INTERSECTION TYPE SYSTEM AND LAMBDA CALCULUS WITH DIRECTOR STRINGS

Xinxin Shen and Kougen Zheng

Department of Computer Science and Technology,
Zhejiang University, Hangzhou, China

ABSTRACT

The operation of substitution in λ-calculus is treated as an atomic operation. It makes that substitution operation is complex to be analyzed. To overcome this drawback, explicit substitution systems are proposed. They bridge the gap between the theory of the λ-calculus and its implementation in programming languages and proof assistants. λ_o-calculus is a name-free explicit substitution. Intersection type systems for various explicit substitution calculi, not including λo-calculus, have been studied by researchers. In this paper, we put our attention to λ_o-calculus. We present an intersection type system for λ_o-calculus and show it satisfies the subject reduction property.

KEYWORDS

Intersection type, Lambda calculus, Director strings, Subject reduction

1. INTRODUCTION

In λ-calculus [1], the operation of substitution is treated as an atomic operation. But in the presence of variable binding, substitution is a complex operation to define and implement and may cause size explosion. Therefore, substitutions are delayed and explicitly recorded, in practice. Contrast to the λ-calculus, explicit substitution decomposes the higher-order substitution operation into more atomic steps. In these last years, several explicit substitution systems have been proposed [2-5]. They are divided into two kinds: *named*, such as λx_{gc}, and *unnamed*, such as λs_e, $\lambda\sigma$. Director strings were introduced by Kennaway and Sleep [6] and generalized by Sreedhar and Taghva [7] to capture strong reduction. Fernández et al. [8-9] present the open calculus λ_o which can fully simulate the β-reduction. λ_o-calculus offers an alternative to de Bruijn notation [10] for unnamed calculi. Terms are annotated by director strings which indicate how the substitutions should do. All these explicit substitution calculi provide bridges between formal calculus and their concrete implementations. They lead to a more pertinent analysis of the correctness and efficiency of compilers, theorem proves, and proof-checkers.

Intersection type was introduced in [11-12] to overcome the limitations of Curry's type assignment system and to provide a characterization for the solvable terms of the λ-calculus. It extends simple types to include intersections and adds corresponding rules to the type assignment system. It has been used to characterize strongly (weakly or head) normalizing or solvable terms in many variants of the λ-calculus [13-15] and to prove properties in λ-calculus, such as termination [16]. Moreover, approximation theorem, which is an important result in λ-calculus, also can be proved by intersection types [17].

Dhinaharan Nagamalai et al. (Eds) : ACSIT, SIPM, ICITE, ITCA - 2019
pp. 147–158, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90313

Related works. Several intersection types for explicit substitution were studied. Dougherty and Lescanne [18] studied the relationship between intersection types and reduction (left reduction and head reduction) of λx.Lengrand [13] characterized strongly normalizing terms of $\lambda x_{gc}$with intersection types.Ventura et al. [19] presented an intersection type system for $\lambda_{db}$and showed the subject reduction property. Ventura et al. [20] introduced intersection type systems for $\lambda s_e, \lambda\sigma, \lambda\nu$-calculus and proved the subject reduction property for them. The intersection type system in [20] cannot be directly adapted to λ_o-calculus because there are no number indexes for variables. We cannot find the type of the variable from the type environment considered by searching the index. So, the difficulty for the intersection type system for $\lambda_o$is to build the correspondence from variables to the type environment.

To our knowledge the intersection type system for λ_o-calculus has not been studied. In this paper, we introduce an intersection type system for λ_o-calculus and prove the subject reduction property. The rest of this paper is structured as follows. In Section 2, we provide the term syntax of λ_o-calculus. We present the intersection type system for λ_o- calculus and show the subject reduction property in Section 3. We conclude in Section 4.

2. LAMBDA CALCULUS WITH DIRECTOR STRINGS λ_o

2.1. Term Syntax

We recall some definitions and properties of the λ_o from [8], adding some notations.

Definition 1. (**λ-calculus with Director Strings** [8])

Four syntactic categories are defined:

- Directors: We use five special symbols, called directors, ranged over by α, γ, δ:

1. '↘' indicates that the substitution should be propagated only to the right branch of a binary construct (application or substitution, as given below).
2. '↙' indicates that the substitution should be propagated only to the left branch of a binary construct.
3. '⇄' indicates that the substitution should be propagated to both branches of a binary construct.
4. '↓' indicates that the substitution should traverse a unary construct (abstraction and variables, see below).
5. '−' indicates that the substitution should be discarded (when the variable concerned does not occur in a term).

- Strings: A director string is either empty, denoted by ϵ, or built from the above symbols (so is of the form $\alpha_1\alpha_2\cdots\alpha_n$ where the α_i 's are directors). We use Greek letters such as ρ, σ ... to range over strings.

- The length of a string σ is denoted by |σ|. If α is a director, then α^n denotes a string of α 's of length n. If σ is a director string of length n and $1 \leq i \leq j \leq n$, σ_i denotes the i th director of σ and $\sigma_{\setminus i} = \sigma_1 \cdots \sigma_{i-1}\sigma_{i+1} \cdots \sigma_n$ is σ where the i th director has been removed. $\sigma_{i..j} = \sigma_i \cdots \sigma_j$ is our notation for substrings. We use σ_+ to represent the σ where all - have been removed.

$|\sigma|_l$ denotes the number of ↙ and ⇄ occurring in σ , $|\sigma|_r$ the number of ↘ and ⇄, $|\sigma|_{lr}$ the number of ⇄, and $|\sigma|_+$ the number of directors that are not $-$. $|\sigma|_-$ is the number of directors that are $-$.

- Preterms: Let σ range over strings, k be a natural number and $\boldsymbol{t}, \boldsymbol{u}$ range over preterms, which are defined by the following grammar: $t::=\square^{\sigma}|\ (\lambda\ t)^{\sigma}|(tu)^{\sigma}|\ t[k/u]\hat{}\sigma$.

- Terms: Well-formed terms are preterms that recursively satisfy the conditions in Figure 1. where $\mathcal{U} = (\downarrow\ |-)^*$ and $\mathcal{B} = (\swarrow\ |\ \rightleftarrows\ |\ \searrow\ |-)\hat{}*$.

