Characterizing of Strong Normalization for $\lambda\mu$-Calculus

Xinxin Shen¹, Kougen Zheng²*
¹,² Department of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, 310027, China
*Corresponding author’s e-mail: zkg@cs.zju.edu.cn

Abstract. $\lambda\mu$-calculus is introduced by Parigot as an extension isomorphic to an alternative presentation of classical natural deduction. Since then, many properties of it have been studied and, in particular, it does not enjoy the separation property shown by David and Py. $\lambda\mu$-calculus is proposed by de Groote and developed by Saurin as an extension of $\lambda\mu$-calculus. Saurin demonstrates that the separation property holds for the $\lambda\mu$-calculus. Bakel gives a characterization of strong normalization of $\lambda\mu$-calculus in the view of intersection type. In this paper, we will extend the intersection type assignment system to the $\lambda\mu$-calculus, and show that it characterizes those terms that are strongly normalizing. The system satisfies the subject reduction and subject expansion properties.

1. Introduction

The intersection type assignment systems, introduced into the lambda calculus in the late 1970s by Coppo and Dezani [1-2] are devised to characterize the set of solvable terms and later extended by Barendregt [3] and Pottinger [4]. For an overview of the various existing systems, please refer to [5]. They extend the simple typed assignment system to include intersections and corresponding rules, allowing for term variables (and terms) to have more than one type. Intersection types have been used in a series of papers for characterizing evaluation properties of $\lambda$-terms [6-15].

Meanwhile, the ordinary $\lambda$-calculus [16] has been extended in several ways. There is a correspondence, called Curry-Howard correspondence, between simply typed $\lambda$-calculus and intuitionistic natural deduction. In order to extend the Curry-Howard correspondence to the classical natural deduction, the $\lambda\mu$-calculus [17] is proposed. Properties of $\lambda\mu$-calculus have been extensively studied both as a typed and an untyped language. Among them, separation property (also called Böhm Theorem) does not hold as shown by [18]. To recover the Böhm Theorem, Saurin [19] develops the $\lambda\mu$-calculus. The $\lambda\mu$-calculus can be seen as a stream calculus that enjoys some fundamental properties [20-21].

The full characterisation of strong normalization is a property that is shown for various intersection systems for the $\lambda$-calculus. To show that all typeability terms are strongly normalizable, reducibility method introduced by Tait [22] is suggested. The converse of this result is dependent on the subject expansion that can only reliably be shown for left-most outermost reduction [5] or perpetual reduction [23-24]. Bakel [14] characterizes the strongly normalizing $\lambda\mu$-terms and has observed that the intersection type assignment system can be adapted to $\lambda\mu$-calculus, but whether the characterization result can be extended to $\lambda\mu$ is unknown. Saurin [20] introduces a simply-typed $\lambda\mu$-calculus which satisfies the subject reduction and typed terms are strongly normalizable in the type system. However,
the reverse of this property does not hold for the system. The aim of this paper is to provide a characterization of strongly normalizing $\lambda\mu$-terms and our approach follows the perpetual reduction.

2. $\lambda\mu$-calculus

In the grammar, we use the notation $M\alpha$ [24] instead of $[\alpha]M$. The new notation makes explicit the intuition that $\alpha$ represents a potentially infinite stream of terms to which $M$ is applied.

In Parigot's original $\lambda\mu$-calculus, terms of the form $[\alpha]P$, are distinguished as named term from the ordinary terms, and bodies of $\mu$-abstractions are restricted to the named terms. On the other hand, $\Lambda\mu$-calculus considers $P\alpha$ as an ordinary term and any term can be the body of $\mu$-abstraction in the $\Lambda\mu$-calculus.

Definition 1 (Term syntax)

1) The set $\Sigma_{\lambda\mu}$ of terms are defined inductively by the following grammar

\[ M, N ::= x | \lambda x. M | MN | \mu \alpha. M | (M)\alpha. \]

where $x \in Var_\theta$, a set of term variables (denoted by $x, y, z, ...$), and $\alpha \in Var_\beta$, a set of stream variables (denoted by $\alpha, \beta, ...$), both denumerable and they are disjoint.

2) A stream $S \in \Sigma_{\lambda\mu}$ is an applicative context of the shape:

\[ S ::= []N_1 \ldots N_m\beta \]

$M, N, ...$ are terms and $M, S, P \ldots$ are streams. Streams are from [19-20] As usual, $\lambda$ and $\mu$ are considered to be binders. The form of $\mu \alpha. M$ is called a $\mu$-abstraction and the form of $M\alpha$ is called a $\mu$-application. Terms shall always be considered up to $\alpha$-equivalence.

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 term variables and free stream variables in a term $M$ are defined in the usual way.

Convention.

1. Application is left-associative.

2. Consecutive abstractions may be collapsed to a single one. eg. $\lambda x_1x_2 \cdots x_n. M$ denotes $\lambda x_1. (\lambda x_2. (\cdots (\lambda x_n. M) \cdots ))$ and similarly for $\mu$.

3. The abstractions extend as far to the right as possible.

Definition 2 (Substitution)

There are two forms of substitutions:

- term substitution: $M[\alpha : = N]$ is defined as in $\lambda$-calculus;
- structural substitution: $M[\alpha \leftarrow N]$ is defined as the replacement of any subterm $(P)\alpha$ of $M$ with $\alpha \in fn(M)$, by the subterm $(P)N\alpha$. More precisely, $M[\alpha \leftarrow N]$ is defined by:

\[
\begin{align*}
x[\alpha \leftarrow N] &= x \\
(\lambda x. M)[\alpha \leftarrow N] &= \lambda x. M[\alpha \leftarrow N] \\
(M_1M_2)[\alpha \leftarrow N] &= (M_1[\alpha \leftarrow N])(M_2[\alpha \leftarrow N]) \\
(\mu \alpha. M)[\alpha \leftarrow N] &= \mu \beta. M[\alpha \leftarrow N] \\
(M\alpha)[\alpha \leftarrow N] &= (M[\alpha \leftarrow N])N\alpha \\
(M\beta)[\alpha \leftarrow N] &= (M[\alpha \leftarrow N])\beta \text{ if } \alpha \neq \beta
\end{align*}
\]

Definition 3 (Axioms)

Axioms of the $\lambda\mu$-calculus are defined by:

\[
\begin{align*}
\beta_T & \quad (\lambda x. M)N \rightarrow M[x := N] \\
\beta_S & \quad (\mu \alpha. M)\beta \rightarrow M[\alpha := \beta] \\
\eta_T & \quad \lambda x. (M)x \rightarrow M \text{ if } x \notin fv(M) \\
\eta_S & \quad \mu \alpha. (M)\alpha \rightarrow M \text{ if } \alpha \notin fn(M) \\
(fst) & \quad \mu \alpha. M \rightarrow \lambda x. \mu \alpha. M[\alpha \leftarrow x] \text{ if } x \notin fv(M)
\end{align*}
\]

The $M[x := N]$ and $M[\alpha := \beta]$ are the usual capture-avoiding substitutions.

