Explicit Substitutions Calculi with Explicit Eta rules which Preserve Subject Reduction*

Daniel Lima Ventura[1†], **Mauricio Ayala-Rincón**[1‡] **and Fairouz Kamareddine**[2]

[1]Grupo de Teoria da Computação, Departamento de Matemática,
Universidade de Brasília, Brasília D.F., Brasil
[2]School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh, Scotland

{ventura,ayala}@mat.unb.br fairouz@macs.hw.ac.uk

Abstract. *Subject reduction (for short SR) is an essential property of any type system. This property guarantees that all terms of the system preserve their types during any possible computation. It is well-known that the classic simply typed λ-calculus has this property, which means that any well-typed λ-term preserves its type under β- and η-contractions. It has been argued in the past decade that the notion of substitution in the λ-calculus needs to be made explicit. In this paper, we show that SR poses computational difficulties when the λ-calculus is extended with explicit substitutions. In particular, we show that two important calculi of explicit substitutions enlarged with Eta rules, when no explicit type and normalisation considerations are given, do not preserve subject reduction. However, we show also that if Eta reduction was made "explicit" then SR will hold for these calculi. More specifically, our results can be summarized as follows:*

- *We show that one needs to define constructively the Eta rule in both the $\lambda\sigma$-calculus and the λs_e-calculus in order to guarantee SR when Eta is applied to well-typed unrestricted terms (note that these calculi without Eta already preserve SR).*
- *We introduce constructive and explicit Eta rules for $\lambda\sigma$ and λs_e and prove that the enlarged calculi satisfy SR. The formalization of these rules involves the development of specific calculi for explicitly checking the condition of the proposed Eta rules while constructing the Eta contractum.*

1. Introduction

The development of well-behaved calculi of explicit substitutions is of great interest in order to bridge the formal study of the λ-calculus and its real implementations. Since β and η contractions depend on the definition of the operation of substitution, which is informally given in the theory of λ-calculus, most computational environments develop in an ad-hoc way an explicit notion of substitutions.

In the formal study of making substitutions explicit, several alternatives have been proposed, most of which were concerned with essential properties such as the simulation of beta-reduction, confluence, noetherianity (of the associated substitution calculus), subject reduction, principal typing, preservation of strong normalization etc. This is a non trivial task; for instance,

*Research supported by the CNPq Brazilian Research Council grants CT-INFO and Universal.
†Author supported by the Brazilian CNPq Research Council.
‡Author partially supported by the Brazilian CNPq Research Council.

the $\lambda\sigma$-calculus [1], the first proposed calculus of explicit substitutions, was reported to break the last property after some years of its introduction [11]: this implies that infinite derivations starting from well-typed λ-terms are possible in this calculus. Here, our focus is on the property of subject reduction (for short SR), which guarantees that after any contraction, all well-typed terms preserve their types.

Originally, the simply typed version of $\lambda\sigma$ was introduced (without an Eta rule) and was proved to preserve SR [1]. This calculus was enlarged with an Eta rule in [7] which was restricted to terms normalised according to the associated substitution calculus; that is to σ-normal forms. This enlarged calculus was used in [6] for treating higher-order unification (HOU) problems. In this paper, we show how to constructively define an adequate Eta rule which preserves SR for unrestricted terms. We also show how to do it for the implicit Eta rule defined in [3] for the λs_e-calculus. The proposed implementations of these Eta rules, presented respectively in [4] and [2], can be considered as informal formulations of constructive Eta rules. Of course, when restricted to well-typed normalised terms the presentations of these rules have no problem, but in [6] and [3] this was given marginally and not as an explicit part of the definitions. Here we present new constructive formalizations of explicit Eta rules for $\lambda\sigma$ and λs_e and prove that they preserve SR for unrestricted terms. These definitions involve a constructive treatment of the generation of the Eta-contractums while deciding simultaneously whether the rule applies or not.

Related works include [5], where the $\lambda\upsilon$-calculus was enlarged with an explicit noncondinional Eta rule extending directly the associated substitution calculus υ instead of extending the η-contraction, and [9], were η-expansion (extensionality) instead contraction of general explicit substitution calculi was formulated. Although η-expansion is relevant in HOU (in fact, in Huet's HOU method the η-rule is defined as the Eta expansion which makes the method more efficient), it is guided by the types of the terms which makes it possible to apply in a unique step, many rewriting steps of Huet's η-rule. Our motivation is on HOU via explicit substitutions and for this, it is necessary to use η-contractions separately.[1]

In Section 2, we enlarge $\lambda\sigma$ with a constructive and explicit Eta rule which preserves SR. In Section 3, the same is done for λs_e. Then we conclude and present future work. Omitted proofs can be found in the appendix.

2. The $\lambda\sigma$-Calculus with an explicit Eta rule which preserves SR

We assume familiarity with the simply typed λ-calculus, TA_λ (cf [8]) and de Bruijn notation. We recall the simply typed $\lambda\sigma$-calculus and show that for this calculus as enlarged with the Eta rule defined in [7, 6] when no restriction are given on terms may not satisfy SR. Then, we define an explicit Eta rule.

[1] The normalization rule for unification in $\lambda\sigma$, for instance, converts problems of the form $a =^?_{\lambda\sigma} b$ into $a' =^?_{\lambda\sigma} b'$, where a' and b' are the *long normal form* of a and b respectively. For obtaining this kind of normal forms, firstly, terms are $\beta\eta$-normalized (we use η-contraction!); secondly, these normal forms are converted to *η-long normal forms*. This corresponds to η-expansions which are normally done in a unique step according to the type of the term. An analogous situation occurs in λs_e. The practical evidence can be found in [4] where *weak η-normal forms* are defined which avoids the complete construction of η-long normal forms. In this way the set of inference rules of the HOU algorithm is modified as follows: the rule of the decomposition of applications is split in two specialized dec-application rules based on adequate application of *weak long/head normal forms*; the rule of normalisation is dropped but the $(\eta\beta)$ normalization is used as part of the rule of expansion of applications exp-app.

2.1. The $\lambda\sigma$-Calculus

The $\lambda\sigma$-calculus is a first-order rewriting system which explicits substitutions by extending the language with two sorts of objects: **terms** and **substitutions**.

Definition 1 *The syntax of the simply typed $\lambda\sigma$-calculus is given by*

Types	$A ::= K \mid A \to A$	***Terms***	$a ::= \underline{1} \mid (a\ a) \mid \lambda_A.a \mid a[s]$
Contexts	$\Gamma ::= nil \mid A.\Gamma$	***Substitutions***	$s ::= id \mid\ \uparrow\ \mid a.s \mid s \circ s$

Substitutions are lists of the form $b/\underline{i}$ indicating that the index $\underline{i}$ should be changed to the term b. The identity substitution id is of the form $\{\underline{1}/\underline{1}, \underline{2}/\underline{2}, \ldots\}$ while the lift substitution $\uparrow$ is $\{\underline{i+1}/\underline{i}\}$. The composition of substitutions is given by $\circ$. When $n \in \mathbb{N}^* = \mathbb{N} \setminus \{0\}$, $\underline{1}[\uparrow^n]$, codifies the de Bruijn index $\underline{n+1}$ and $\uparrow^0$ represents id. The value of $\underline{i}$ by the substitution s is written $\underline{i}[s]$ and can be seen as a function $s(i)$. The substitution $a.s$ has the form $\{a/\underline{1}, s(i)/\underline{i+1}\}$, and is called the **cons of** a **in** s. The β-reduction of $(\lambda_A.a\ b)$ in $\lambda\sigma$ leads to $a[b.id]$. Thus, in addition to the substitution of the free occurrences of the index $\underline{1}$ by the corresponding term, free occurrences of indices should be actualized (decreased) because of the elimination of the abstractor. Table 1 includes the rewriting system of the $\lambda\sigma$-calculus augmented with an Eta rule for the η-reduction, as presented in [6].