Name	Term	Constraints
Variable	$\square^{\sigma}$	$\sigma \in \mathcal{U}, \lvert\sigma\rvert_+ = 1$
Abstraction	$(\lambda t^{\rho})^{\sigma}$	$\sigma \in \mathcal{U}, \lvert\rho\rvert = \lvert\sigma\rvert_+ + 1$
Application	$(t^{\rho}u^{\nu})^{\sigma}$	$\sigma \in \mathcal{B}, \lvert\rho\rvert = \lvert\sigma\rvert_l, \lvert\nu\rvert = \lvert\sigma\rvert_r$
Substitution	$(t^{\rho}[k/u^{\nu}])^{\sigma}$	$\sigma \in \mathcal{B}, \lvert\rho\rvert = \lvert\sigma\rvert_l + 1, \lvert\nu\rvert = \lvert\sigma\rvert_r, 1 \le k \le \lvert\rho\rvert$

Figure 1.Term Condition

A variety of different term constructs [8]:

- □presents variables,
- $(\lambda t^{\rho})^{\sigma}$ is an abstraction,
- $(\boldsymbol{tu})^{\sigma}$ is an application,
- $\boldsymbol{t}[k/\boldsymbol{u}]^{\sigma}$ is an explicit substitution, meaning that the variable corresponding to the k^{th} director in $\boldsymbol{t}$'s string is to be replaced by $\boldsymbol{u}$.

Remark 1.

- Given a term t^{ρ}, $|\rho|_+$ is equal to the number of free variables in t.
- In an abstraction $(\lambda t^{\rho})^{\sigma}$, the last director in ρ corresponds to the bound variable.
- Parentheses will be dropped whenever we can, and omit the empty string ϵ unless it is essential.

We give an example:

Example 1. If we consider variables $\{y, z, w\}$, and the pure λ -term $t = \lambda x.xyw$. The term in λ_o-calculus corresponding to (use the function defined in [8]) t is $((\lambda\ (\square^{\downarrow}\square^{\downarrow})^{\searrow\swarrow}\square^{\downarrow})^{\swarrow\ \searrow\swarrow})^{\downarrow-\downarrow}$.

2.2. Reduction rules

The *Beta* rule is aimed at eliminating β-redexes and introduce an explicit substitution. It is defined by

$$\lambda \boldsymbol{t}^{\rho}\boldsymbol{u} \rightsquigarrow_o (\boldsymbol{t}[|\rho|_+ + 1/\boldsymbol{u}])^{\tau}$$

where $\tau = \psi_b(\sigma, \rho)$with φ_b defined in Figure 2.

Definition 2. (Reduction Rules [8])

Reduction rules in λ_o contain the *Beta* rule and propagation rules in Figure 2 with:

$$
\begin{array}{llll}
\pi(\epsilon, \epsilon) = \epsilon & & \pi'(\epsilon,\epsilon) = \epsilon \\
\pi(\searrow \sigma, \rho) = -\pi(\sigma,\rho) & & \pi'(\searrow \sigma, \alpha\nu) = -\pi \\
\pi(\swarrow \sigma, \alpha\rho) = \alpha\,\pi(\sigma,\rho) & & \pi'(\swarrow \sigma, \nu) = -\pi \\
\pi(\rightleftarrows \sigma, \alpha\rho) = \alpha\ \pi(\sigma,\rho) & & \pi'(\rightleftarrows \sigma, \alpha\nu) = \alpha\pi' \\
\pi(-\sigma, \rho) = -\pi(\sigma,\rho) & & \pi'(-\sigma, \nu) = -\pi
\end{array}
$$

Other functions used in the propagation rules are defined in Figure 3. The functions used in the propagation rules just compute the *ad hoc* director strings. They are generated recursively in the same way as above from the tables Figure 3 [8].

Name	*Reduction*			*Cond.*
Var	$(\Box^{\rho}[i/v^{\nu}])^{\sigma}$	$\rightsquigarrow_o$	v^{τ} where $\tau = \pi'(\sigma, \nu)$	$\rho_i = \downarrow$
App_1	$((\mathbf{t}\ \mathbf{u})^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	$((\mathbf{t}[j/\mathbf{v}])^{\upsilon}\ \mathbf{u})^{\tau}$	$\rho_i = \swarrow$
	where $\upsilon = \phi_l(\sigma, \rho_{\setminus i}), \tau = \psi_1(\sigma, \rho_{\setminus i}), j = \lvert\rho_{1..i}\rvert_l$			
App_2	$((\mathbf{t}\ \mathbf{u})^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	$(\mathbf{t}\ (\mathbf{u}[k/\mathbf{v}])^{\omega})^{\tau}$	$\rho_i = \searrow$
	where $\omega = \phi_r(\sigma, \rho_{\setminus i}), \tau = \psi_2(\sigma, \rho_{\setminus i}), k = \lvert\rho_{1..i}\rvert_r$			
App_3	$((\mathbf{t}\ \mathbf{u})^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	$((\mathbf{t}[j/\mathbf{v}])^{\upsilon}\ (\mathbf{u}[k/\mathbf{v}])^{\omega})^{\tau}$	$\rho_i = \rightleftarrows$
where $\upsilon = \phi_l(\sigma, \rho_{\setminus i}), \omega = \phi_r(\sigma, \rho_{\setminus i}), \tau = \psi_3(\sigma, \rho_{\setminus i}), j = \lvert\rho_{1..i}\rvert_l, k = \lvert\rho_{1..i}\rvert_r$				
Lam	$((\lambda\mathbf{t})^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	$\left(\lambda(\mathbf{t}[i/\mathbf{v}])^{\upsilon\cdot\swarrow}\right)^{\tau}$	$\rho_i = \downarrow$
	where $\upsilon = \phi_d(\sigma, \rho_{\setminus i}), \tau = \psi_d(\sigma, \rho_{\setminus i})$			
Comp	$((\mathbf{t}[j/\mathbf{u}])^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	$(\mathbf{t}[j/(\mathbf{u}[k/\mathbf{v}])^{\omega}])^{\tau}$	$\rho_i = \searrow$
	where $\omega = \phi_r(\sigma, \rho_{\setminus i}), \tau = \psi_2(\sigma, \rho_{\setminus i}), k = \lvert\rho_{1..i}\rvert_r$			
Erase	$(t^{\rho}[i/\mathbf{v}])^{\sigma}$	$\rightsquigarrow_o$	t^{τ} where $\tau = \pi(\sigma, \rho_{\setminus i})$	$\rho_i = -$