Remark. For $S = [N_1 \ldots N_m\beta$,

\[
\begin{align*}
MS &= MN\beta \\
M[\alpha := S] &= M[\alpha := \beta][\beta := \bar{N}] \\
M; S &= [MN_1 \ldots N_k\beta
\]
\]

Definition 4 (Reduction)

The reduction relation $M \rightarrow N$ is defined as the compatible closure of $(\beta_T), (\beta_S), (\eta_T), (\eta_S)$ and (fst).

3. Intersection types for $\lambda\mu$-calculus

Ugo [25] presents an intersection type for $\Lambda\mu$ to prove the approximation theorem, here we give a modification inspired by [14] and [25].
Definition 5 \( \mathcal{T}_F, \mathcal{T}_S \): 
\( \delta : \sigma \rightarrow \delta' \sigma \rightarrow \delta \cap \delta' \)

\( \mathcal{T}_F \) is the set of type atoms and \( \mathcal{T}_S \) the set of stream types. Comparing with [14], we allow intersection type in the end of functional type. \( \omega \) cannot be a proper type while all terms are typeable with \( \omega_F \) or \( \omega_S \) in the full system of [25].

Notation.
- We use \( \rho, \delta, \rho', \delta', \delta_1, ... \) to denote term types, \( \sigma, \sigma_1, \sigma', \tau, ... \) stream types.
- \( \cap \) has precedence over \( \rightarrow \) and \( \times \) has precedence over \( \rightarrow \). \( \times \) and \( \rightarrow \) associate to the right. For example \((\rho \times \sigma \rightarrow \tau) \cap \rho \rightarrow \sigma \rightarrow \tau \equiv (((\rho \times \sigma) \rightarrow \tau) \cap \rho) \rightarrow (\sigma \rightarrow \tau)\).

A statement is an expression of the form \( M : \delta \), where \( M \) is a term, the subject of a statement, and \( \delta \) is a term type. An environment for term variables \( \Gamma \) is a set of statements with distinct term variables as subjects. An environment for stream variables \( \Delta \) is a set of statements with distinct stream variables as subjects. Denote \( x \in \Gamma(\alpha \in \Delta) \) if \( x(\alpha) \) is a subject of a statement in \( \Gamma \) (\( \Delta \)) and \( \emptyset \) for empty environment. If we write \( \Gamma, x: \delta(\Delta, \alpha: \sigma) \), we presuppose that \( x(\alpha) \) does not occur in \( \Gamma(\Delta) \).

Definition 6 (Intersection type assignment)
The intersection type system is defined by Figure 1 where the \( \sigma \) in rules (\( \lambda \)) and (app) may be \( \omega \).

Figure 1. The intersection type system

Definition 7 (Preorder)
The relations \( \leq_T \) and \( \leq_S \) over \( \mathcal{T}_F \) and \( \mathcal{T}_S \) respectively are the least preorders such that the relations in Table 1 are satisfied.

The preorders on \( \mathcal{T}_F \) and \( \mathcal{T}_S \) are remarked as \( \leq_T \) and \( \leq_S \), respectively. The subscripts are normally omitted when there is no confusion. Types may be considered modulo \( \sim \). Then \( \leq \) becomes a partial order.

Table 1. The Preorder Relation

| Expression | Description |
|------------|-------------|
| \( \delta_1 \leq_T \delta_2 \leq_T \delta_3 \Rightarrow \delta_1 \leq_T \delta_3 \) | \( \sigma_1 \leq S \sigma_2 \leq S \sigma_3 \Rightarrow \sigma_1 \leq S \sigma_3 \) |
| \( \sigma \rightarrow \delta_1 \cap \sigma \rightarrow \delta_2 \leq_T \delta_3 \cap \delta_1 \cap \delta_2 \leq_S \delta_3 \cap \delta_1 \cap \delta_2 \) | \( \delta_1 \cap \delta_2 \leq T \delta_1 \cap \delta_2 \leq S \delta_1 \cap \delta_2 \Rightarrow \sigma_1 \cap \sigma_2 \leq S \sigma_1 \cap \sigma_2 \) |
| \( \delta_1 \times \delta_2 \leq_S \delta_1 \times \delta_2 \cap \delta_1 \times \delta_2 \leq_T \delta_1 \times \delta_2 \) | \( \delta_1 \leq T \delta_2 \), \( \sigma_1 \leq S \sigma_2 \Rightarrow \sigma_1 \leq S \sigma_2 \) |
| \( \delta \leq T \sigma \rightarrow \delta_1 \cap \sigma \rightarrow \delta_2 \leq_S \delta_1 \cap \sigma \rightarrow \delta_2 \) | \( \delta \leq T \omega \rightarrow \delta \leq T \delta \) |
| \( \delta_1 \leq S \sigma_1 \leq S \sigma_2 \leq S \sigma_2 \) | \( \delta_2 \leq T \delta_2 \times \delta_2 \leq S \delta_2 \times \delta_2 \) |
| \( \delta_1 \times \delta_2 \leq S \delta_1 \times \delta_2 \cap \delta_1 \times \delta_2 \leq_T \delta_1 \times \delta_2 \) | \( \delta_1 \times \delta_2 \leq T \delta_1 \times \delta_2 \cap \delta_1 \times \delta_2 \leq S \delta_1 \times \delta_2 \) |
| \( \delta \leq T \omega \rightarrow \delta \leq T \delta \) | \( \omega \leq_T \delta \) |
| \( \delta_1 \leq T \sigma_1 \leq S \sigma_2 \leq S \sigma_2 \) | \( \sigma_1 \leq T \sigma_2 \leq T \sigma_2 \leq S \sigma_2 \) |
| \( \delta \leq T \omega \rightarrow \delta \leq T \delta \) | \( \omega \leq_T \delta \) |

Definition 8 The preorder on environments is extended as follows.
\( \Gamma \leq \Gamma' \) iff \( \{ x : \delta \} \subseteq \Gamma' \) implies \( \exists \delta'. (x : \delta) \in \Gamma' \) and \( \delta \leq \delta' \).