$(\lambda_A.a\ b)$	$\longrightarrow$	$a[b.id]$	$(Beta)$
$(a\ b)[s]$	$\longrightarrow$	$(a[s]\ b[s])$	(App)
$\underline{1}[a.s]$	$\longrightarrow$	a	$(VarCons)$
$a[id]$	$\longrightarrow$	a	(Id)
$(\lambda_A.a)[s]$	$\longrightarrow$	$\lambda_A.(a[\underline{1}.(s\circ\uparrow)])$	(Abs)
$(a[s])[t]$	$\longrightarrow$	$a[s\circ t]$	$(Clos)$
$id \circ s$	$\longrightarrow$	s	(IdL)
$\uparrow \circ (a.s)$	$\longrightarrow$	s	$(ShiftCons)$
$(s_1 \circ s_2) \circ s_3$	$\longrightarrow$	$s_1 \circ (s_2 \circ s_3)$	$(AssEnv)$
$(a.s) \circ t$	$\longrightarrow$	$a[t].(s \circ t)$	$(MapEnv)$
$s \circ id$	$\longrightarrow$	s	(IdR)
$\underline{1}.\uparrow$	$\longrightarrow$	id	$(VarShift)$
$\underline{1}[s].(\uparrow \circ s)$	$\longrightarrow$	s	$(Scons)$
$\lambda_A.(a\ \underline{1})$	$\longrightarrow$	b if $a =_\sigma b[\uparrow]$	(Eta)

Table 1. The rewriting system for the $\lambda\sigma$-calculus

The rules (Beta), (Abs) and (Eta) include the relevant information about types. Without (Eta), this system is equivalent to the one presented in [1] originally. The associated substitution calculus, denoted as σ, is the one induced by all the rules except (Beta) and (Eta), and its equality is denoted as $=_\sigma$.

The typing rules of the $\lambda\sigma$-calculus provide types for objects of sort term as well as for objects of sort substitution. An object of sort substitution is a list of terms, consequently its type corresponds to a list of types or context. Finally, $s \rhd \Gamma$ denotes that the object of sort substitution s has type Γ.

Definition 2 *$TA_{\lambda\sigma}$, the **Simple Type system for** $\lambda\sigma$ is given by the following typing rules:*

(var)	$A.\Gamma \vdash \underline{1} : A$	*(lambda)*	$\dfrac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda_A.b : A \to B}$
(app)	$\dfrac{\Gamma \vdash a : A \to B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\ b) : B}$	*(clos)*	$\dfrac{\Gamma \vdash s \rhd \Gamma' \qquad \Gamma' \vdash a : A}{\Gamma \vdash a[s] : A}$
(id)	$\Gamma \vdash id \rhd \Gamma$	*(shift)*	$A.\Gamma \vdash \uparrow \rhd \Gamma$
(cons)	$\dfrac{\Gamma \vdash a : A \qquad \Gamma \vdash s \rhd \Gamma'}{\Gamma \vdash a.s \rhd A.\Gamma'}$	*(comp)*	$\dfrac{\Gamma \vdash s'' \rhd \Gamma'' \quad \Gamma'' \vdash s' \rhd \Gamma'}{\Gamma \vdash s' \circ s'' \rhd \Gamma'}$

Example 1 *In TA_λ (and $TA_{\lambda\sigma}$), $a = (\lambda_C.(\lambda_{C\to A}.(\underline{1}\ \underline{2}))\ \underline{2})$ has type $(C \to A) \to A$ in context $B.C$. By $(Beta)$ and (Abs), a reduces to $b = \lambda_{C\to A}.((\underline{1}\ \underline{2})[\underline{1}.(\underline{2}.id) \circ \uparrow])$. Note that $\underline{2}$ abbreviates $\underline{1}[\uparrow]$. We show that $B.C \vdash b : (C \to A) \to A$.*

First, we have $\dfrac{B.C \vdash \uparrow \rhd C\, (shift) \qquad C \vdash \underline{1} : C\, (var)}{B.C \vdash \underline{1}[\uparrow] : C}(clos)$. *Analogously we have* $C \to A.C.B.C \vdash \underline{1}[\uparrow] : C$. *Furthermore,*

$$\dfrac{C \to A.B.C \vdash \uparrow \rhd B.C\, (shift) \qquad \dfrac{B.C \vdash \underline{2} : C \quad B.C \vdash id \rhd B.C\, (id)}{B.C \vdash \underline{2}.id \rhd C.B.C}(cons)}{C \to A.B.C \vdash (\underline{2}.id) \circ \uparrow \rhd C.B.C}(comp)$$

Hence $\dfrac{C \to A.B.C \vdash \underline{1} : C \to A\, (var) \quad C \to A.B.C \vdash (\underline{2}.id) \circ \uparrow \rhd C.B.C}{C \to A.B.C \vdash \underline{1}.(\underline{2}.id) \circ \uparrow \rhd C \to A.C.B.C}(cons)$

Also, $\dfrac{C \to A.C.B.C \vdash \underline{1} : C \to A\, (var) \quad C \to A.C.B.C \vdash \underline{2} : C}{C \to A.C.B.C \vdash (\underline{1}\ \underline{2}) : A}(app)$

Consequently $\dfrac{\dfrac{C \to A.B.C \vdash \underline{1}.(\underline{2}.id) \circ \uparrow \rhd C \to A.C.B.C \qquad C \to A.C.B.C \vdash (\underline{1}\ \underline{2}) : A}{C \to A.B.C \vdash (\underline{1}\ \underline{2})[\underline{1}.(\underline{2}.id) \circ \uparrow] : A}(clos)}{B.C \vdash \lambda_{C\to A}.((\underline{1}\ \underline{2})[\underline{1}.(\underline{2}.id) \circ \uparrow]) : (C \to A) \to A}(lambda)$

It is known that SR holds for the simply typed $\lambda\sigma$ without the rule (Eta).

Theorem 1 (SR for $\lambda\sigma$ without the Eta rule [1]) *Let a be a If $\Gamma \vdash_{TA_{\lambda\sigma}} a : A$ and $a \to_{\lambda\sigma} a'$, then $\Gamma \vdash_{TA_{\lambda\sigma}} a' : A$. Analogously, if $\Gamma \vdash_{TA_{\lambda\sigma}} s \rhd \Gamma'$ and $s \to_{\lambda\sigma} s'$, then $\Gamma \vdash_{TA_{\lambda\sigma}} s' \rhd \Gamma'$.*

2.2. An explicit Eta rule for $\lambda\sigma$ which preserves SR

The rule Eta introduced in [7] explicitly states that both a and b must be σ-normal forms: "The ground $\lambda\sigma$-terms in σ-normal form are exactly the λ-terms so we may apply the η-reduction of λ-calculus to those terms". And when b is restricted as well to be in σ-normal form in principle no problem appears, but without these restrictions subject reduction may be violated. In fact, assuming the definition of Eta in Table 1 and without any restriction on terms, observe that $\lambda(\underline{2}\ \underline{1}) \longrightarrow_\eta \underline{2}[\lambda(\underline{1}\ \underline{1}).\underline{1}.\uparrow]$, since $\underline{2} =_\sigma \big(\underline{2}[\lambda(\underline{1}\ \underline{1}).\underline{1}.\uparrow]\big)[\uparrow]$, where $\underline{2}$ abbreviates $\underline{1}[\uparrow]$, because $(\underline{1}[\uparrow])[\lambda(\underline{1}\ \underline{1}).\underline{1}.\uparrow] \longrightarrow_{Clos} \underline{1}[\uparrow \circ (\lambda(\underline{1}\ \underline{1}).\underline{1}.\uparrow)] \longrightarrow_{ShiftCons} \underline{1}[\underline{1}.\uparrow] \longrightarrow_{VarCons} \underline{1}$. But the term $\lambda(\underline{1}\ \underline{1})$ is not typable in the simply typed λ-calculus. Thus, without restrictions on terms this rule violates the SR property. Also, notice that this rule gives non terminality directly. In addition, notice that adding conditions to Eta such as b is well-typed is not enough for guaranteeing that computations preserve SR. In fact, the reduction $\lambda(\underline{2}\ \underline{1}) \longrightarrow_\eta \underline{1}$ can be decided by infering that $\underline{1}[\uparrow] =_\sigma \underline{2}$ by a σ-expansion that goes through ill-typed terms: $\underline{1}[\uparrow] =_\sigma \underline{2}[\lambda(\underline{1}\ \underline{1}).\uparrow] =_\sigma \underline{2}$. Consequently, in order to have a constructive and implementable definition of Eta one needs to explicitly define how the condition of the rule should be decided. The definition of Eta for the $\lambda\sigma$-calculus in Table 1 is inherited from the usual (non constructive) definition of η-reduction given in the literature for the λ-calculus *à la de Bruijn*: $\lambda(a\,\underline{1}) \to_\eta b$ if $b^+ = a$, where b^+ denotes the *lifting* of b, operator which increases by one the free indices in b. By the given definition, using the σ-equality in the condition, one has infinite possibilities of reduction, since $\underline{1}\,[b.s][\uparrow] =_\sigma a$, where $b[\uparrow] =_\sigma a$ and s is any $\lambda\sigma$-substitution.

