ii), also $\Gamma \vdash_\beta M : S \mid \Delta$.

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

**Theorem 4.5** (Head-normalisation) There exists $\Gamma$, $A$, and $\Delta$ such that $\Gamma \vdash_\beta M : A \mid \Delta$, if and only if $M$ has a head normal form.

Proof: (only if): If $\Gamma \vdash_\beta M : A \mid \Delta$, then, by Thm. 4.4, there exists an $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_\bot A : A \mid \Delta$. Then, by Def. 3.1, there exists $N$ such that $M \rightarrow^*_\beta N$ and $A \sqsubseteq N$. Since $A \neq \omega$, $A \neq \bot$, so we know that $A$ is either $x A_1 \cdots A_n \ (n \geq 0)$, $\lambda x.A'$, or $\mu x.[\beta]A'$ with $A' \neq \mu y.[\delta]A''$. Since $A \sqsubseteq N$, $N$ is either $x M_1 \cdots M_n \ (n \geq 0)$, $\lambda x.P$, or $\mu x.[\beta]P$ with $P \neq \mu y.[\delta]Q$. 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 \( M \rightarrow^\#_M N \) and either:

\[ N \equiv xM_1 \cdots M_n : \Gamma \rightarrow x : \alpha \times \cdots \times \alpha \times \Omega \rightarrow v \text{ (with n times } \alpha) \] and \( A = \Omega \rightarrow v \).

\[ N \equiv \lambda x.P : \Gamma \rightarrow x : \alpha \rightarrow y : \alpha \rightarrow \alpha \rightarrow \Omega \rightarrow v \text{ (by the grammar:} \]

- Since P is in head-normal form, by induction there are \( \Gamma' \), \( \alpha \), and \( \Delta ' \) such that \( \Gamma' \vdash P : \alpha \rightarrow \alpha \rightarrow \Omega \rightarrow v \). If \( x : S \in \Gamma' \), take \( \Gamma = \Gamma' \backslash x \), and \( A = S \times \alpha \rightarrow \Omega \rightarrow v \); otherwise take \( \Gamma = \Gamma' \) and \( A = \alpha \times \Omega \rightarrow v \). In either case, by rule \( \text{(Abs)} \), \( \Gamma \vdash \lambda x.P : A \mid \Delta ' \).

\( N = \mu a.[\alpha]P \): Since P is in head-normal form, by induction there are \( \Gamma' \), \( C \), \( D \), and \( \Delta ' \) such that \( \Gamma' \vdash P : D \rightarrow v \mid \alpha : C \rightarrow \Delta ' \). Take \( C' = C \cap D \), then by Thm. 2.4(i) also \( \Gamma' \vdash P : D \rightarrow v \mid \alpha : C' \rightarrow \Delta ' \), and since \( C' \leq D \), by rule \( \mu \) we get \( \Gamma'' \vdash \mu a.[\alpha]P : C' \rightarrow v \mid \Delta ' \).

\( N = \mu a.[\beta]P \), with \( \alpha \neq \beta \): Since P is in head-normal form, by induction there are \( \alpha \), \( C' \), \( D \) such that \( \Gamma'' \vdash P : D \rightarrow v \mid \alpha : C' \rightarrow \Delta ' \). Take \( C'' = C' \cap D \), then by Thm. 2.4(i) also \( \Gamma'' \vdash P : D \rightarrow v \mid \alpha : C'' \rightarrow \Delta ' \), and since \( C'' \leq D \) we get \( \Gamma'' \vdash \mu a.[\beta]P : C' \rightarrow v \mid \beta : C'' \rightarrow \Delta ' \) by \( \mu \).

Notice that in all cases, \( \Gamma \vdash N : A \mid \Delta \), for some A, and by Thm. 2.4(ii), \( \Gamma \vdash M : A \mid \Delta \). □

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 \( \vdash_s \) by restricting the use of the type assignment rule \( (\cap) \) to at least two sub-derivations, thereby eliminating the possibility to assign \( \omega \) to a term.

**Definition 5.1** (SN type assignment) i) We define the \( \omega \)-free types by the grammar:

\[
A, B \ ::= \ C \rightarrow v \\
R, S, T \ ::= \ A_1 \cap \cdots \cap A_n \quad (n \geq 1) \\
C, D \ ::= \ \Omega \mid S \times C
\]

ii) SN type assignment is defined using the natural deduction system of Def. 2.2, but allowing only \( \omega \)-free types, so restricting \( (\cap) \) to:

\[
(\cap) : \quad \frac{\Gamma \vdash M : A_i \mid \Delta \quad (\forall i \in n)}{\Gamma \vdash M : \cap \_A_i \mid \Delta} \quad (n \geq 2)
\]

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

Notice that the only real change in the system compared to \( \vdash_s \) is that \( \omega \) is no longer an intersection type, so in rule \( (\cap) \), the empty intersection \( \omega \) is excluded.\(^3\)

The following properties hold:

**Lemma 5.2** i) If \( S \preceq T \), then \( S = \cap_1 A_i, T = \cap_1 B_j, \) and for every \( j \in J \) there exists \( i \in I \) such that \( A_i = B_j \).

ii) \( \Gamma, x : S \vdash_{SN} x : T \mid \Delta \), if and only if \( S \preceq_s T \).