\( \Delta \leq \Delta' \) is defined in the same way.

Definition 9 \( \Gamma_1 \cup \Gamma_2 = \{ x : \delta_1 | x : \delta_1 \in \Gamma_1 \cap x \notin \Gamma_2 \} \cup \{ x : \delta_2 | x \notin \Gamma_1 \cap x : \delta_2 \in \Gamma_2 \} \cup \{ x : \delta_1 \cap \delta_2 | x : \delta_1 \in \Gamma_1 \cap x : \delta_2 \in \Gamma_2 \}. \Delta_1 \wedge \Delta_2 \) is constructed in a similar way.
Proposition 1 \( \Gamma_1 \wedge \Gamma_2 \leq \Gamma_i \) and \( \Delta_1 \wedge \Delta_2 \leq \Delta_i \) for \( i \in \{1,2\} \).

4. Properties of the type system

In this section, we show some properties on the type system. In particular, the system satisfies the subject reduction property.

Lemma 1 (Weakening and Strengthening)

- (Weakening) \( \Gamma \vdash M: \delta|\Delta \) and let \( \Gamma' = \{x: \delta \in \Gamma | x \in fv(M)\}, \Delta' = \{\alpha: \sigma \in \Delta | \alpha \in fn(M)\} \), then \( \Gamma' \vdash M: \delta | \Delta' \).
- (Strengthening) \( \Gamma \vdash M: \delta | \Delta \) and let \( \Gamma' = \{x: \delta \in \Gamma | x \in fv(M)\}, \Delta' = \{\alpha: \sigma \in \Delta | \alpha \in fn(M)\} \), then \( \Gamma' \vdash M: \delta | \Delta' \).

Proof.

- Prove by induction on the structure of the derivation.
- This is a particular case of the first one.

The above lemma and Proposition 1 lead immediately the following corollary.

Corollary 1 If \( \Gamma_1 \vdash M: \delta|\Delta \), then for any \( \Gamma_2 \), \( \Delta_2 \), \( \Gamma_1 \wedge \Gamma_2 \vdash M: \delta | \Delta_2 \).

By the definition of relations, we can get the following that will used in some proofs.

Proposition 2

1) \( (\sigma_1 \rightarrow \delta_1) \cap (\sigma_2 \rightarrow \delta_2) \leq (\sigma_1 \cap \sigma_2) \rightarrow (\delta_1 \cap \delta_2) \).
2) \( (\delta_1' \times \sigma_1 \rightarrow \delta_1) \cap (\delta_2' \times \sigma_2 \rightarrow \delta_2) \leq (\delta_1' \cap \delta_2') \times (\sigma_1 \cap \sigma_2) \rightarrow (\delta_1 \cap \delta_2) \).

The next lemma will always be needed if a given type assignment is analyzed.

Lemma 2 (Generation Lemma)

1) \( \Gamma \vdash x: \delta|\Delta \) iff there exists \( \delta' \in \mathcal{T} \) such that \( \Gamma, x: \delta' \vdash M: \delta | \Delta \).
2) \( \Gamma \vdash \lambda x. M: \delta | \Delta \) iff there exists \( \delta', \sigma \) such that \( \Gamma, \sigma \vdash \lambda x. M: \delta | \Delta \).
3) \( \Gamma \vdash M \cdot N: \delta | \Delta \) iff there exists \( \delta', \sigma \) such that \( \Gamma, \sigma \vdash M: \delta | \Delta \) and \( \Gamma, \sigma \vdash N: \delta' | \Delta \).
4) \( \mu \alpha. M: \delta | \Delta \) iff there exists \( \delta', \sigma, \alpha \) such that \( \Gamma, \alpha: \sigma, \Delta \vdash M: \delta | \Delta \).
5) \( \Gamma \vdash (M)\alpha: \delta | \Delta \) iff there exists \( \delta', \sigma, \alpha \) such that \( \Gamma, \alpha: \sigma, \Delta \vdash M: \delta | \Delta \).

Proof.

- The right-to-left is proved by induction on the structure of derivations.
- The conversations follow by induction on the structure of derivations.

Lemma 3 (Substitution Lemma)

1) \( \Gamma \vdash M[x := N]: \delta | \Delta \) with \( x \in fv(M) \), iff there exists \( \delta' \in \mathcal{T} \) such that \( \Gamma \vdash N: \delta' | \Delta \) and \( \Gamma, x: \delta' \vdash M: \delta | \Delta \).
2) \( \Gamma \vdash M[\alpha \Rightarrow N]: \delta | \Delta \) with \( \alpha \in fn(M) \), iff there exists \( \delta' \in \mathcal{T} \) such that \( \Gamma, \alpha: \sigma, \Delta \vdash M: \delta | \Delta \).

Proof.

- The right-to-left is proved by induction on the structure of derivations.
- The right-to-left is proved by induction on the structure of \( M \) and using the Lemma 2. We only consider the case \( M \equiv M_1 M_2 \):

  \( M[x := N] = M_2[x := N|M_2[x := N]]. \) By (3) of Lemma 2, there exist \( n, \delta_1, \sigma_1, \delta_1' \) such that \( \Gamma \vdash M_1[x := N]: \delta_1 \times \sigma_1 \rightarrow \delta_1' | \Delta, \Gamma \vdash M_2[x := N]: \delta_1 | \Delta, \Gamma \vdash N: \delta_1 | \Delta \). Then by rules (\( \cap \)-I) and (\( \leq \)).

b) \( x \notin fv(M_1) \) and \( x \notin fv(M_2) \). By I.H., there exists \( \delta'' \) such that \( \Gamma, x: \delta'' \vdash M_1: \delta_1 \rightarrow \delta_1' | \Delta \). Then by rules (\( \cap \)-I) and (\( \leq \)).

c) \( x \notin fv(M_1) \) and \( x \notin fv(M_2) \). By I.H., there exist \( \delta_1, \delta_2 \) such that \( \Gamma \vdash N: \delta_1 | \Delta, \Gamma, x: \delta_1 \vdash M_1: \delta_1 \rightarrow \delta_1' | \Delta \). Then by rules (\( \cap \)-I) and (\( \leq \)).
Lemma 1. So if \( \Gamma, x : \delta_1 \cap \delta_2 \vdash M_1 M_2 ; \delta_2 \rightarrow \delta_i \), then the result follows by rules (\( \cap \)-I) and (\( \leq \)).