A constructive definition of the rule Eta should depend on the original term a where in the equality $a =_\sigma b[\uparrow]$, b should be obtained from a. The implementation of the rule Eta for the $\lambda\sigma$-calculus given in [4] may be considered as an informal effort in this direction. Similarly, this happens for the definition and implementation of the Eta rule for the λs_e given in [3] and [2], respectively.

In order to give a constructive Eta rule, we use a counterpart of $\uparrow$ (see [12]).

Definition 3 *Let a be a λ-term. The **i-dash** of a, denoted as a^{-i}, is given by*

1. $(a_1\, a_2)^{-i} = (a_1^{-i}\, a_2^{-i})$
2. $(\lambda a_1)^{-i} = \lambda a_1^{-(i+1)}$
3. $$\underline{n}^{-i} = \begin{cases} \underline{n-1}\,, & \textit{if } n > i \\ \textit{undefined}, & \textit{if } n = i \\ \underline{n}\,, & \textit{if } n < i. \end{cases}$$

The **dash** of a term a is its 1-dash, denoted as a^-. If a^- is well-defined, then there is a b such that $b^+ = a$. This happens when a has no free occurrences of $\underline{1}$. This gives rise to an adequate definition of η-reduction:

Definition 4 *The **Eta rule** for λ-terms* à la de Bruijn *is given by*

$$\lambda(a\ \underline{1}\,) \to_\eta a^-, \quad \textit{whenever } a^- \textit{ is well defined} \quad \textit{(Eta)}$$

For a constructive definition of η-reduction in $\lambda\sigma$ some relevant aspects have to be considered. Given a potential η-redex $\lambda(a\ \underline{1}\,)$, the central point is how to provide a definition of *free occurrences of the index* $\underline{1}$ *in* a. If (Eta) is restricted to σ-normalized terms (i.e., free of substitutions), then one can use a notion like "$\underline{i}$ occurs free in a, if a has occurrences of $\underline{i+n}$ in the depth n of a, where a subterm a_1 of a is said to be in the depth r of a if the smaller free index in a_1 is greater than r; i.e., if a_1 is between the scope of r abstractors".

$(a\ b)[s]$	$\longrightarrow$	$(a[s]\ b[s])$ if $\diamond$ occurs in s	(η-App)
$\underline{1}\,[a.s]$	$\longrightarrow$	a if $\diamond$ occurs in s	(η-VarCons)
$(\lambda_A.a)[s]$	$\longrightarrow$	$\lambda_A.(a[\underline{1}\,.(s\circ\uparrow)])$ if $\diamond$ occurs in s	(η-Abs)
$(a[s])[t]$	$\longrightarrow$	$a[s\circ t]$ if $\diamond$ occurs in t	(η-Clos)
$(s_1 \circ s_2) \circ t$	$\longrightarrow$	$s_1 \circ (s_2 \circ t)$ if $\diamond$ occurs in t	(η-AssEnv)
$(a.s) \circ t$	$\longrightarrow$	$a[t].(s\circ t)$ if $\diamond$ occurs in $(a.s)\circ t$	(η-MapEnv)
$\uparrow \circ (a.s)$	$\longrightarrow$	s if $\diamond$ occurs in $a.s$	(η-ShiftCons)
$\diamond[\uparrow]$	$\longrightarrow$	$\diamond$	(η-Cons)
$id \circ s$	$\longrightarrow$	s if $\diamond$ occurs in s	(η-IdL)
$\underline{1}\,[\diamond.s]$	$\longrightarrow$	$\diamond$	(Error)

Table 2. $R_{\eta_{\lambda\sigma}}$: the rewriting system for η-reduction in $\lambda\sigma$

Here, the syntax of the $\lambda\sigma$-calculus is enlarged with a dummy symbol $\diamond$. $\diamond$ **occurs in a substitution** s if $s = \diamond.t$ or $s = a.t$ and $\diamond$ occurs in t. Although $\diamond$ belongs to the sort of terms, it will always occur between a substitution, according to the definition of (Eta) (to be given) and any possible application of the rules above. $N_{R_{\eta_{\lambda\sigma}}}(a)$ denotes the normal form of a in relation to $R_{\eta_{\lambda\sigma}}$, whose termination and confluence are easy to check, because respectively normalisation with $R_{\eta_{\lambda\sigma}}$ simply propagates the symbol $\diamond$ between the finite structure of the terms and because joinability of all its critical pairs. The rule (Error) points out free occurrences of the index $\underline{1}$. Conditions of the rules (η-IdL) and (η-Clos) are for avoiding simplifications which do not belong to the η-reduction in course.

The following definition of (Eta) is proposed.

Definition 5 *Let $\lambda_A.(a\ \underline{1})$ be a $\lambda\sigma$-term. The rule **(Eta)** is given by*

$$\lambda_A.(a\ \underline{1}) \to N_{R_{\eta_{\lambda\sigma}}}(a[\diamond.id]), \qquad \text{if } N_{R_{\eta_{\lambda\sigma}}}(a[\diamond.id]) \text{ is a } \lambda\sigma\text{-term.} \qquad \text{(Eta)}$$

Having the convergence of $R_{\eta_{\lambda\sigma}}$ their rewriting rules can be modified in order to quickly verify whether the $R_{\eta_{\lambda\sigma}}$-normal form of terms either belong or not to Λ_σ. This is done enlarging the language with an $error$ symbol and changing three rules by:

$$\begin{array}{lll}
(a[s])[t] & \longrightarrow & \left\{\begin{array}{ll} a[s \circ t] & \text{if } \diamond \text{ occurs in } t \text{ and } a = \underline{1} \\ a[N_{R_{\eta_{\lambda\sigma}}}(s \circ t)] & \text{if } \diamond \text{ occurs in } t \text{ and} \\ & N_{R_{\eta_{\lambda\sigma}}}(s \circ t) \in \Lambda_\sigma \\ error & \text{if } \diamond \text{ occurs in } t \text{ and} \\ & N_{R_{\eta_{\lambda\sigma}}}(s \circ t) \notin \Lambda_\sigma \end{array}\right. \qquad (\eta\text{-Clos}) \\
id \circ s & \longrightarrow & error \quad \text{if } \diamond \text{ occurs in } s \qquad (\eta\text{-IdL}) \\
\underline{1}[\diamond.s] & \longrightarrow & error \qquad (\text{Error})
\end{array}$$

This modified system will be called $R_{\eta_{\lambda\sigma}}$ too. $N_{R_{\eta_{\lambda\sigma}}}(a[\diamond.id])$ is a $\lambda\sigma$-term whenever the reduction finishes without error. With this definition, $a =_\sigma b[\uparrow]$ where $b = N_{R_{\eta_{\lambda\sigma}}}(a[\diamond.id])$. Here, "$b$" depends on a, which avoids infinite reductions in this new system.

Normalized $\lambda\sigma$-terms with respect to the σ-calculus are terms whose subterms of the form $a[s]$ are of the form $\underline{1}[\uparrow^n]$. All other subterms are of the form $\underline{1}$, $(a\ b)$ or $\lambda_A.a$. I.e., normalized terms are terms without pending substitutions. The $\lambda\sigma$-substitution $(\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1}))$ will be denoted by s_i, where $i > 1$, and s_1 denotes $\diamond.id$.

Lemma 1 *Let $i \in \mathbb{N}^*$ and a be a $\lambda\sigma$-term in normal form with respect to the σ-calculus. Then $N_{R_{\eta_{\lambda\sigma}}}(a[s_i])$ is a $\lambda\sigma$-term, whenever a has no free occurrences of $\underline{i}$. In this case, one has that $N_{R_{\eta_{\lambda\sigma}}}(a[s_i])$ corresponds to a, with the free indices greater than i decremented by one. Otherwise, $N_{R_{\eta_{\lambda\sigma}}}(a[s_i]) = error$.*

Proof. The proof is by induction on the structure of a.