Figure 2. Propagation Rules $\mathcal{P}$

σ_1	ρ_1	ϕ_l	ϕ_r	ψ_1	ψ_2	ψ_3	π
$\searrow$	ϵ	$\searrow$	$\searrow$	$\swarrow$	$\searrow$	$\rightleftarrows$	$-$
$\swarrow$	$\searrow$	ϵ	$\swarrow$	$\searrow$	$\searrow$	$\searrow$	$\searrow$
$\swarrow$	$\swarrow$	$\swarrow$	ϵ	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$
$\swarrow$	$\rightleftarrows$	$\swarrow$	$\swarrow$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$
$\rightleftarrows$	$\searrow$	$\searrow$	$\rightleftarrows$	$\rightleftarrows$	$\searrow$	$\rightleftarrows$	$\searrow$
$\rightleftarrows$	$\swarrow$	$\rightleftarrows$	$\searrow$	$\swarrow$	$\rightleftarrows$	$\rightleftarrows$	$\swarrow$
$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$
$-$	ϵ	ϵ	ϵ	$-$	$-$	$-$	$-$
$\swarrow$	$-$	ϵ	ϵ	$-$	$-$	$-$	$-$
$\rightleftarrows$	$-$	$\searrow$	$\searrow$	$\swarrow$	$\searrow$	$\rightleftarrows$	$-$

σ_1	ρ_1	ψ_d	ϕ_d	ψ_b
$\searrow$	ϵ	$\downarrow$	$\searrow$	$\searrow$
$\swarrow$	$\downarrow$	$\downarrow$	$\swarrow$	$\swarrow$
$\rightleftarrows$	$\downarrow$	$\downarrow$	$\rightleftarrows$	$\rightleftarrows$
$-$	ϵ	$-$	ϵ	$-$
$\swarrow$	$-$	$-$	ϵ	$-$
$\rightleftarrows$	$-$	$\downarrow$	$\searrow$	$\searrow$

Figure 3. Functions used in the Propagation Rules

3. THE TYPE SYSTEM

3.1 Intersection Types

We take the idea from [20]. Environments are sequences of types instead of type assignments and types are non-idempotent intersection types [21].

Definition 3.

1. Intersection types are defined by:

$$\varrho, \varsigma \in \mathbb{T} ::= \mathcal{A} \mid \mathbb{U} \to \mathbb{T}$$
$$\mu, \xi \in \mathbb{U} ::= \Omega \ \mathbb{U} \cap \mathbb{U} \mid \mathbb{T}$$

where $\mathcal{A}$ is a denumerable infinite set of type variables.

2. Environments are ordered lists of types $\mu \in \mathbb{U}$, defined by $\Gamma ::= nil \mid \mu.\Gamma$. *nil* is the empty environment. We use Γ, Δ..to denote environments.

3. A term **t** is typable if there are some Γ, ϱ such that $\Gamma \vdash \boldsymbol{t}: \varrho$.

$|\Gamma|$ is the length of Γ and $|nil| = 0$. Γ_i is the *i* th type in Γ. $\Gamma_{<i}$ is the first $i-1$ types in Γ and $\Gamma_{\leq i}$, $\Gamma_{>i}$, $\Gamma_{\geq i}$ are similarly defined. If $i = 0$, then $\Gamma_{\leq 0}.\Gamma = \Gamma_{<0}.\Gamma = \Gamma$. If i is equal to the length of Γ, then $\Gamma.\Gamma_{\geq i} = \Gamma.\Gamma_{>0} = \Gamma$.

$\Gamma_{\setminus i} = \Gamma_{<i}.\ \Gamma_{>i}$, is Γ where the i th type has been removed. $\Gamma_{i..j} = \Gamma_i.\cdots.\Gamma_j$, is a sub-environment of Γ. Γ_+ is Γ where all $-$ has been removed. Ω^n denotes the environment $\Omega.\cdots.\Omega$ of length n.

3.2 Type System for λ_o

We first define some functions which will be used in defining the typing rules. It gets a new environment from two environments according to a director string.

Definition 4. The function Intersection(Γ, Δ, σ) is defined by Algorithm 1.

Algorithm 1 Intersection(Γ, Δ, σ)

```
Let n = |σ|; Γ' = Ω^n;
for i = 1; i ≤ n; i++ do
    if σ_i = ⇌ then
        Γ'_i = Γ_{|σ_{1..i}|_l} ∩ Δ_{|σ_{1..i}|_r};
    else if σ_i = ↙ then
        Γ'_i = Γ_{|σ_{1..i}|_l};
    else if σ_i = ↘ then
        Γ'_i = Δ_{|σ_{1..i}|_r};
    end if
end for
return Γ'.
```

Definition 5. The function *AddErase* (Γ, σ) is defined by Algorithm 2.

Algorithm 2 AddErase (Γ, σ)

Let $n = |\sigma|; \Gamma' = \Omega^n$;
for $i = 1; i \leq n; i++$ **do**
  if $\sigma_i \neq -$ **then** $\Gamma'_i = \Gamma_{|\sigma_{1..i}|_+}$;
  end if
end for
return Γ'.

Definition 6. The function *Drop*(Γ, σ) is defined by Algorithm 3.

Algorithm 3 Drop (Γ, σ)

Let $n = |\sigma|; \Gamma' = \Omega^n$;
for $i = 1; i \leq n; i++$ **do**
  if $\sigma_i = \not\downarrow$ or $\sigma_i = \not\rightleftarrows$ **then** $\Gamma'_i = \Gamma_{|\sigma_{1..i}|_l}$;
  end if
end for
return Γ'.