2) The right-to-left is proved by induction on the derivation of \( \Gamma \vdash \Rightarrow \delta | \alpha : \delta' \times \sigma, \Delta \). The other direction is proved by induction on the structure of \( M \) and using the Lemma 2. We only consider one case: \( M \equiv P\beta \).

\[ a = \beta. M[\alpha \leftarrow N] = P[\alpha \leftarrow N] Na. \]

That is, \( \Gamma \vdash P[\alpha \leftarrow N] Na \vdash \delta | \alpha : \sigma, \Delta \). By (5) of Lemma 2, \( \exists \delta', \sigma', (\alpha : \sigma, \Delta)(\alpha) = \sigma' \& \Gamma \vdash P[\alpha \leftarrow N] : \sigma' \rightarrow \delta' | \alpha : \sigma, \Delta \& \delta' \leq \delta \).

Then \( \sigma' = \sigma \). By (3) of Lemma 2 there exists \( \delta'' \) such that \( \Gamma \vdash P[\alpha \leftarrow N] : \delta'' \times \sigma \rightarrow \delta' \mid \alpha : \sigma, \Delta \). Then by rule (\( \cap \)-I) and Lemma 1.

\[ a \in fn(P) : \text{By induction hypothesis, there exists } \rho \text{ such that } \Gamma \vdash P : \delta'' \times \sigma \rightarrow \delta' \mid \alpha : \rho \times \sigma, \Delta \) and \( \Gamma \vdash N : \rho | \Delta \). Note that \( \alpha \notin \Delta \) so that \( \alpha \notin fn(N) \), hence \( \Gamma \vdash \Delta : \delta'' \mid \Delta \). Then we drive \( \Gamma \vdash \Delta : \rho \times \sigma, \Delta \) and \( \Gamma \vdash N : \rho | \Delta \). On the other hand \( \rho \cap \delta'' \leq \rho, \delta'' \) implies \( \delta'' \times \sigma \rightarrow \delta' \leq (\rho \cap \delta'') \times \sigma \rightarrow \delta' \) and \( (\rho \cap \delta'') \times \sigma \leq \rho \times \sigma \).

Therefore, \( \Gamma \vdash P : (\rho \cap \delta'') \times \sigma \rightarrow \delta' | \alpha : (\rho \cap \delta'') \times \sigma, \Delta \) by rule (\( \leq \)) and Lemma 1.

\[ \alpha \notin fn(P) : P[\alpha \leftarrow N] : \Gamma \vdash P : \delta'' \times \sigma \rightarrow \delta | \alpha : \sigma, \Delta \). By Lemma 1, \( \Gamma \vdash P : \delta'' \times \sigma \rightarrow \delta | \Delta \) and then \( \Gamma \vdash P : \delta'' \times \sigma \rightarrow \delta | \alpha : \delta'' \times \sigma, \Delta \). Then by rule (\( \leq \)), \( \Gamma \vdash P : \rho \times \sigma, \Delta \).

Then by convention \( a \notin fn(N) \), hence \( \Gamma \vdash \Delta : \delta'' \mid \Delta \).

2) \( a \neq \beta \). \( M[\alpha \leftarrow N] = P[\alpha \leftarrow N] \beta \) and \( \alpha \in fn(P) \). By (5) of Lemma 2, \( \exists \delta', \sigma', (\alpha : \sigma, \Delta)(\beta) = \sigma' \& \Gamma \vdash P[\alpha \leftarrow N] : \sigma' \rightarrow \delta' \mid \alpha : \sigma, \Delta \& \delta' \leq \delta \).

By induction hypothesis, \( \exists \delta'', \Gamma \vdash P : \sigma' \rightarrow \delta' \mid \alpha : \delta'' \times \sigma, \Delta \) with \( (\beta(\delta) = \sigma' \& \Gamma \vdash N : \delta'' \mid \Delta \) and \( \Gamma \vdash N : \delta'' | \alpha : \delta'' \times \sigma, \Delta \) by Lemma 1. Then we drive that \( \Gamma \vdash P \beta : \delta' \mid \alpha : \delta'' \times \sigma, \Delta \).

Lemma 4 1) \( \forall \delta \exists k \geq 1 \exists \delta_1, \ldots, \delta_k, \sigma_1 \ldots \sigma_k \sim \delta \sim (\sigma_1 \rightarrow \delta_1) \cap \cdots \cap (\sigma_k \rightarrow \delta_k) \). The \( \sigma_i \) may be \( \omega \).

2) \( \forall \sigma \exists k \geq 1 \exists \delta_1, \ldots, \delta_k \sim \sigma \sim \delta_1 \times \cdots \times \delta_k \times \omega \)

Proof. Induction on the structure of \( \delta \) and \( \sigma \).

Theorem 1 (Subject Reduction)

If \( \Gamma \vdash \Rightarrow \delta \mid \Delta \) and \( M \rightarrow N \), then \( \Gamma \vdash \Rightarrow N : \delta \mid \Delta \).

Proof. It suffices to check the rules in Definition 3 using Lemma 2 and Lemma 3. We treat (\( f s t \)) only: \( M \equiv \mu \alpha . P \).

By (4) of Lemma 2, \( \exists \eta, \delta_i \eta \sigma_i \) such that \( \Gamma \vdash P : \delta_i \mid \alpha : \sigma_i, \Delta \) and \( \cap \sigma_i (\delta_i \rightarrow \delta_i) \leq \delta \).

Then we distinguish two cases:

1) \( a \notin f n(P) : P[\alpha \leftarrow x] = P \) for any \( x \). By Lemma 4, there exist \( \rho_i, \sigma_i \) such that \( \sigma_i \sim \rho_i \times \sigma_i \).

We can get \( \Gamma, x : \rho_i \rightarrow P : \delta_i \mid \alpha : \sigma_i, \Delta \) by Lemma 1 since \( a \notin f n(P) \) and \( x \notin f n(P) \). Therefore \( \Gamma \vdash \lambda x. \mu \alpha . P[\alpha \leftarrow x] : \rho_i \times \sigma_i \rightarrow \delta_i | \Delta \) by rules (\( \mu \)) and (\( \lambda \)).

2) \( a \in f n(P) \). By Lemma 2, there exist \( \rho_i, \sigma_i \) such that \( \sigma_i \sim \rho_i \times \sigma_i \).

Let \( x : \rho \rightarrow x : \rho | \Delta \), then \( \Gamma, x : \rho_i \rightarrow P : \delta_i \mid \alpha : \sigma_i, \Delta \).