1) $a = \underline{1}$: if $i = 1$, then by (Error) one has $\underline{1}[\diamond.id] \to error$. If $i > 1$, then by (η-VarCons) one obtains $\underline{1}[\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1})] \to \underline{1}$.
2) $a = \underline{1}[\uparrow^n]$: by ($\eta$-Clos), $\underline{1}[\uparrow^n][\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1})] \to \underline{1}[\uparrow^n \circ (\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1}))]$.
   - If $n < i-1$, then by (η-ShiftCons) one obtains $\underline{1}[\uparrow^n \circ (\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1}))] \to^n \underline{1}[\underline{n+1}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1})]$. So, by ($\eta$-VarCons), $\underline{1}[\underline{n+1}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1})] \to \underline{n+1} = \underline{1}[\uparrow^n]$.
   - If $n = i-1$, then if $i = 1$, by (η-IdL),$\underline{1}[id \circ s_1] \to error$.
     If $i > 1$, then by (η-ShiftCons), $\underline{1}[\uparrow^n \circ (\underline{1}.\underline{2}.\cdots.\underline{i-1}.\diamond.(id \circ \uparrow^{i-1}))] \to^n \underline{1}[\diamond.(id \circ \uparrow^{i-1})]$.
     Then, by (Error), one obtains $\underline{1}[\diamond.(id \circ \uparrow^{i-1})] \to error$.

– If $n \geq i$, then if $i = 1$, by (η-ShiftCons), $\underline{1}\,[\uparrow^n \circ(\diamond.id)] \rightarrow \underline{1}\,[\uparrow^{n-1} \circ id]$.
If $i > 1$, by (η-ShiftCons), $\underline{1}\,[\uparrow^n \circ (\underline{1}\,.\underline{2}\,.\cdots.\underline{i-1}\,.\diamond.(id \circ \uparrow^{i-1}))] \rightarrow^i \underline{1}\,[\uparrow^{n-i} \circ(id \circ \uparrow^{i-1})]$.

3) $a = (b\ c)$: by (η-App) one has $(b\ c)[s_i] \rightarrow (b[s_i]\ c[s_i])$. By induction hypothesis (IH), if either b or c have free occurrences of $\underline{i}$, then either $N_{R_{\eta_{\lambda\sigma}}}(b[s_i]) = error$ or $N_{R_{\eta_{\lambda\sigma}}}(c[s_i]) = error$. Otherwise $N_{R_{\eta_{\lambda\sigma}}}(b[s_i])$ and $N_{R_{\eta_{\lambda\sigma}}}(c[s_i])$ are $\lambda\sigma$-terms of the desired form.

4) $a = \lambda_A.b$: by (η-Abs), $(\lambda_A.b)[s_i] \rightarrow \lambda_A.(b[\underline{1}\,.(s_i \circ \uparrow)])$. By ($\eta$-MapEnv), $\lambda_A.(b[\underline{1}\,.(s_i \circ \uparrow)]) \rightarrow^{i+1}$
$\lambda_A.(b[\underline{1}\,.\cdots.\underline{i}\,.\diamond[\uparrow].(id \circ \uparrow^i)])$.
By (η-Cons), $\lambda_A.(b[\underline{1}\,.\cdots.\underline{i}\,.\diamond[\uparrow].(id \circ \uparrow^i)]) \rightarrow \lambda_A.(b[\underline{1}\,.\cdots.\underline{i}\,.\diamond.(id \circ \uparrow^i)])$.
By IH $N_{R_{\eta_{\lambda\sigma}}}(b[\underline{1}\,.\cdots.\underline{i}\,.\diamond.(id \circ \uparrow^i)]) = error$, if there are free occurrences of $\underline{i+1}$, else,
$N_{R_{\eta_{\lambda\sigma}}}(b[\underline{1}\,.\underline{2}\,.\cdots.\underline{i}\,.\diamond.(id \circ \uparrow^i)])$ is the desired $\lambda\sigma$-term. $\square$

Taking $i = 1$ in the previous lemma, one has that for σ-normalized $\lambda\sigma$-terms, (Eta) implements the η-reduction correctly. Let $t_i = \underline{1}\,.\underline{2}\,.\cdots.\underline{i-1}\,.\diamond.\uparrow^i$ and a be a σ-normalized term, where $i \in \mathbb{N}^*$. Observe that $N_{R_{\eta_{\lambda\sigma}}}(a[t_i])$ signalizes with an *error* a free occurrence of $\underline{i}$. For any substitution s, making $N_{R_{\eta_{\lambda\sigma}}}(s \circ t_i)$, one can give a definition of free occurrence of $\underline{i}$ in a substitution.

Definition 6 *Let $t_i = \underline{1}\,.\underline{2}\,.\cdots.\underline{i-1}\,.\diamond.\uparrow^i$. If $N_{R_{\eta_{\lambda\sigma}}}(a[t_i]) = error$, we say a has **free occurrences** of $\underline{i}$.*

Notice that the rules of the system $R_{\eta_{\lambda\sigma}}$ are a subset of restricted rules of the σ-calculus with conditions and augmenting the rule (η-Cons). Consequently, in order to have well-typed intermediate terms, it is enough to verify how the symbol $\diamond$ and the substitutions containing this symbol can be typed.

Definition 7 *Let Γ be a context. If $\Gamma \vdash \underline{i} : A$, then $\diamond$ **represents** $\underline{i}$ **in** Γ, denoted as $\Gamma_{<i}.\Gamma_{>i} \vdash \diamond : A$.*

From this definition, one has a rule for inferring the type of $\diamond$, that depends on the given context and has semantics related to this context only. As a consequence, the rule (η-Cons) allows $\diamond$ to own the same type in other contexts. It is necessary to check the correctness of Definition 7 in relation to its semantics.

Lemma 2 *If $\diamond$ represents $\underline{i}$ in context Γ, then $\diamond[\uparrow]$ represents $\underline{i+1}$ in $A.\Gamma$.*

Proof. Let$\Gamma \vdash \underline{i} : B$. Then $\Gamma_{<i}.\Gamma_{>i} \vdash \diamond : B$. By (shift), $A.\Gamma_{<i}.\Gamma_{>i} \vdash \uparrow \rhd \Gamma_{<i}.\Gamma_{>i}$. By (clos), $A.\Gamma_{<i}.\Gamma_{>i} \vdash \diamond[\uparrow] : B$. By (shift) and (clos), we get $A.\Gamma \vdash \underline{i}\,[\uparrow] : B$. $\square$

Lemma 3 *Let $\Gamma \vdash \underline{i} : A$ where $\diamond$ represents $\underline{i}$ in Γ. Then $\Gamma_{<i}.\Gamma_{>i} \vdash s_i \rhd \Gamma$.*

Proof. By (shift), $\Gamma_{<i}.\Gamma_{>i} \vdash \uparrow^{i-1} \rhd \Gamma_{>i}$. By (id) and (comp), $\Gamma_{<i}.\Gamma_{>i} \vdash id \circ \uparrow^{i-1} \rhd \Gamma_{>i}$. By (cons), $\Gamma_{<i}.\Gamma_{>i} \vdash \diamond.(id \circ \uparrow^{i-1}) \rhd \Gamma_{\geq i}$ holds. By (shift) and (clos), $\Gamma_{<i}.\Gamma_{>i} \vdash \underline{j} : A_j$, for all $j < i$, where $\Gamma_{<i} = A_1.\cdots.A_{i-1}$. Hence, by (cons) applied $i-1$ times, we get $\Gamma_{<i}.\Gamma_{>i} \vdash s_i \rhd \Gamma$. $\square$

Lemma 4 *Let a be a $\lambda\sigma$-term, such that $\Gamma \vdash a : A$, $\Gamma \vdash \underline{i} : B$ and $\diamond$ represents $\underline{i}$ in Γ. If a has no free occurrences of $\underline{i}$, then $\Gamma_{<i}.\Gamma_{>i} \vdash N_{R_{\eta_{\lambda\sigma}}}(a[s_i]) : A$.*

Proof. By Lemma 3, one has that $\Gamma_{<i}.\Gamma_{>i} \vdash s_i \rhd \Gamma$. By (clos) one obtains $\Gamma_{<i}.\Gamma_{>i} \vdash a[s_i] : A$. In order to guarantee that $\Gamma_{<i}.\Gamma_{>i} \vdash N_{R_{\eta_{\lambda\sigma}}}(a[s_i]) : A$, it is enough to verify that each rule of $R_{\eta_{\lambda\sigma}}$ preserves the context and type of the derived terms. Since a substitution with occurrence of $\diamond$ is well-typed, SR for the rules of the σ-calculus enlarged with the condition is preserved. It is necessary to verify that the property holds for the rule (η-Cons). The rule will always be applied after one application of the rule (η-Abs).