Definition 7. (Typing Rules)

Typing rules for λo are defined by Figure 4 where (*) is $\rho = \sigma\_+$ and functions are defined in Definition 4, Definition 5 and Definition 6.

$$\text{(var)}\ \frac{}{\varrho.nil \vdash \square^{\downarrow} : \varrho} \qquad \text{(abs)}\ \frac{\Gamma.\mu \vdash t^{\rho} : \varrho}{AddErase(\Gamma, \sigma) \vdash (\lambda t^{\rho})^{\sigma} : \mu \to \varrho}$$

$$\text{(-i)}\ \frac{\Gamma \vdash t^{\rho} : \varrho}{\text{AddErase}(\Gamma, \sigma) \vdash t^{\sigma} : \varrho}(*)$$

$$\text{(app)}\ \frac{\Gamma \vdash \boldsymbol{t} : \wedge\varsigma_k \to \varrho \quad \Delta^k \vdash \boldsymbol{u} : \varsigma_k \quad \forall k \in \{1, ..., n\}}{\text{Intersection}(\Gamma, \cap\Delta^k, \sigma) \vdash (\boldsymbol{tu})^{\sigma} : \varrho}$$

$$\text{(cut)}\ \frac{\Gamma_{<i}. \wedge \varsigma_k . \Gamma_{>i} \vdash \boldsymbol{t} : \varrho \quad \Delta^k \vdash \boldsymbol{u} : \varsigma_k \quad \forall k \in \{1, ..., n\}}{\text{Intersection}(\Gamma_{\setminus i}, \cap\Delta^k, \sigma) \vdash (\boldsymbol{t}[i/\boldsymbol{u}])^{\sigma} : \varrho} \quad \rho_i \neq -$$

$$\text{(drop)}\ \frac{\Gamma \vdash \boldsymbol{t} : \varrho \quad \Delta \vdash \boldsymbol{u} : \varsigma}{\text{Drop}(\Gamma_{\setminus i}, \sigma) \vdash (\boldsymbol{t}[i/\boldsymbol{u}])^{\sigma} : \varrho} \quad \rho_i = -$$

Figure 4. Typing Rules

Lemma 1. (Generation Lemma)

1. $\Gamma \vdash t\hat{}\sigma : \varrho$ and $\rho = \sigma_+$, then $\Gamma = \text{AddErase}(\Gamma', \sigma)$ and $\Gamma' \vdash t^{\rho} : \varrho$.
2. $\Gamma \vdash \square\hat{}\sigma : \varrho$, if $\sigma_i \neq -$ then $\Gamma_i = \varrho$.
3. $\Gamma \vdash (\lambda \boldsymbol{t})^{\sigma} : \varrho$, then $\varrho = \mu \to \varsigma$ for some $\mu \in \mathbb{U}$ and $\varsigma \in \mathbb{T}$, where $\Gamma_+.\mu \vdash t^{\rho} : \varrho$.
4. $\Gamma \vdash (\boldsymbol{tu})^{\sigma} : \varrho$, then $\Gamma = \text{Intersection}(\Gamma', \Delta', \sigma)$ such that $\Gamma' \vdash \boldsymbol{t} : \wedge\varsigma_k \to \varrho$, $\Delta' = \cap_1^n \Delta^k$ and $\forall\, 1 \leq k \leq n, \Delta^k \vdash \boldsymbol{u} : \varsigma_k$.

5. $\Gamma \vdash (t^{\rho}[i/\boldsymbol{u}])^{\sigma}:\varrho$ and $\rho_i \neq -$, then $\Gamma = \text{Intersection}(\Gamma'_{\setminus i}, \Delta', \sigma)$ such that $\Gamma'_{<i}.\wedge\varsigma_k.\Gamma_{>i} \vdash \boldsymbol{t}:\varrho, \Delta' = \cap_1^n \Delta^k$ and $\forall\, 1 \leq k \leq n, \Delta^k \vdash \boldsymbol{u}\}:\varsigma_k$.
6. $\Gamma \vdash (t^{\rho}[i/\boldsymbol{u}]))^{\sigma}:\varrho$ and $\rho_i = -$, then $\left(\Gamma_{\leq|\rho|}\right)_{\setminus i} \vdash t^{\rho}:\varrho$ and $\Delta \vdash \boldsymbol{u}:\varsigma$.

Proof. By induction on the structure of derivations.

Lemma 2. $\boldsymbol{\Gamma \vdash t^{\rho}:\varrho}$ **,** then $|\Gamma| = |\rho|$ and the Γ_i is the type of the variable which ρ_i corresponds to.

Proof. Induction on the structure of t^{ρ}.

1. $\square^{\rho}$, it is immediately.
2. $(\lambda t^{\rho})^{\sigma}$. By **Lemma 1** $\Gamma_+.\mu \vdash t^{\rho}:\varrho$ for some $\mu \in \mathbb{U}$ and $\varsigma \in \mathbb{T}$, where $\varrho = \mu \to \varsigma$. By induction hypothesis, $|\Gamma_+.\mu| = |\rho|$ and Γ_{+i} is the type of ρ_i. By term conditions, $|\rho| = |\sigma|_+ + 1$. From the definition of *AddErase*, we can get $|\Gamma| = |\sigma|$ and Γ_i is the type ofσ_i.
3. $(t^{\rho}u^{\nu})^{\sigma}$. By **Lemma 1**, $\Gamma = \text{Intersection}(\Gamma', \Delta', \sigma)$ such that $\Gamma' \vdash t^{\rho}: \wedge\varsigma_k \to \varrho$, $\Delta' = \cap_1^n \Delta^k$ and $\forall\, 1 \leq k \leq n, \Delta^k \vdash u^{\nu}:\varsigma_k$. By induction hypothesis, $|\Gamma'| = |\rho| = |\sigma|_l$, Γ'_i is the type of the variable which ρ_i corresponds to and $|\Delta'| = |\Delta^k| = |\nu| = |\sigma|_r$, Δ'_i is the type of the variable which ν_i corresponds to. From the definition of the function *Intersection*, we easily get $|\text{Intersection}(\Gamma', \Delta', \sigma)| = |\sigma|$ and Γ_i is the type of the variable whichσ_i corresponds to.
4. $\Gamma \vdash (t^{\rho}[i/\mathbf{u}])^{\sigma}:\varrho$ and $\rho_i \neq -$, it is similar to the last case.
5. $\Gamma \vdash (t^{\rho}[i/\mathbf{u}])^{\sigma}:\varrho$ and $\rho_i = -$, it is immediately by induction hypothesis and the reduction rule.