Therefore \( \Gamma, x : \rho_i \vdash P[\alpha \leftarrow x] : \delta_i \mid \alpha : \sigma_i, \Delta \) by Lemma 3. Therefore \( \Gamma, x : \rho_i \vdash P[\alpha \leftarrow x] : \delta_i \mid \alpha : \sigma_i, \Delta \) by Lemma 3. So \( \Gamma \vdash \lambda x. \mu \alpha . P[\alpha \leftarrow x] : \rho_i \times \sigma_i \rightarrow \delta_i | \Delta \) by rules (\( \mu \)) and (\( \lambda \)).

Then by the rules (\( \cap \)-I) and (\( \leq \)), we get \( \Gamma \vdash \lambda x. \mu \alpha . P[\alpha \leftarrow x] : \delta | \Delta \).

5. Strong normalization implies typeability

Normal forms are not strictly speaking since a term always has an infinite reduction sequence because of the (\( f s t \))-rule. Normal forms in this paper are indeed canonical normal forms in [19]. \( M \) is strongly normalizable if there is no infinite reduction sequence originating in it.

Definition 10 The set of normal forms \( NF \) can be defined by the following:

\[ \overline{M}_0, \ldots, \overline{M}_m \in NF \Rightarrow \lambda \overline{x}_0 \mu \alpha_0 \lambda \overline{x}_1 \mu \alpha_1 \overline{x}_n \mu \alpha_n \overline{x}_m y_{\overline{M}_0 \beta_0 \overline{M}_1 \cdots \beta_m \overline{M}_m} \in NF. \]

Lemma 5 For any term \( M \in NF \), there exists \( \Gamma, \Delta \) and a type \( \sigma \rightarrow \delta | \Delta \).
Proof. Induction on the shape of M.

If M is a variable, then the statement is immediate.

For simplicity, let

\[ M = \lambda x_{0} \alpha_{1} \lambda x_{1} \cdots \mu \alpha_{n} \lambda x_{n} \cdot y M_{0} \beta_{1} M_{1} \cdots \beta_{m} M_{m} \]  

with \( M_{0}, \ldots, M_{m} \in NF \). By induction hypothesis, there exist \( \Gamma_{i}, \sigma_{i}, \delta_{i} \) (0 ≤ \( i \leq m \)) such that \( \Gamma_{i} \vdash M : \delta_{i} \mid \Delta_{i} \) (the structure of each \( \delta_{i} \) plays no rules). Take \( \Gamma' = \Gamma_{1} \land \cdots \land \Gamma_{m} \land y \delta_{0} \times \sigma_{1} \times \delta_{1} \times \cdots \times \sigma_{m} \times \delta_{m} \times \omega \rightarrow \delta \), and \( \Delta' = \Delta_{0} \land \cdots \land \Delta_{m} \land \beta_{i} : \sigma_{i} \) for all (1 ≤ \( i \leq m \)). By successive application of (app) and (s) rules, \( \Gamma' \vdash y M_{0} \beta_{1} \cdots \beta_{m} M_{m} : \omega \rightarrow \delta \mid \Delta' \). Let \( S = \{ \alpha_{j} | \alpha_{j} = \beta_{k} \text{ for some } k \} \). If every \( \alpha_{i} \) is different from \( \beta_{i} \), the set \( S \) is empty. Then let \( \Gamma = \Gamma' \land x_{i} : \rho_{i} \) (for all 0 ≤ \( i \leq n \)), \( \Delta = \Delta' - S \), \( \tau_{j} = \tau_{j} \cap \sigma_{k} \) where \( \tau_{j} \), \( \sigma_{k} \) on the right side are the type of \( \alpha_{j} \), \( \beta_{k} \), respectively. By successive application of rules (\( \lambda \)) and (\( \mu \)) and Lemma 1, we get \( \Gamma \vdash \lambda x_{0} \alpha_{1} \cdots \mu \alpha_{n} \lambda x_{n} \cdot y M_{0} \beta_{1} M_{1} \cdots \beta_{m} M_{m} : (\rho_{0} \times \tau_{1}) \rightarrow \cdots \rightarrow (\rho_{n} \times \omega \rightarrow \delta) \mid \Delta \).

We define our perpetual strategy for the \( \Lambda \mu \)-calculus, using the method in [8]. If a term M is not strongly normalizing, the perpetual path of M is infinite. That is, the perpetual reduction terminates only when the term is strongly normalizing. If M is strongly normalizing, it has a perpetual reduction.

Definition 11 (Perpetual redex)

For any term not in normal form, we define its perpetual redex by Figure 2.

The perpetual strategy is the strategy that reduces always the perpetual redex. It is denoted by \( \sim \).

Figure 2. Perpetual redex

Lemma 6 (Subject Expansion)

Let \( M \sim N \) and \( \Gamma \vdash N : \delta \mid \Delta \), then \( \Gamma \vdash M : \delta \mid \Delta \).

Proof. By induction on the structure of M.

If M is its own perpetual redex and the rule used is \( \beta_{T} \) or \( \beta_{S} \):

1) if \( M \equiv (\lambda x.P)Q \) and \( x \in fv(P) \), we wish to prove: if \( \Gamma \vdash P[x := Q] : \delta \mid \Delta \), then \( \Gamma \vdash (\lambda x.P)Q : \delta \mid \Delta \). By Lemma 3, there exists \( \delta' \in T_{\delta} \) such that \( \Gamma \vdash Q : \delta' \mid \Delta \) and \( \Gamma, x : \delta' \vdash P : \delta \mid \Delta \). By Lemma 4, there exist \( \delta_{1}, \ldots, \delta_{k}, \sigma_{1}, \ldots, \sigma_{k} \) such that \( \delta \sim (\sigma_{1} \rightarrow \delta_{1}) \cap \cdots \cap (\sigma_{k} \rightarrow \delta_{k}) \). So \( \Gamma, x : \delta' \vdash P : \sigma_{i} \rightarrow \delta_{i} \). By rule (\( \lambda \)), \( \Gamma \vdash \lambda x.P : \delta' \times \sigma_{i} \rightarrow \delta_{i} \mid \Delta \). Then \( \Gamma \vdash (\lambda x.P)Q : \sigma_{i} \rightarrow \delta_{i} \mid \Delta \). The result follows by rule (\( \land \)-I).