Suppose that $\Gamma_{<i}.\Gamma_{>i} \vdash (\lambda_C.b)[s_i] : A$, where $\Gamma \vdash \lambda_C.b : A$ and $\diamond$ represents $\underline{i}$ in Γ. One has that $\Gamma_{<i}.\Gamma_{>i} \vdash \lambda_C.(b[\underline{1}\,.(s_i \circ \uparrow)]) : A$, by SR for ($\eta$-Abs). By (lambda) one has $C.\Gamma_{<i}.\Gamma_{>i} \vdash b[\underline{1}\,.(s_i \circ \uparrow)] : D$ and $C.\Gamma \vdash b : D$, where $A = C \to D$. After i applications of (η-MapEnv) one obtains $C.\Gamma_{<i}.\Gamma_{>i} \vdash b[\underline{1}\,.\underline{2}\,.\cdots.\underline{i}.\,\diamond[\uparrow].\,(id \circ \uparrow^i)] : D$. By Lemma 2, $\diamond[\uparrow]$ represents $\underline{i+1}$ in $C.\Gamma$. Thus, when (η-Cons) is applied, $b[\underline{1}\,.\underline{2}\,.\cdots.\underline{i}.\,\diamond[\uparrow].\,(id \circ \uparrow^i)] \to b[s_{i+1}]$, with $\diamond$ representing $\underline{i+1}$ in $C.\Gamma$. By Lemma 3, $C.\Gamma_{<i}.\Gamma_{>i} \vdash s_{i+1} \rhd C.\Gamma$. By (clos), $C.\Gamma_{<i}.\Gamma_{>i} \vdash b[s_{i+1}] : D$. Hence, by (lambda), we get$\Gamma_{<i}.\Gamma_{>i} \vdash \lambda_C.(b[s_{i+1}]) : C \to D = A$. $\square$

Theorem 2 (SR for the rule (Eta)) *If* $\Gamma \vdash \lambda_B.(a\ \underline{1}\,) : A$ *and* $\lambda_B.(a\ \underline{1}\,) \to_{Eta} b$*, then* $\Gamma \vdash b : A$*.*

Proof. Suppose that $\Gamma \vdash \lambda_B.(a\ \underline{1}\,) : A$. By (lambda), $B.\Gamma \vdash (a\ \underline{1}\,) : C$, where $A = B \to C$. By (app), $B.\Gamma \vdash a : D \to C$ and $B.\Gamma \vdash \underline{1} : D$. By (var), $D = B$, thus $B.\Gamma \vdash a : B \to C = A$. One has $b = N_{R_{\eta_{\lambda\sigma}}}(a[\diamond.id])$, where a has no free occurrence of $\underline{1}$. Since $s_1 = \diamond.id$, by Lemma 4 with $i = 1$, $\Gamma \vdash b : A$. $\square$

3. The λs_e-Calculus with an explicit Eta rule which preserves SR

3.1. The λs_e-Calculus

In contrast to the $\lambda\sigma$-calculus, the λs_e-calculus has a sole sort of objects and introduces two operators σ and φ, for substitution and updating.

Definition 8 *The syntax of the simply typed* λs_e*-calculus, where* $n, i, j \in \mathbb{N}^*$ *and* $k \in \mathbb{N}$ *is given by*

$$\textbf{Types} \quad A ::= K \mid A \to A \qquad \textbf{Terms} \quad a ::= \underline{n} \mid (a\ a) \mid \lambda_A.a \mid a\,\sigma^i a \mid \varphi^j_k\, a$$
$$\textbf{Contexts} \quad \Gamma ::= nil \mid A.\Gamma$$

$a\,\sigma^i b$ represents the term $\{\underline{i}\,/b\}a$; i.e., the substitution of the free occurrences of $\underline{i}$ in a by b, updating the free variables in a (and in b). The term $\varphi^j_k\, a$ represents $j-1$ applications of the k-lift to a; i.e., $a^{+k^{(j-1)}}$. Table 3 gives the rules of the λs_e-calculus augmented with the rule (Eta), as introduced in [3].

$=_{s_e}$ denotes equality for the calculus s_e, induced by all the rules except (σ-generation) and (Eta).

Definition 9 $TA_{\lambda_{s_e}}$*, the* ***Simply Typed*** λs_e*, is given by the typing rules:*

(Var)	$A.\Gamma \vdash \underline{1} : A$	*(Varn)*	$\dfrac{\Gamma \vdash \underline{n} : B}{A.\Gamma \vdash \underline{n+1} : B}$
(Lambda)	$\dfrac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda_A.b : A \to B}$	*(App)*	$\dfrac{\Gamma \vdash a : A \to B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\ b) : B}$
(Sigma)	$\dfrac{\Gamma_{\geq i} \vdash b : B \quad \Gamma_{<i}.B.\Gamma_{\geq i} \vdash a : A}{\Gamma \vdash a\,\sigma^i b : A}$	*(Phi)*	$\dfrac{\Gamma_{\leq k}.\Gamma_{\geq k+i} \vdash a : A}{\Gamma \vdash \varphi^i_k\, a : A}$

$(\lambda_A.a\ b)$	$\longrightarrow$	$a\,\sigma^1 b$	(σ-generation)
$(\lambda_A.a)\,\sigma^i b$	$\longrightarrow$	$\lambda_A.(a\,\sigma^{i+1}b)$	(σ-λ-transition)
$(a_1\ a_2)\,\sigma^i b$	$\longrightarrow$	$((a_1\,\sigma^i b)\ (a_2\,\sigma^i b))$	(σ-app-transition)
$\underline{n}\,\sigma^i b$	$\longrightarrow$	$\begin{cases} \underline{n-1} & \text{if } n > i \\ \varphi^i_0\, b & \text{if } n = i \\ \underline{n} & \text{if } n < i \end{cases}$	(σ-destruction)
$\varphi^i_k\,(\lambda_A.a)$	$\longrightarrow$	$\lambda_A.(\varphi^i_{k+1}\,a)$	(φ-λ-transition)
$\varphi^i_k\,(a_1\ a_2)$	$\longrightarrow$	$((\varphi^i_k\,a_1)\ (\varphi^i_k\,a_2))$	(φ-app-transition)
$\varphi^i_k\,\underline{n}$	$\longrightarrow$	$\begin{cases} \underline{n+i-1} & \text{if } n > k \\ \underline{n} & \text{if } n \le k \end{cases}$	(φ-destruction)
$(a_1\,\sigma^i a_2)\,\sigma^j b$	$\longrightarrow$	$(a_1\,\sigma^{j+1}b)\,\sigma^i(a_2\,\sigma^{j-i+1}b)$ if $i \le j$	(σ-σ-transition)
$(\varphi^i_k\,a)\,\sigma^j b$	$\longrightarrow$	$\varphi^{i-1}_k\,a$ if $k < j < k+i$	(σ-φ-transition 1)
$(\varphi^i_k\,a)\,\sigma^j b$	$\longrightarrow$	$\varphi^i_k\,(a\,\sigma^{j-i+1}b)$ if $k+i \le j$	(σ-φ-transition 2)
$\varphi^i_k\,(a\,\sigma^j b)$	$\longrightarrow$	$(\varphi^i_{k+1}\,a)\,\sigma^j(\varphi^i_{k+1-j}\,b)$ if $j \le k+1$	(φ-σ-transition)
$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^j_l\,(\varphi^i_{k+1-j}\,a)$ if $l+j \le k$	(φ-φ-transition 1)
$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^{j+i-1}_l\,a$ if $l \le k < l+j$	(φ-φ-transition 2)
$\lambda_A.(a\ \underline{1})$	$\longrightarrow$	b if $a =_{s_e} \varphi^2_0\,b$	(*Eta*)

Table 3. The rewriting system of the λs_e-calculus

In this definition, $\Gamma_{\le i}$ denotes the context built as the ordered list of the first i types in Γ. The contexts $\Gamma_{<i}$, $\Gamma_{>i}$ and $\Gamma_{\ge i}$ are defined similarly.