Theorem 1. (Subject Reduction)

Let $\Gamma \vdash \boldsymbol{t}:\varrho$. If $\boldsymbol{t} \rightsquigarrow_{\varrho} \boldsymbol{u}$, then $\Gamma \vdash \boldsymbol{u}:\varrho$.

Proof. By the verification of subject reduction for each reduction rule of the λ_o-calculus.

• *(Beta)*: Let $\Gamma \vdash ((\lambda\,\mathbf{t})^{\rho}\mathbf{u})\hat{}\sigma : \varrho$. We want to prove that $\Gamma \vdash (\mathbf{t}[|\rho|_+ + 1 / \mathbf{u}])^{\tau}:\varrho$. By **Lemma 1**, we have the following derivation:

$$\frac{\dfrac{\Gamma_2.\wedge\varsigma_k \vdash \boldsymbol{t}:\varrho}{\Gamma' = AddErase(\Gamma_2,\rho) \vdash (\lambda\boldsymbol{t})^{\rho}:\wedge\varsigma_k \to \varrho} \quad \forall 1 \leq k \leq n, \Delta^k \vdash \boldsymbol{u}:\varsigma_k}{\Gamma = \text{Intersection}(\Gamma', \cap\Delta^k, \sigma) \vdash ((\lambda\boldsymbol{t})^{\rho}\boldsymbol{u})^{\sigma}:\varrho}$$

By Lemma 2, $|\Gamma'| = |\rho|$ and Γ'_i is the type of the variable which ρ_i corresponds to. $\Gamma_1 = \text{Intersection}\}(\Gamma_2, \Delta', \tau) \vdash (\mathbf{t}[|\rho|_+ + 1 / \mathbf{u}])^{\tau}:\varrho$ by rule (cut). Suppose the variable σ_i corresponding to is x, observing the procedure and the definition of φ_b:

1. $\sigma_i = \searrow$. Then $\tau_i = \searrow$; $\Gamma_{1_i} = \Delta'_{|\tau_{1..i}|_r}$. $\Gamma_i = \Delta'_{|\sigma_{1..i}|_r}$. They are the type of $\boldsymbol{u}_x$ (variable x in $\boldsymbol{u}$). $\Gamma_{1_i} = \Gamma_i$.

2. $\sigma_i = \swarrow$. σ_i indicates the substitution is propagated to the left branch of $((\lambda \boldsymbol{t})^{\rho}\boldsymbol{u})$. If $\rho_i \neq -$, $\tau_i = \swarrow = \sigma_i$. Γ_i is the type of $\boldsymbol{t}_x$. Let $m = |\sigma_{1..i}|_l$ and $n = |\rho_{1..m}|_+$. $\Gamma_i = \Gamma_{2_n}$. τ_i indicate the substitution is propagated to the left branch $\boldsymbol{t}$. It is the type of $\boldsymbol{t}_x$. $\Gamma_{1_i} = \Gamma_{2_{|\tau_{1..i}|_l}}$. If $\rho_i = -$, then $\tau_i = -$. The variable does not occur in the two terms. Then $\Gamma_i = \Omega = \Gamma_{1_i}$.
3. $\sigma_i = \rightleftarrows$ indicates the substitution should be propagated into both branches. If $\rho_i \neq -$, $\tau_i = \rightleftarrows$.Let $m = |\sigma_{1..i}|_l$ and $n = |\sigma_{1..i}|_r$. $p = |\rho_{1..m}|_+$. $\Gamma_i = \Gamma_{2_p} \cap \left(\cap \Delta^{\mathrm{k}}\right)_{\mathrm{n}}$. It is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{u}_x$.$\tau_i = \rightleftarrows$ indicate the substitution should be propagated into both branches. Let $m' = |\tau_{1..i}|_l$ and $n' = |\tau_{1..i}|_r$. $\Gamma_{1_i} = \Gamma_{2_{m'}} \cap \left(\cap \Delta^k\right)_{n'}$. It is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{u}_x$. If $\rho_i = -$, $\tau_i = \searrow$. Then $\Gamma_{1_i} = \Delta'_{|\tau_{1..i}|_r}$, it is the type of $\boldsymbol{u}_x$. $\Gamma_i = \Delta'_{|\sigma_{1..i}|_r}$, it is also the type of $\boldsymbol{u}_x$.
4. $\sigma_i = -$, then $\tau_i = -$. So $\Gamma_{1_i} = \Gamma_i$.

Therefore, $\Gamma = \Gamma_1$.

- (Var): Let $\Gamma \vdash (\square^{\rho}[i/\boldsymbol{v}])^{\sigma}: \varrho$ and $\rho_i = \downarrow$. We want to prove that $\Gamma \vdash \boldsymbol{v}: \varrho$.By **Lemma 1**, $\Gamma = \text{Intersection}\left(\Gamma'_{\setminus i}, \Delta', \sigma\right)$, $\Gamma'_{<i} . \wedge \varsigma_k . \Gamma'_{>i} \vdash \square^{\rho} : \varrho$ and $\forall\, 1 \leq k \leq n \Delta^k \vdash \boldsymbol{v}: \varsigma_k$.By term definition, $\rho_j = -$ for all$j \neq i$. So $\Gamma'_{\setminus i} = \Omega^{|\rho|-1}$.Observing the procedure of function Intersection.

  - $\sigma_i = \rightleftarrows$ or $\sigma_i = \searrow$. $\Gamma_i = \Omega \cap \Delta'_{|\sigma_{1..i}|_r} = \Delta'_{|\sigma_{1..i}|_r}$. i.e. Γ_i is type of the $\{|\sigma_{1..i}|_r$th variable.
  - $\sigma_i = -$ or $\sigma_i = \swarrow$. $\Gamma_i = \Omega$.