2) \( M \equiv (\mu \alpha.P)\beta \). We wish to prove: if \( \Gamma \vdash P[\alpha := \beta] : \delta \mid \Delta \), then \( \Gamma \vdash (\mu \alpha.P)\beta : \delta \mid \Delta \). Suppose \( \Delta(\beta) = \sigma \). By assumption, \( \Gamma \vdash P : \delta \mid \Delta' \) where \( \Delta'(\alpha) = \Delta(\beta) = \sigma, \Delta'(\gamma) = \Delta(\gamma) \). Then \( \Gamma \vdash (\mu \alpha.M) : \sigma \rightarrow \delta \mid \Delta' \setminus \alpha \). By Lemma 1, \( \Gamma \vdash (\mu \alpha.M) : \sigma \rightarrow \delta \mid \Delta' \setminus \alpha, \beta : \sigma \). So \( \Gamma \vdash (\mu \alpha.M) \beta : \delta \mid \Delta \).

3) \( M \equiv \lambda x.Px \) and \( x \notin fv(P) \), then \( N = P \). By Lemma 4, \( \exists k \geq 1 \exists \delta_{1}, \ldots, \delta_{k}, \sigma_{1}, \ldots, \sigma_{k} \) such that \( \delta \sim (\sigma_{1} \rightarrow \delta_{1}) \cap \cdots \cap (\sigma_{k} \rightarrow \delta_{k}) \) and then \( \Gamma \vdash P : \sigma_{i} \rightarrow \delta_{i} \). On the other hand,
∃ δ₁′ ⋯ δₙ′, σ₁ ∼ δ₁′ × ⋯ × δₙ′ × ω. Therefore, Γ, x: δ₁′ ⊢ Px: δ₂′ ⋯ δₙ′ × ω → δ₁.
So we reason by Lemma 1, rules (λ), (app) to get Γ ⊢ λ x. Px: δ₁′ × δ₂′ ⋯ δₙ′ × ω → δ₁|Δ. That is Γ ⊢ λ x. Px: σ₁ → δ₁|Δ. The result follows by rule (∩-I).

4) \( M \equiv \mu \alpha. P \alpha \) and \( \alpha \not\in f_{n}(P) \), then \( N = P \). By Lemma 4, ∃ \( k \geq 1 \) ∃ δ₁, ⋯, δₙ, σ₁, ⋯, σₙ such that \( \delta \sim (σ₁ → δ₁) \cap ⋯ \cap (σₙ → δₙ) \) and then Γ ⊢ P: σ₁ → δ₁, \( \alpha \not\in f_{n}(P) \), then by Lemma 1, Γ ⊢ P: σ₁ → δ₁|Δ, α: σ₁. So the result follows by rules (s), (≤) and (∩-I).

5) \( M \equiv \mu \alpha. P \) and \( \alpha \not\in f_{v}(P) \). Here we suppose the type of N is not an intersection type. We suppose \( \lambda x. \mu \alpha. P \alpha \equiv x \) where \( x \not\in f_{v}(P) \). Hence the result follows by rules (∩-I), (≤) and the preorder. Else, it easily follows by rules and Lemma 1.

6. Typeability implies strong normalization

In this section, we use the reducibility method [22] to show that typeable terms are strongly normalizing. The general idea of reducibility method is to interpret types by suitable sets that satisfy certain realizability properties and then to develop semantics in order to obtain the soundness of the type assignment.

Let \( \mathcal{SN} \) be the set of strongly normalizable terms and \( \mathcal{SN}^{*} \) the set of streams whose elements are in \( \mathcal{SN} \). By the definition of strong normalization, we have the following.

**Proposition 3**

1) \( x \in \mathcal{SN} \Rightarrow \forall P \in \mathcal{SN}^{*} xP \in \mathcal{SN} \).

2) \( M \downarrow N \in \mathcal{SN} \land N \in \mathcal{SN} \Rightarrow (\lambda x. M) NS \in \mathcal{SN} \).

3) \( M \equiv P \alpha \) then \( N = P \alpha \) where \( P \equiv P \). Application of induction hypothesis and Lemma 2 suffices.

**Theorem 2** \( M \) is strong normalizing, then \( M \) is typeable.

**Proof.** The proof is by induction over the length of the perpetual derivation of \( M \). For the base case we observe that normal forms are typeable by Lemma 5, the induction step follows by Lemma 6.
4) \( \mathcal{M} / \rho(x = N) = \mathcal{M} / \rho(x = x)[x := N] \) if \( x \) is not free in the image of \( \rho \).

5) \( \mathcal{M} / \rho(\alpha) = \rho(\alpha) \mathcal{M} / \rho^\prime \), \( \alpha \notin \text{dom}(\rho) \).

6) \( \mathcal{M} / \rho = \mathcal{M} / \rho \) if \( \alpha \notin \text{dom}(\rho) \).

7) \( \mathcal{M} / \rho(\alpha = S) = \mathcal{M} / \rho(\alpha = S) \) if \( \alpha \) does not occur free in the image of \( \rho \).

Free (term or stream) variable is not free in the image of \( \rho \) is important. If it occurs free, only those \( x \) free in \( \mathcal{M} \) are substituted by \( N \) when computing \( \mathcal{M} / \rho(x = N) \) while all \( x \) are substituted when computing \( \mathcal{M} / \rho(\alpha = S)[x := N] \). They will not be equal.

**Definition 13 (Type Interpretation)**

The interpretation of types is defined as follows:

1) We first define \( \mathcal{I} \) of types:

\[
\mathcal{I} \omega \rightarrow \delta = \mathcal{N} \\
\mathcal{I} \delta \times \omega = \mathcal{N} \\
\mathcal{I} \delta \times \sigma = \mathcal{N} \\
\mathcal{I} \tau_1 \cap \tau_2 = \mathcal{N} \\
\mathcal{I} \alpha : \sigma = \mathcal{N}
\]

where \( \tau_1, \tau_2 \in \mathcal{T} \) or \( \tau_1, \tau_2 \in \mathcal{T} \).

2) \( \rho \models M : \delta \) iff \( \mathcal{M} / \rho \subseteq \mathcal{N} \);

\( \rho \models \Gamma, \Delta \) iff \( \rho(x) \in \mathcal{I} \delta \) and \( \rho(\alpha) \in \mathcal{I} \sigma \) for all \( x : \delta \in \Gamma, \alpha : \sigma \in \Delta \).

The preorder on types is interpreted as set inclusion.

**Lemma 7** For all \( \rho, \tau \in \mathcal{T} \), if \( \rho \leq \tau \), then \( \mathcal{M} / \rho \leq \mathcal{T} / \tau \).