Example 2 *As in Example 1, applying λs_e-rules to $(\lambda_C.\lambda_{C\to A}.(\underline{1}\ \underline{2}))\,\underline{2}$ leads to $\lambda_{C\to A}.((\underline{1}\,\sigma^2\underline{2})\ (\varphi^2_0\,\underline{2}))$. We show that $B.C \vdash_{TA_{\lambda s_e}} \lambda_{C\to A}.((\underline{1}\,\sigma^2\underline{2})\ (\varphi^2_0\,\underline{2})) : (C \to A) \to A$. Initially, one has $\dfrac{C \vdash \underline{1} : C\ (Var)}{B.C \vdash \underline{2} : C}(Varn)$. Thus*

$$\dfrac{\dfrac{\dfrac{B.C \vdash \underline{2} : C \quad C \to A.C.B.C \vdash \underline{1} : C \to A}{C \to A.B.C \vdash \underline{1}\,\sigma^2\underline{2} : C \to A}(\mathit{Sigma}) \quad \dfrac{B.C \vdash \underline{2} : C}{C \to A.B.C \vdash \varphi^2_0\,\underline{2} : C}(\mathit{Phi})}{C \to A.B.C \vdash ((\underline{1}\,\sigma^2\underline{2})\ (\varphi^2_0\,\underline{2})) : A}(\mathit{App})}{B.C \vdash \lambda_{C\to A}.((\underline{1}\,\sigma^2\underline{2})\ (\varphi^2_0\,\underline{2})) : (C \to A) \to A}(\mathit{Lambda})$$

For the simply typed λs_e without the rule (Eta), SR holds.

Theorem 3 (SR for λs_e without the Eta rule [10]) *If $\Gamma \vdash_{T\lambda s_e} a : A$ and $a \to_{\lambda s_e} a'$, then $\Gamma \vdash_{T\lambda s_e} a' : A$.*

3.2. An explicit Eta rule for the λs_e which preserves SR

Similarly to the $\lambda\sigma$-calculus, for the definition of (Eta) for λs_e given in Table 3, inherited from the usual definition based on lifting, when no restrictions are given on terms, derivation of ill-typed terms may happen. For instance, let $\Gamma \vdash \underline{i} : B$ and $\Gamma \vdash \underline{n} : A$, where $n < i$. Assume $\Gamma_{\ge i} \vdash N : C$, where $C \ne A$. By (σ-destruction), $\underline{n}\,\sigma^i N \to \underline{n}$. Note that $\underline{n}\,\sigma^i N$ is not typable. Thus, $\lambda(\underline{n+1}\ \underline{1}) \to \underline{n}\sigma^1 N$, because $\varphi^2_0(\underline{n}\sigma^i N) \to \varphi^2_0(\underline{n}) \to \underline{n+1}$ ($\underline{n+1} =_{s_e} \varphi^2_0(\underline{n}\sigma^i N)$). In order to give a constructive and explicit definition of Eta, one should define the free occurrences of $\underline{i}$ in a λs_e-term. We propose the following:

Definition 10 *Let $m, n \in \mathbb{N}^*$. The* ***Calculus of detection of Occurrences of Free Variables in***

λs_e*, denoted as COFV, is given by the following rules*

$$\frac{\langle \varphi^i_k\, a, n\rangle}{\langle a, n-i+1\rangle}\ \text{(if } n \geq k+i\text{)} \qquad \frac{\langle \underline{m}\,, n\rangle}{\textit{False}}\ \text{(if } m \neq n\text{)} \qquad \frac{\langle \lambda_A.a, n\rangle}{\langle a, n+1\rangle}$$

$$\frac{\langle a\,\sigma^i b, n\rangle}{\langle a, n\rangle}\ \text{(if } n < i\text{)} \qquad \frac{\langle a\,\sigma^i b, n\rangle}{\langle a, n+1\rangle \vee \langle b, n-i+1\rangle}\ \text{(if } n \geq i\text{)} \qquad \frac{\langle \underline{n}\,, n\rangle}{\textit{True}}$$

$$\frac{\langle \varphi^i_k\, a, n\rangle}{\langle a, n\rangle}\ \text{(if } n \leq k\text{)} \qquad \frac{\langle \varphi^i_k\, a, n\rangle}{\textit{False}}\ \text{(if } k < n < k+i\text{)} \qquad \frac{\langle (a\ b), n\rangle}{\langle a, n\rangle \vee \langle b, n\rangle}$$

With this definition the notion of occurrence of free variables in λs_e-terms can be formalized.

Definition 11 (Free indices in λs_e) *If $\langle a, i\rangle \vdash_{COFV} True$, we say a **has a free occurrence of** $\underline{i}$.*

We let $N_{s_e}(a)$ denote the normal form of a with respect to the s_e-calculus.

Lemma 5 *Let a, b, λs_e-terms, in normal forms with respect to the s_e-calculus.*

1. *For $i \leq k$, $N_{s_e}(\varphi^j_k a)$ has a free occurrence of $\underline{i}$ if, and only if, a has too.*
2. *For $k < i < k+j$, $N_{s_e}(\varphi^j_k a)$ has no free occurrence of $\underline{i}$.*
3. *For $i \geq k+j$, $N_{s_e}(\varphi^j_k a)$ has a free occurrence of $\underline{i}$ if, and only if, a has a free occurrence of $\underline{i-j+1}$.*
4. *For $i < j$, $N_{s_e}(a\,\sigma^j b)$ has a free occurrence of $\underline{i}$ if, and only if, a has too.*
5. *For $i \geq j$, $N_{s_e}(a\,\sigma^j b)$ has a free occurrence of $\underline{i}$ if, and only if, either a has a free occurrence of $\underline{i+1}$ or a has a free occurrence of $\underline{j}$ and b has a free occurrence of $\underline{i-j+1}$.*

Lemma 6 *If $N_{s_e}(a)$ has free occurrences of $\underline{i}$, then $\langle a, i\rangle \vdash_{COFV} True$.*

The rewriting system for checking η-redices in λs_e, denoted as $R_{\eta s_e}$, is given in Table 4. Note that the language of λs_e is enlarged with the symbol η.

$(a\ b)\,\eta^i$	$\longrightarrow$	$(a\,\eta^i\ b\,\eta^i)$	(η-app-transition)
$(\lambda_A.a)\,\eta^i$	$\longrightarrow$	$\lambda_A.a\,\eta^{i+1}$	(η-λ-transition)
$\underline{n}\,\eta^i$	$\longrightarrow$	$\begin{cases} \underline{n} & \text{if } n < i \\ \underline{n-1} & \text{if } n > i \end{cases}$	(η-destruction)
$(a\,\sigma^j b)\,\eta^i$	$\longrightarrow$	$(a\,\eta^i)\,\sigma^{j-1} b$ if $i < j$	(η-σ-transition 1)
$(a\,\sigma^j b)\,\eta^i$	$\longrightarrow$	$(a\,\eta^{i+1})\,\sigma^j (b\,\eta^{i-j+1})$ if $i \geq j$	(η-σ-transition 2)
$(\varphi^j_k\, a)\,\eta^i$	$\longrightarrow$	$\varphi^j_{k-1}\,(a\,\eta^i)$ if $i \leq k$	(η-φ-transition 1)
$(\varphi^j_k\, a)\,\eta^i$	$\longrightarrow$	$\varphi^{j-1}_k\ a$ if $k < i < k+j$	(η-φ-transition 2)
$(\varphi^j_k\, a)\,\eta^i$	$\longrightarrow$	$\varphi^j_k\,(a\,\eta^{i-j+1})$ if $i > k$ and $i \geq k+j$	(η-φ-transition 3)

Table 4. $R_{\eta s_e}$: the rewriting system for η-reduction in λs_e

Note that the $R_{\eta s_e}$-rules have similar structure to the rules of detection of free variables of Definition 10, with a simple adaptation for checking and updating the free variables of the rule (η-destruction), and for updatings coming from the elimination of abstractors from η-reduction, given in the transition rules. $R_{\eta s_e}$ is easily checked to be terminating and confluent. For the former, notice that for any term a, the $R_{\eta s_e}$ normalisation of $a\eta$ is simply a propagation of the symbol η between the finite structure of a. For the later, notice that $R_{\eta s_e}$ is left linear and there are no possible overlapping between left-hand sides of the rules; thus, by orthogonality, confluence holds.