By **Lemma 2**, it coincides with the definition of π'.

- (*Erase*): Let $\Gamma \vdash (t^{\rho}[i/\boldsymbol{v}])^{\sigma}: \varrho$ and $\rho_i = -$. We want to prove that $\Gamma \vdash t^{\tau} : \varrho$.By **Lemma 1**, and **Lemma 2**, it is easily to get that Drop(Γ, σ) is coincides with π.

- (*Lam*): Let $\Gamma \vdash ((\lambda \boldsymbol{t})^{\rho}[i/\boldsymbol{v}])^{\sigma}: \varrho$ and $\rho_i = \downarrow$. We want to prove that $\Gamma \vdash (\lambda(\boldsymbol{t}[i/\boldsymbol{v}])^{\nu.\swarrow})^{\tau}: \varrho$.By **Lemma 1**, we have the following derivation:

$$\frac{\dfrac{\Gamma'.\mu \vdash \boldsymbol{t} : \zeta}{\Gamma' \vdash (\lambda \boldsymbol{t})^{\rho} : \varrho} \quad \forall 1 \leq k \leq n\ \Delta^k \vdash \boldsymbol{v} : \varsigma_k \quad \Gamma'_i = \wedge \varsigma_k}{\Gamma = \text{Intersection}(\Gamma'_{\setminus i}, \cap \Delta^k, \sigma) : ((\lambda \boldsymbol{t})^{\rho}[i/\boldsymbol{v}])^{\sigma} : \varrho}$$

where $\mu \to \zeta = \varrho$. Hence

$$\frac{\dfrac{\Gamma'.\mu \vdash \boldsymbol{t} : \zeta \quad \forall 1 \leq k \leq n\ \Delta^k \vdash \boldsymbol{v} : \varsigma_k \Gamma'_i = \cap_1^n \varsigma_k}{\Gamma_1 = \text{Intersection}(\Gamma'_{\setminus i}, \Delta', \nu.\swarrow).\mu \vdash (\boldsymbol{t}[i/\boldsymbol{v}])^{\nu.\swarrow} : \mu \to \zeta}}{\Gamma_2 = \text{AddErase}(\Gamma_1, \nu.\swarrow)) \vdash (\lambda(\boldsymbol{t}[i/\boldsymbol{v}])^{\nu.\swarrow})^{\tau} : \varrho}$$

Observing the definition of ϕ_d, there are no $-$'s in ν. Suppose the j th variable is x.

1. $\sigma_j = \swarrow, \rho_j = -,\ \nu_j = \epsilon, \tau_j = -.\ \Gamma_{2_j} = \Omega = \Gamma_j.$

2. $\sigma_j = \swarrow, \rho_j = \downarrow,\ \nu_j = \swarrow,\ \tau_j = \downarrow. \sigma_j = \swarrow$ indicates the substitution is propagated to the left branch $(\lambda\, \boldsymbol{t})^{\rho}$. Let $m = \left|\sigma_{1..j}\right|_l$. Γ_j is type of the variable ρ_m corresponding to, indicated by $\boldsymbol{t}_x$ (variable x in $\boldsymbol{t}$). $\Gamma_j = \Gamma'_{\backslash i_m}. \nu_j = \swarrow$ indicates the substitution is propagated to the left branch $\boldsymbol{t}$. Γ_{2_j} is the the type of $\boldsymbol{t}_x$. Let $m' = \left|\tau_{1..j}\right|_+$ and $n' = |\nu.\swarrow|_l$. $\Gamma_{2_j} = \Gamma_{1_{m'}} = \Gamma'_{\backslash i_{n'}}$.

3. $\sigma_j = \searrow, \rho_j = \epsilon, \nu_j = \searrow, \tau_j = \downarrow$. Let m be $\left|\tau_{1..j}\right|_+$. The type of $\Gamma_{2_j} = \Gamma_{1_m} = \Delta'_{|\nu_{1..m}|_r}$. $\Gamma_j = \Delta'_{\left|\sigma_{1..j}\right|_r}. \sigma_j = \searrow$ indicates the substitution is propagated to the right branch $\boldsymbol{v}$. Let $m = \left|\sigma_{1..j}\right|_r$. Γ_j is the type of $\boldsymbol{v}_x$. $\Gamma_j = \left(\cap\, \Delta^k\right)_m$. $\nu_j = \searrow$ indicates the substitution is propagated to the right branch $\boldsymbol{v}$. Γ_{2_j} is the type of $\boldsymbol{v}_x$. Let $m' = \left|\tau_{1..j}\right|_+$ and $n' = \left|\nu.\swarrow_{1..j}\right|_l$. $\Gamma_{2_j} = \Gamma_{1_{m'}} = \Gamma'_{\backslash i_{n'}}$.

4. $\sigma_j = \rightleftarrows, \rho_j = \downarrow,\ \nu_j = \rightleftarrows, \tau_j = \downarrow. \sigma_j = \rightleftarrows$ indicates the substitution is propagated to both branches $\boldsymbol{t}$ and $\boldsymbol{v}$. Let $m = \left|\sigma_{1..j}\right|_l, n = \left|\sigma_{1..j}\right|_r$. Γ_j is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{u}_x$. $\Gamma_j = \Gamma'_{\backslash i_m} \cap \left(\cap\, \Delta^k\right)_n$. $\nu_j = \rightleftarrows, \tau_j = \downarrow$ indicates the substitution is propagated to both branches $\boldsymbol{t}$ and $\searrow$. $\Gamma_{2_j}$is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{v}_x$. Let $m' = \left|\tau_{1..j}\right|_+, n' = |\nu.\swarrow_{1..m}|_l, p' = |\nu.\swarrow_{1..m}|_r$. $\Gamma_{2_j} = \Gamma_{1_m} = \Gamma'_{\backslash i_{n'}} \cap \left(\cap\, \Delta^k\right)_{p'}$.