**Proof.** Induction on the definition of \( \leq \).

**Lemma 8** 1) \( \mathcal{I} \delta \subseteq \mathcal{N} \) and \( \mathcal{I} \sigma \subseteq \mathcal{N}^* \) for all \( \delta, \sigma \).

2) \( xS \in \mathcal{N} \) ⇒ \( \forall \Delta xS \in \mathcal{I} \delta \).

3) \( S \in \mathcal{I} \sigma \).

**Proof.** By simultaneous induction on the structure of types. We show some of the cases.

1) We show three cases:

\( (\sigma \rightarrow \delta) \):
\[
M \in \mathcal{I} \sigma \rightarrow \delta \Rightarrow (\text{IH}(3)) \\
S \in \mathcal{I} \sigma \wedge M \in \mathcal{I} \delta \\
MS \in \mathcal{I} \delta \\
MS \in \mathcal{N} \Rightarrow \text{Definition 13} \\
M \equiv N : S \wedge N \in \mathcal{I} \delta \wedge S \in \mathcal{N}^* \Rightarrow \text{IH}(1)
\]

\( (\delta \times \omega) \):
\[
M \in \mathcal{I} \delta \times \omega \\
M \equiv N : S \wedge N \in \mathcal{I} \delta \wedge S \in \mathcal{N}^* \Rightarrow \text{IH}(1)
\]

\( (\delta \times \sigma) \):
\[
M \in \mathcal{I} \delta \times \sigma \\
M \equiv N : S \wedge N \in \mathcal{I} \delta \wedge S \in \mathcal{N}^* \Rightarrow \text{IH}(1)
\]

2) We just show the case \( \sigma \rightarrow \delta \):
\[
xS \in \mathcal{N} \Rightarrow ((1) \text{ of Proposition 3} \\
\forall P \in \mathcal{N}^* xSP \in \mathcal{N} \Rightarrow \text{IH}(1)
\]
\[ \forall P \in \mathcal{J}, xSP \in SN \Rightarrow (IH(2)) \]
\[ \forall P \in \mathcal{J}, xSP \in \mathcal{J} \Rightarrow \text{Definition 13} \]
\[ xS \in \mathcal{J} \Rightarrow \delta_j \]

3) Similar to the proof in [14].

From (2) of Lemma 8, we immediately get the following corollary.

**Corollary 2** For any variable \( x, x \in \mathcal{J} \) for all \( \delta \).

**Lemma 9**

1) If \( M[x := N]P \in \mathcal{J} \) and \( N \in \mathcal{J} \), then \( (\lambda M)NP \in \mathcal{J} \).

2) If \( M[\alpha := S]P \in \mathcal{J} \) and \( S \in \mathcal{J} \), then \( (\mu \alpha M)SP \in \mathcal{J} \).

**Proof.** By induction on the structure of types.

1) \((\varphi) \) and \((\omega \rightarrow \delta_1)\): The assumptions become \( M[x := N]P \in SN \) and \( N \in \mathcal{J} \). By Lemma 7, \( N \in SN \). Then by Proposition 3, \((\lambda \alpha M)NP \in SN \).

2) \((\varphi) \) and \((\omega \rightarrow \delta_1)\): immediate by Proposition 3.

3) \((\alpha \rightarrow \delta_1)\): \( \forall L \in \mathcal{J}, M[\alpha := S]PL \in \delta_1 \). By induction hypothesis, \( (\mu \alpha M)SPL \in \delta_1 \).

4) \((\mu \alpha : \delta_1)\): \( \forall \alpha \in \mathcal{J}, \mu \alpha M : \delta \rightarrow \delta_1 \). By induction hypothesis, \( (\mu \alpha \lambda \mu \alpha M)\alpha \in \delta_1 \).

5) \((\lambda \alpha M)\alpha : \mathcal{J} \Rightarrow \delta_1 \).

**Lemma 10 (Soundness) \( \Gamma \vdash M : \delta \Rightarrow \Gamma \vdash M : \delta \).**

**Proof.** Induction on the derivation of \( \Gamma \vdash M : \delta \).

1) \( (ax) \): \( \Gamma \vdash x : \delta \Rightarrow \Gamma \vdash x \in \mathcal{J} \). For any \( \rho \vdash \Gamma, \delta \), \( x \in \mathcal{J} \).

2) \( (\lambda) \): \( \Gamma \vdash \lambda M : \delta_1 \times \sigma \Rightarrow \delta_2 \) since \( \Gamma \vdash \sigma : \delta \). Let \( \rho \vdash \Gamma, \delta \). Take \( N \) \( \in \mathcal{J} \). Since \( x \) is bound, \( x \) does not occur free in the image of \( \rho \) and \( \rho[x := N] \in \mathcal{J} \).

3) \( (app) \): \( M \equiv PQ \) and \( \Gamma \vdash PQ : \delta \Rightarrow \delta_2 \) since \( \Gamma \vdash P : \delta_1 \times \sigma \Rightarrow \delta_2 \) and \( \Gamma \vdash Q : \delta_1 \). By induction hypothesis, \( \Gamma \vdash \Pi : \delta_1 \times \sigma \Rightarrow \delta_2 \). That is for any \( \rho \vdash \Gamma, \delta \), \( \Pi \in \mathcal{J} \).

4) \( (\mu) \): \( M \equiv \mu \alpha M' \) and \( \Gamma \vdash \mu \alpha M' : \delta \Rightarrow \delta_2 \) since \( \Gamma \vdash M' \). Since \( \mu \alpha M' \) does not occur free in the image of \( \rho \). By Barendregt convention, \( \alpha \) does not occur free in the image of \( \rho \). By Definition 13, \( \Gamma \vdash \mu \alpha M' \rightarrow \mathcal{J} \).

5) \( (s) \): \( \Gamma \vdash \mu \alpha \rightarrow \delta | \alpha : \sigma, \Delta \) since \( \Gamma \vdash P : \sigma \rightarrow \delta | \alpha : \sigma, \Delta \). By induction hypothesis, \( \Gamma \vdash P : \sigma \rightarrow \delta | \alpha : \sigma, \Delta \), \( \rho \vdash \Gamma, \delta \), \( \rho(\alpha := S) \in \mathcal{J} \). Therefore \( \mu \alpha \lambda \mu \alpha M \in \mathcal{J} \).

By the definition of type interpretation \( \Gamma \vdash \mu \alpha M' \in \mathcal{J} \).
6) \( (\leq) \): By induction hypothesis and Lemma 7.
7) \((\cap-1)\): By induction hypothesis and Definition 13.