One has the following property of $R_{\eta s_e}$ which guarantees reductions for λs_e-terms.

Lemma 7 *If $\langle a, i\rangle \vdash_{COFV} False$, then $N_{R_{\eta s_e}}(a\,\eta^i)$ has no occurrence of the operator η.*

Now it is possible to give the following definition of (Eta).

Definition 12 *Let $\lambda_A.(a\ \underline{1}\,)$ be a λs_e-term. The rule **(Eta)** is given by*

$$\lambda_A.(a\ \underline{1}\,) \to N_{R_{\eta s_e}}(a\,\eta^1) \qquad \textit{if } \langle a, 1\rangle \vdash_{COFV} False \qquad \textit{(Eta)}$$

By Lemma 7, the condition $\langle a, 1\rangle \vdash_{COFV} False$ guarantees that $N_{R_{\eta s_e}}(a\,\eta^1)$ is a λs_e-term. The following property of $R_{\eta s_e}$, related to types, is necessary in order to prove SR for the proposed rule (Eta).

Lemma 8 *If $\Gamma \vdash a : A$ and $\langle a, i\rangle \vdash_{COFV} False$, then $\Gamma_{<i}.\Gamma_{>i} \vdash N_{R_{\eta s_e}}(a\,\eta^i) : A$.*

Theorem 4 (SR for (Eta)) *If $\Gamma \vdash \lambda_B.(a\ \underline{1}\,) : A$ and $\lambda_B.(a\ \underline{1}\,) \to_{Eta} b$, then $\Gamma \vdash b : A$.*

Proof. Suppose that $\Gamma \vdash \lambda_B.(a\ \underline{1}\,) : A$. By (Lambda) one has that $B.\Gamma \vdash (a\ \underline{1}\,) : C$, where $A = B \to C$. By (App) and (Var) one has that $B.\Gamma \vdash a : B \to C = A$ and $B.\Gamma \vdash \underline{1} : B$. By hypothesis $\langle a, 1\rangle \vdash_{COFV} False$ and $b = N_{R_{\eta s_e}}(a\,\eta^1)$. Consequently, by case $i = 1$ of Lemma 8, one obtains $\Gamma \vdash b : A$. □

When the λs_e-term a of the Eta rule has no free occurrences of $\underline{1}$, deciding the applicability of the Eta rule and building the η-contractum are practically equivalent processes. Then, a straightforward adaptation of the calculus $R_{\eta s_e}$ will do both tasks: detecting the presence of free occurrences of $\underline{1}$ in a and, in the negative case, simultaneously building the corresponding η-contractum. This is possible by changing the rule η-destruction in Table 4 to

$$\underline{n}\,\eta^i \quad \longrightarrow \quad \begin{cases} \underline{n} & \text{if } n < i \\ error & \text{if } n = i \\ \underline{n-1} & \text{if } n > i \end{cases} \qquad (\eta\text{-destruction2})$$

The new system $R'_{\eta s_e}$, allows the following definition of Eta which does both tasks simultaneously.

$$\lambda_A.(a\ \underline{1}\,) \to N_{R'_{\eta s_e}}(a\,\eta^1) \qquad \text{if } N_{R'_{\eta s_e}}(a\,\eta^1) \text{ is a } \lambda s_e\text{-term} \qquad \text{(Eta2)}$$

In implementations, (Eta2) is more adequate than the first new version of (Eta) which duplicates work.

4. Conclusions and Future Work

We defined constructive explicit Eta rules for the $\lambda\sigma$- and the λs_e-calculi which preserve SR. This formalization involves the presentation of specific sub-calculi, given as well-behaved rewriting systems, for verifying the condition of Eta in each of these two calculi. The proposed definitions work in such a way that the construction of the Eta-contractum is given, while the condition of the rule is checked. In this way, we presented a formalization which is directly implementable from the constructive definition of the Eta rules for the simply-typed version of these calculi. In addition to making explict the definitions of Eta in [7, 6] and [3], our work contributes to making precise the informal implementations of these rules suggested in [4] and [2] respectively. In contrast with the rule Eta introduced in [7] (for $\lambda\sigma$) our constructive definition can be applied to non (σ-)normal forms in such a way that the rules Beta and Eta are put on an equal footing.

As future work, it is interesting to compare the efficiency of the implementation of the suggested Eta rules in both calculi. Superficially, notice that the condition "$\diamond$ occurs in" for $R_{\eta_{\lambda\sigma}}$, suggests a quadratic verification of the applicability of Eta in $\lambda\sigma$, while this appears to be linear in λs_e. In order to be conclusive, it is necessary to discard the possibility of doing this task correctly in $\lambda\sigma$ in a more efficient manner.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Functional Programming*, 1(4):375–416, 1991.

[2] M. Ayala-Rincón, F. de Moura, and F. Kamareddine. Comparing and Implementing Calculi of Explicit Substitutions with Eta-Reduction. *Annals of Pure and Applied Logic*, 134:5–41, 2005.

[3] M. Ayala-Rincón and F. Kamareddine. Unification via the λs_e-Style of Explicit Substitution. *Interest Group in Pure and Applied Logics*, 9(4):489–523, 2001.

[4] P. Borovanský. Implementation of Higher-Order Unification Based on Calculus of Explicit Substitutions. Volume 1012 of *LNCS*, pages 363–368. 1995.

[5] D. Briaud. An explicit Eta rewrite rule. In *Typed lambda calculi and applications*, volume 902 of *LNCS*, pages 94–108. 1995.

[6] G. Dowek, T. Hardin, and C. Kirchner. Higher-order Unification via Explicit Substitutions. *Information and Computation*, 157(1/2):183–235, 2000.

[7] T. Hardin. Eta-conversion for the languages of explicit substitutions. In *Algebraic and logic programming*, volume 632 of *LNCS*, pages 306–321. 1992.

[8] J. R. Hindley. *Basic Simple Type Theory*. Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[9] D. Kesner. Confluence of extensional and non-extensional λ-calculi with explicit substitutions. *TCS*, 238(1-2):183–220, 2000.

[10] F. Kamareddine and A. Ríos. Relating the Lambda-sigma and Lambda-s styles of Explicit Substitutions. Logic and Computation volume, 10(3): 349–380. 2000.

[11] P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate in Proceedings of TLCA'95. *LNCS*, 902, 1995.

[12] A. Ríos. *Contribution à l'étude des λ-calculs avec substitutions explicites*. PhD thesis, Université de Paris 7, 1993.

A. Proofs

Proof. (Lemma 5) All the proofs are by induction on the structure of a. The notation $N(a)$ stands for $N_{s_e}(a)$.

1.
   - $a = \underline{n}$: by (φ-destruction), $\varphi^j_k \underline{n} \to \underline{n}$. Thus, $N(\varphi^j_k \underline{n})$ has a free occurrence of $\underline{i}$ iff $n = i$.
   - $a = (c\ d)$: by (φ-app-transition), $\varphi^j_k (c\ d) \to (\varphi^j_k c\ \varphi^j_k d)$. By IH, $N(\varphi^j_k c)$ has a free occurrence of $\underline{i}$ iff c has too. Analogously for $N(\varphi^j_k d)$ and d. Note that $N(\varphi^j_k a) = (N(\varphi^j_k c)\ N(\varphi^j_k d))$. Thus, $N(\varphi^j_k a)$ has a free occurrence of $\underline{i}$ iff either c or d has too.