5. $\sigma_j = \rightleftarrows, \rho_j = -,\ \nu_j = \searrow, \tau_j = \downarrow.\ \sigma_j = \rightleftarrows$ indicates the substitution is propagated to both branches $\boldsymbol{t}$ and $\boldsymbol{v}$. Γ_j is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{v}_x$. $\rho_j = -$, the type of $\boldsymbol{t}_x$ is Ω. So Γ_j is the type of $\boldsymbol{v}_x$. Let $n = \left|\sigma_{1..j}\right|_r$, $\Gamma_j = \left(\cap\, \Delta^k\right)_n$.Let $m' = \left|\tau_{1..j}\right|_+, n' = |\nu_{1..m}|_r$. The substitution is propagated to right branch $\boldsymbol{v}$. $\Gamma_{2_j}$is the type of $\boldsymbol{v}_x$. $\Gamma_{2_j} = \Gamma_{1_m} = \left(\cap\, \Delta^k\right)_{n'}$.

6. $\sigma_j = -, \rho_j = \epsilon,\ \nu_j = \epsilon, \tau_j = -.\ \Gamma_{2_j} = \Omega = \Gamma_j$.
   So $\Gamma_2 = \Gamma$.

- App_1, App_2, App_3 and Comp are similar. We just show the case App_1. Let $\Gamma \vdash ((\boldsymbol{t}\boldsymbol{u})^{\rho}[i/\boldsymbol{v}])^{\sigma} : \varrho$ and $\rho_i = \swarrow$. We want to prove that $\Sigma \vdash ((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})^{\tau} : \varrho$. By Lemma 1, we have the following derivation.

$$\frac{\dfrac{\Gamma_1 \vdash \boldsymbol{t} : \wedge\vartheta_k \to \varrho \quad \forall 1 \le k \le n\ \Gamma^k \vdash \boldsymbol{u} : \vartheta_k}{\Gamma' = \text{Intersection}(\Gamma_1, \cap\Gamma^k, \rho) : (\boldsymbol{t}\boldsymbol{u})^{\rho} : \varrho} \quad \forall 1 \le k \le n\ \Delta^k \vdash \boldsymbol{v} : \varsigma_k \quad \Gamma_i = \wedge\varsigma_k}{\Gamma = \text{Intersection}(\Gamma'_{\backslash i}, \cap\Delta^k, \sigma) : ((\boldsymbol{t}\boldsymbol{u})^{\rho}[i/\boldsymbol{v}])^{\sigma} : \varrho}$$

Hence

$$\frac{\dfrac{\Gamma_1 \vdash \boldsymbol{t} : \wedge\vartheta_k \to \varrho \quad \forall 1 \le k \le n\ \Delta^k \vdash \boldsymbol{v} : \varsigma_k \quad \Gamma_{1_j} = \wedge\varsigma_k}{\Gamma_2 = \text{Intersection}(\Gamma_{1\backslash j}, \cap\Delta^k, \nu) \vdash (\boldsymbol{t}[j/\boldsymbol{v}])^{\nu} : \wedge\vartheta_k \to \varrho} \quad \forall 1 \le k \le n\ \Gamma^k \vdash \boldsymbol{u} : \vartheta_k}{\Gamma' = \text{Intersection}(\Gamma_2, \cap\Gamma^k, \tau) \vdash ((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})^{\tau} : \varrho}$$

For better reading, we use variable names. Suppose the mth variable is x.

1) $\sigma_m = -, \rho_m = \epsilon,\ \nu_m = \epsilon, \tau_m = -$. The m th variable does not occurs in $((\boldsymbol{tu})^{\rho}[i/\boldsymbol{v}])^{\sigma}$ and $((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})^{\tau}$. $\Gamma'_m = \Gamma_m = \Omega$.

2) $\sigma_m = \swarrow$. The m th substitution is only considered on the left branch of $((\boldsymbol{tu})^{\rho}[i/\boldsymbol{v}])^{\sigma}$. It is the $n = |\sigma_{1..m}|_l$th in ρ.

   a. $\rho_m = \searrow,\ \nu_m = \epsilon, \tau_m = \searrow$. The n th substitution is only considered on the right brach of $\boldsymbol{tu}$. It is the $p = |\rho_{1..n}|_r$ th in $\boldsymbol{u}$'s director string. The variable ρ_p corresponding to is denoted by $\boldsymbol{u}_x$ (variable x in $\boldsymbol{u}$). So $\Gamma_m = \left(\cap \Delta^k\right)_p$. It is the type of $\boldsymbol{u}_x$. The m th substitution is only considered on the right branch of $((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})$. It is the $n' = |\tau_{1..m}|_r$th in $\boldsymbol{u}$'s director string. So $\Gamma'_m = \left(\cap \Delta^k\right)_{n'}$. It is the type of $\boldsymbol{u}_x$.

   b. $\rho_m = \swarrow,\ \nu_m = \swarrow, \tau_m = \swarrow$. The nth substitution is only considered on the left brach of $\boldsymbol{tu}$. It is the $p = |\rho_{1..n}|_l$th in $\boldsymbol{t}$'s director string. The variable is denoted by $\boldsymbol{t}_x$. $\Gamma_m = \Gamma_{1_p}$ and it is the type of $\boldsymbol{t}_x$. The m th substitution is only considered on the left branch of $((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})$. It is the $n' = |\tau_{1..m}|_l$th in $(\boldsymbol{t}[j/\boldsymbol{v}])$. Then it is the $p' = |\nu_{1..n'}|_l$ in $\boldsymbol{t}$'s string. So $\Gamma'_m = \Gamma_{1_{p'}}$ and it is the type of $\boldsymbol{t}_x$.

   c. $\rho_m = \rightleftarrows,\ \nu_m = \swarrow, \tau_m = \rightleftarrows$. The n th substitution is considered both on the left and right braches of $\boldsymbol{tu}$. It is the $p = |\rho_{1..n}|_l$th in $\boldsymbol{t}$'s string and $q = |\rho_{1..n}|_r$th in $\boldsymbol{u}$'s string. $\Gamma_m = \Gamma_{1_p} \cap \left(\cap \Gamma^k\right)_q$. It is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{u}_x$. The m th substitution is considered both on the left and right branches of $((\boldsymbol{t}[j/\boldsymbol{v}])^{\nu}\boldsymbol{u})$. It is the $n' = |\tau_{1..m}|_l$th in ν. Then it is the $p' = |\nu_{1..n'}|_l$ in $\boldsymbol{t}$'s string. $q' = |\tau_{1..m}|_r$ in $\boldsymbol{u}$'s string. So $\Gamma'_m = \Gamma_{1_{p'}} \cap \left(\cap \Gamma^k\right)_{q'}$. It is intersection of the type of $\boldsymbol{t}_x$ and $\boldsymbol{u}_x$.

   d. $\rho_m = -,\ \nu_m = \epsilon, \tau_m = -$. $\Gamma_m = \Gamma'_m = \Omega$.
   So $\Gamma = \Gamma'$.