A consequence of soundness is that every term typeable in the type system belongs to the interpretation of its type. Each type is interpreted as a suitable set satisfying the reduction property considered. Then we obtain the following theorem.

**Theorem 3** \( \Gamma \vdash M : \delta | \Delta \), then \( M \) is strongly normalizing.

Let \( \rho \) be an evaluation such that \( \rho(x) = x \) for \( x \in \Gamma \) and \( \rho(\alpha) = \alpha \) for \( \alpha \in \Delta \). By Lemma 10, \( \Gamma \vdash M : \delta | \Delta \). We observe that \( \rho(x) = x \in f\delta /' \) for all \( x : \delta' \in \Gamma \) and \( \rho(\alpha) = \alpha \in /\sigma / \) for \( \alpha : \sigma \in \Delta \) by Lemma 8 and definition of type interpretation. Hence \( \rho \models \Gamma, \Delta \) and \( fM /' \rho \in f\delta /' \), then \( M = fM /' \rho \in SN \).

**7. Conclusion**

We have shown that the strongly normalizing terms can be characterized by an intersection type assignment system. The system enjoys type preservation under reduction and satisfies the subject expansion property under perpetual reduction.

\( \lambda \mu \bar{\mu} \)-calculus is a term calculus embodying a Curry-Howard propositions-as-types correspondence for classical logic. Characterization of strong normalization in the calculus has been given by Dougherty [12]. We will investigate how to characterize properties of \( \lambda \mu \bar{\mu} \)-calculus with explicit substitution.

**References**

[1] Coppo, M., Dezani-Ciancaglini, M. (1978) A new type assignment for lambda-terms. Archiv fur mathematische Logik und Grundlagenforschung, 19:139–156.
[2] Coppo, M., Dezani-Ciancaglini, M. (1980) An extension of the basic functionality theory for the \( \lambda \)-calculus. Notre Dame J. Formal Logic, 21 (4): 685–693.
[3] Barendregt, H., Coppo, M., Dezani-Ciancaglini, M. (1983) A Filter Lambda Model and the Completeness of Type Assignment. J. Symbolic Logic 48 (4): 931–940.
[4] Pottinger, G. (1980) A type assignment for the strongly normalizable lambda-terms. To H.B.Curry: Essays on Combinatroy Logic, Lambda Calculus and Formalism: 561–578.
[5] Van Bakel, S., (2011) Strict intersection types for the lambda calculus. ACM Computing Surveys, 43 (3): 1-49.
[6] Van Bakel, S. (1992) Complete restrictions of the intersection type discipline. Theoret. Comput. Sci., 102 (1) 135–163.
[7] Dougherty, D., Lescanne, P., (2003) Reductions, intersection types, and explicit substitutions. Math. Structures Comput. Sci. 13 (1): 55–85.
[8] Lengrand, S., Lescanne, P., Dougherty, D., Dezani-Ciancaglini, M., Van Bakel, S. (2004) Intersection types for explicit substitutions. Information and Computation 189 (1): 17–42.
[9] Dezani-Ciancaglini, M., Honsell, F., Motohama, Y. (2005) Compositional characterisations of \( \lambda \)-terms using intersection types. Theoret. Comput. Sci. 340 (3): 459–495.
[10] Matthes, R. (2000) Characterizing strongly normalizing terms of \( \lambda \)-calculus with generalized applications via intersection types. In: Goos, G., Hartmanis, J. and van Leeuwen, J. (Eds) CALP. Springer, Geneva, Switzerland. pp.339-354.
[11] Koletsos, G., Stavrinos, G. (2008) Church-Rosser property and intersection types. Australasian Journal of Logic, 6: 37–54.
[12] Daugherty, D.J., Ghilezan, S., Lescanne, P. (2008) Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: Extending the coppodezani heritage. Theoret. Comput. Sci., 398: 114–128.
[13] Koletsos, G. (2012) Intersection Types and Termination Properties. Fundamenta Informaticae, 121(1-4): 185–202.
[14] Van Bakel, S., Barbanera, F., De’Liguoro, U. (2012) Characterisation of strongly normalising lambda-mu-terms. In: Proceedings ITRS 2012. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association. pp. 1–17.

[15] Santo, J.E., Ghilezan, S. (2017) Characterization of strong normalizability for a sequent lambda calculus with co-control. In Proceedings of the 19th International Symposium on Principles and Practice of Declarative Programming. ACM New York, NY, USA, Namur, Belgium. pp. 163–174.

[16] Barendregt, H. (1984) The Lambda Calculus: Its Syntax and Semantics, North-Holland

[17] Parigot, M. (1992) \(\lambda\mu\)-calculus: an algorithmic interpretation of classical natural deduction. In: Voronkov, A. (Eds) Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Lecture Notes in Computer Science. Springer, Petersburg, Russia. pp. 190–201.

[18] David, R. Py, W. \(\lambda\mu\)-calculus and b"ohm theorem. Journal of Symbolic Logic. 66(1):407-413.

[19] Saurin, A. (2005) Separation with streams in the \(\Lambda\mu\)-calculus. In: 20th Annual IEEE Symposium on Logic in Computer Science. pp. 356–365.

[20] Saurin, A. (2008) On the Relations between the Syntactic Theories of \(\lambda\mu\)-Calculi. In: Kaminski, M., Martini, S. (Eds) Computer Science Logic. CSL 2008. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. Bertinoro, Italy. pp 154-168.

[21] Saurin, A. (2010) Standardization and Bohm Trees for \(\Lambda\mu\)-Calculus. In: Kaminski, M., Martini, S. (Eds) Computer Science Logic. CSL 2008. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. Sendai, Japan. pp.134-149.

[22] Tait, W.W. (1967) Intensional interpretations of functionals of finite type i. Journal of Symbolic Logic, 32:198–212.

[23] Dougherty, D., Ghilezan, S., Lescanne, P. (2004) Characterizing strong normalization in a language with control operators. In: Proceedings of the 6th ACM SIGPLAN international conference on Principles and practice of declarative programming. Verona, Italy. pp. 155–166.

[24] Neergaard, P. (2005) Theoretical pearls- a bargain for intersection types- a simple strong normalization proof. Journal of Functional Programming, 15(5): 699-677.

[25] De’Liguoro, U. (2017) The approximation theorem for the \(\Lambda\mu\)-calculus. Math.Structures Comput. Sci, 27(5):560-580.