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

iv) \( \Gamma \vdash_{SN} M : S \mid \Delta \& \& \Gamma' \supseteq \Gamma \& \& \Delta' \supseteq \Delta \Rightarrow \Gamma' \vdash_{SN} M : S \mid \Delta' \).

v) \( D :: \Gamma \vdash_{SN} M : S \mid \Delta \Rightarrow D :: \Gamma \vdash S : M \mid \Delta \).

As for \( \vdash_s \), we can show that \( \preceq_s \) is an admissible rule in \( \vdash_{SN} \).

\(^3\) 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.
Theorem 5.6 (Characterisation of Normalisation) There exists $\omega$-free $\Gamma$, $\Delta$, and $A$ such that $\vdash_{SN} M : A | \Delta$, if and only if $M$ has a normal form.

Proof: \((\Rightarrow)\): If $\vdash_{SN} M : A | \Delta$, by Thm. 4.4 there exists $A \in \mathcal{A}(M)$ such that $\vdash_{SN} A : A | \Delta$. Since $\Gamma$, $A$, and $\Delta$ are $\omega$-free, by Lem. 5.5(i), this $A$ is $\perp$-free. By Def. 3.1 there exists $N$ such that $M \rightarrow_{B}\beta_{\mu} N$ and $A \subseteq N$. Since $A$ contains no $\perp$, $A \equiv N$, 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 $\perp$-free approximate normal form. By Lem. 5.5(ii) there are $\Gamma$, $A$, and $\Delta$ such that $\vdash_{SN} N : S | \Delta$. By Lem. 5.2(v) also $\vdash_{SN} N : S | \Delta$, and by Thm. 2.4(ii), $\vdash_{SN} M : S | \Delta$, and $\Gamma$, $S$, and $\Delta$ are $\omega$-free.

\[\square\]

In [7] it is shown that it is possible to characterise the set of all terms that are strongly normalisable with respect to $\rightarrow_{B}\beta_{\mu}$, using Thm. 2.4(iv), and the proof for the property that all terms in normal form can be typed in $\vdash_{SN}$, a property that follows here from Lem. 5.5 (see the proof of the previous result). Other than that, the proof is identical.

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

Lemma 5.7 i) If $\vdash_{SN} M[N/x] : A | \Delta$ and $\vdash_{SN} N : B | \Delta$, then $\vdash_{SN} (\lambda x. M)N : A | \Delta$.

ii) If $\vdash_{SN} \mu \gamma. [\gamma] P[\gamma/\beta]Q : A | \Delta$ and $\vdash_{SN} Q : B | \Delta$, then $\vdash_{SN} (\mu \beta.[\beta] P)Q : A | \Delta$.

iii) If $\vdash_{SN} \mu \gamma.[\delta] P[\gamma/\beta] : A | \Delta$ (with $\beta \neq \delta$) and $\vdash_{SN} Q : B | \Delta$, then $\vdash_{SN} (\mu \beta.[\delta] P)Q : A | \Delta$. 

\[\square\]
Definition 5.8 An occurrence of a redex $R$ in a term $M$ is called the leftmost, outermost redex of $M$ (lor($M$)), if:

i) There is no redex $R'$ in $M$ such that $R' = C[R]$ with $C[\cdot] \neq \cdot$ (outer-most);

ii) There is no redex $R'$ in $M$ such that $M = C_0[C_1[R']C_2[R]]$ (leftmost).

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

Theorem 5.10 $\exists \Gamma, \Delta, A \ [\Gamma \vdash_{SN} M : A \mid \Delta] \iff M$ is strongly normalisable with respect to $\rightarrow_{SN}$.

Proof: ($\Rightarrow$): If $\Delta :: \Gamma \vdash_{SN} M : A \mid \Delta$, then by Lem. 5.2(v), also $\Delta :: \Gamma \vdash M : A \mid \Delta$. Then, by Thm. 2.4(iv), $\Delta$ is strongly normalisable with respect to $\rightarrow_{Der}$. Since $\Delta$ contains no $\omega$, all redexes in $M$ correspond to redexes in $\Delta$, a property that is preserved by derivation reduction (it does not introduce $\omega$). So also $M$ is strongly normalisable with respect to $\rightarrow_{SN}$.

($\Leftarrow$): 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. 5.5(ii), there exist $\Gamma$, $\Delta$, and $A$ such that $\Gamma \vdash_{SN} M : A \mid \Delta$.

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 $\Gamma$, $\Delta$, $\Delta'$, $A$, and $B$, we have $\Gamma \vdash_{SN} M : A \mid \Delta$ and $\Gamma' \vdash_{SN} Q : B \mid \Delta'$. Then, by Lem. 5.9, there exist $\Gamma_1, \Delta_1, C$ such that $\Gamma_1 \vdash_{SN} M : C \mid \Delta_1$. 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.

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 $\omega$ characterises the strongly normalisable terms and that we can characterise normalisation as well.

References

A Note on the Entscheidungsproblem


On the Relation between the Ph. de Groote (1994): Classical logic: Continuation Semantics and Abstract Machines