3) $\sigma_m = \rightleftarrows$ or $\sigma_m = \searrow$. These two cases are similar to $\sigma = \swarrow$. We can get $\Gamma = \Gamma'$ from the sketch of the last case.

3. CONCLUSIONS

λ_o-calculus is an unnamed explicit substitution calculus. Director strings are added to indicate how substitutions should do. It offers an alternative to de Bruijn notation. It can be used in theorem prover implementation. λ_o-calculus fully simulates the β-reduction in classical λ-calculus and it preserves the PSN property. In this paper, we propose an intersection type system for λ_o and prove the type system satisfies the subject reduction property. If a term *M* can reduce to *N*, if *M* is typed by ϱ, then *N* is also typed byϱ. In the future work, we will try to prove that a typable term in this type system is strongly normalizing and try to show a term is strongly normalizing if and only if it is typable in a certain intersection type.

REFERENCES

[1] Barendregt, H., (1984) The Lambda Calculus: Its Syntax and Semantics, Elsevier

[2] Lescanne, P. (1994) "From λσ to λν a journey through calculi of explicit substitutions", In: Proceedings of POPL, pp 60–69.

[3] Bloo, R. & Rose, K., (1995) "Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection", In: Computing Science in the Netherlands, pp 62–72.

[4] Bloo, R. & Geuvers, J., (1999) "Explicit substitution: on the edge of strong normalization", Theoretical Computer Science, Vol. 211, pp 375–395.

[5] Ayala-Rincón,M. & Kamareddine, F., (2001) "Unification via the λse-style of explicit substitution", Logic Journal of the IGPL, Vol 9, pp 489–523.

[6] Kennaway, R. & Sleep, R., (1988) "Director strings as combinators", ACM Transactions on Programming Languages and Systems, Vol. 10, No. 4, pp 602–626.

[7] Sreedhar.V. &Taghva, K.,(1993)"Capturing strong reduction in director string calculus",Theoretical Computer Science, Vol. 107, No. 2, pp 333–347.

[8] Fernández,M., Mackie,I. &Sinot,F.R.,(2005)"Lambda-calculus with director strings", Applicable Algebra in Engineering, Communication and Computing, Vol. 15, No.6, pp 393–437.

[9] Sinot, F.R.,Fernández,M. & Mackie,I.,(2003)"Efficient Reductions with Director Strings", In: Nieuwenhuis R. (eds) Rewriting Techniques and Applications. RTA 2003. Lecture Notes in Computer Science, Vol 2706. Springer, pp 46-60.

[10] De Bruijn,N., (1972)"Lambda calculus notation with nameless dummies",IndagationesMathematicaeVol 34, pp 381–392.

[11] Coppo,M. &Dezani-Ciancaglini,M.,(1978)"A new type assignment for lambda-terms",Archivfu¨r mathematischeLogik und Grundlagenforschung, Vol 19, pp 139–156.

[12] Coppo, M. &Dezani-Ciancaglini,M.,(1980)"An extension of the basic functionality theory for the λ-calculus", Notre Dame J. Formal Logic, Vol. 21, No.4, pp 685–693.

[13] Lengrand, S.,Lescanne, P.,Dougherty,D.,Dezani-Ciancaglini,M. & van Bakel.S.,(2004)"Intersection types for explicit substitutions", Information and Computation, Vol 189, No.1, pp 17 – 42.

[14] Dezani-Ciancaglini, D., Honsell, F. &Motohama, Y.,(2005)"Compositional characterisations of λ-terms using intersection types", Theoretical Computer Science, Vol. 340, No. 3, pp 459 – 495.

[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 Programmin, pp 163–174.

[16] Koletsos,G.,(2012)"Intersection Types and Termination Properties",FundamentaInformaticae, Vol. 121, No.1-4, pp 185–202.

[17] Barendregt, H., Coppo, M. &Dezani-Ciancaglini,M. (1983) A filter lambda model and the completeness of type assignment, The Journal of Symbolic Logic, Vol. 48, No.4,pp 931–940.

[18] Dougherty, D. &Lescanne, P.,(2003) Reductions, intersection types, and explicit substitutions, Mathematical Structures in Computer Science, Vol. 13, No.1, pp 55-85.

[19] Ventura, D. L., Ayala-Rincón,M., &Kamareddine,F., (2009) Intersection Type System with de Bruijn Indices, The Many Sides of Logic, Sudies in Logic Vol. 21, W., Carnielli,Coniglio, M. E. and D'Ottaviano, I. M. L., eds. pp. 557-576.

[20] Ventura,D. L.,Kamareddine,F.& Ayala-Rincón,M., (2015) Explicit substitution calculi with de bruijn indices and intersection type systems, Logic Journal of the IGPL, Vol. 23, No. 2, pp 295–340.

[21] De Carvalho,D.,(2009)"Execution time of lambda-terms via denotational semantics and intersection types", Mathematical Structures in Computer Science Conference

AUTHORS

Xinxin Shen received the BE degree in 2011 from Henan University. She is currently a Master student at College of Computer Science and Technology, Zhejiang University, China. Her area of interests are logic, lambda calculus and type theory.

Kougen Zheng received the B.E. degree in 1986 and the PhD in 1990 from Warwick University, UK. He has 23 years of teaching experience and 7 years of industry experience. He is presently working as professor in Zhejiang University, China. His areas of interest include Artificial Intelligence, Rail Traffic Signal Processing, Logic, Theory of Computation.