- $a = \lambda_A.c$: by (φ-λ-transition), $\varphi^j_k(\lambda_A.c) \rightarrow \lambda_A.(\varphi^j_{k+1}c)$. Since $i+1 \leq k+1$, by IH, $N(\varphi^j_{k+1}c)$ has a free occurrence of $\underline{i+1}$ iff c has too. Hence, $N(\varphi^j_{k+1}a)$ has a free occurrence of $\underline{i}$ iff a has too.

2.
- $a = \underline{n}$: If $n \leq k$, then by (φ-destruction) one has $\varphi^j_k\underline{n} \rightarrow \underline{n}$, where $n \leq k < i$. If $n > k$, then by (φ-destruction) one has $\varphi^j_k\underline{n} \rightarrow \underline{n+j-1}$, where $n+j-1 > k+j-1 \geq i$. Thus $n+j-1 > i$.
- $a = (c\ d)$: by (φ-app-transition), $\varphi^j_k(c\ d) \rightarrow (\varphi^j_k c\ \varphi^j_k d)$. Thus, by IH, neither $N(\varphi^j_k c)$ nor $N(\varphi^j_k d)$ has a free occurrence of $\underline{i}$.
- $a = \lambda_A.c$: by (φ-λ-transition), $\varphi^j_k(\lambda_A.c) \rightarrow \lambda_A.(\varphi^j_{k+1}c)$. $k+1 < i+1 < (k+1)+j$, thus, by IH, $N(\varphi^j_k c)$ has no free occurrence of $\underline{i+1}$. Hence, $N(\varphi^j_{k+1}a)$ has no free occurrence of $\underline{i}$.

3.
- $a = \underline{n}$: If $n \leq k$, then by (φ-destruction), $\varphi^j_k\underline{n} \rightarrow \underline{n}$, where $i \geq k+j > k \geq n$. If $n > k$, then by (φ-destruction), $\varphi^j_k\underline{n} \rightarrow \underline{n+j-1}$. Thus, $N(\varphi^j_k\underline{n})$ has a free occurrence of $\underline{i}$ iff $n = i-j+1$.
- $a = (c\ d)$: by (φ-app-transition), $\varphi^j_k(c\ d) \rightarrow (\varphi^j_k c\ \varphi^j_k d)$. By IH, $N(\varphi^j_k a) = (N(\varphi^j_k c)\ N(\varphi^j_k d))$ has a free occurrence of $\underline{i}$ iff either c or d has a free ocurrence of $\underline{i-j+1}$.
- $a = \lambda_A.c$: by (φ-λ-transition), $\varphi^j_k(\lambda_A.c) \rightarrow \lambda_A.(\varphi^j_{k+1}c)$. Since $i+1 \geq (k+1)+j$, by IH, $N(\varphi^j_{k+1}c)$ has a free ocurrence of $\underline{i+1}$ iff c has a free ocurrence of $\underline{(i+1)-j+1}$. Hence, $N(\varphi^j_{k+1}a)$ has a free occurrence of $\underline{i}$ iff a has a free ocurrence of $\underline{i-j+1}$.

4.
- $a = \underline{n}$: If $n < j$, then by (σ-destruction), $\underline{n}\,\sigma^j b \rightarrow \underline{n}$. Thus, $N(\underline{n}\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff $n = i$. If $n = j$, then by (σ-destruction), $\underline{n}\,\sigma^j b \rightarrow \varphi^j_0\, b$. Since $0 = k < i < k+j$, by item 2, $N(\varphi^j_0\, b)$ has no free occurrence of $\underline{i}$. If $n > j$, then by (σ-destruction), $\underline{n}\,\sigma^j b \rightarrow \underline{n-1}$, where $n-1 \geq j > i$.
- $a = (c\ d)$: by (σ-app-transition), $(c\ d)\sigma^j b \rightarrow ((c\,\sigma^j b)\ (d\,\sigma^j b))$. Thus, by IH, $N(a\,\sigma^j b) = (N(c\,\sigma^j b)\ N(d\,\sigma^j b))$ has a free occurrence of $\underline{i}$ iff either c or d has too.
- $a = \lambda_A.c$: by (σ-λ-transition), $(\lambda_A.c)\sigma^j b \rightarrow \lambda_A.(c\,\sigma^{j+1}b)$. Since $i+1 < j+1$, by IH, $N(c\,\sigma^{j+1}b)$ has a free ocurrence of $\underline{i+1}$ iff c has a free ocurrence of $\underline{i+1}$. Hence, $N(a\,\sigma^{j+1}b)$ has a free ocurrence of $\underline{i}$ iff a has a free ocurrence of $\underline{i}$.

5.
- $a = \underline{n}$: If $n < j$, then by (σ-destruction), $\underline{n}\,\sigma^j b \rightarrow \underline{n}$, where $n < j \leq i$. If $n = j$, then by (σ-destruction), $\underline{n}\,\sigma^j b \rightarrow \varphi^j_0\, b$. Since $i \geq k+j$ where $k = 0$, by item 3, $N(\varphi^j_0\, b)$ has free occurrence of $\underline{i}$ iff b has a free occurrence of $\underline{i-j+1}$. If $n > j$, then by (σ-destruction) one has $\underline{n}\,\sigma^j b \rightarrow \underline{n-1}$. Thus, $N(\underline{n}\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff $n = i+1$. Observe that $i+1 > i \geq j$.
- $a = (c\ d)$: by (σ-app-transition), $(c\ d)\sigma^j b \rightarrow ((c\,\sigma^j b)\ (d\,\sigma^j b))$. Thus, by IH, $N(a\,\sigma^j b) = (N(c\,\sigma^j b)\ \ N(d\,\sigma^j b))$ has a free occurrence of $\underline{i}$ iff either c or d has a free occurrence of $\underline{i+1}$ or either c or d has a free occurrence of $\underline{j}$ and b has a free occurrence of $\underline{i-j+1}$.
- $a = \lambda_A.c$: by (σ-λ-transition), $(\lambda_A.c)\sigma^j b \rightarrow \lambda_A.(c\,\sigma^{j+1}b)$. Since $i+1 \geq j+1$, thus, by IH, $N(c\,\sigma^{j+1}b)$ has a free ocurrence of $\underline{i+1}$ iff c has a free ocurrence of $\underline{(i+1)+1}$ or c has a free ocurrence of $\underline{j+1}$ and b has a free ocurrence of $\underline{(i+1)-(j+1)+1} = \underline{i-j+1}$. Consequently, $N(a\,\sigma^{j+1}b)$ has a free

ocurrence of $\underline{i}$ if, and only if, a has a free ocurrence of $\underline{i+1}$ or a has a free ocurrence of $\underline{j}$ and b has a free ocurrence of $\underline{i-j+1}$. $\square$

Proof. **(Lemma 6)** By induction on the structure of a.

1) $a = \underline{n}$: If $n = i$, then $\langle \underline{n}, i\rangle \vdash True$
2) $a = (b\ c)$: we have $\langle (b\ c), i\rangle \vdash \langle b, i\rangle \vee \langle c, i\rangle$. By IH, if $N_{s_e}(b)$ has free occurrences of $\underline{i}$, then $\langle b, i\rangle \vdash True$. Analogously for $N_{s_e}(c)$. Thus if $N_{s_e}(b)$ or $N_{s_e}(c)$ have free occurrences of $\underline{i}$, then $\langle (b\ c), i\rangle \vdash True$. Note that $N_{s_e}(b\ c) = (N_{s_e}(b)\ N_{s_e}(c))$.
3) $a = \lambda_A.b$: we have $\langle \lambda_A.b, i\rangle \vdash \langle b, i+1\rangle$. By IH, if $N_{s_e}(b)$ has free occurrences of $\underline{i+1}$, then $\langle b, i+1\rangle \vdash True$. Thus, if $N_{s_e}(a) = \lambda_A.N_{s_e}(b)$ has free occurrences of $\underline{i}$, then $\langle a, i\rangle \vdash True$.
4) $a = b\,\sigma^j c$: if $i < j$, then $\langle b\,\sigma^j c, i\rangle \vdash \langle b, i\rangle$. By Lemma 5.4, if $N_{s_e}(b\,\sigma^j c)$ has free occurrences of $\underline{i}$, then $N_{s_e}(b)$ has free occurrences of $\underline{i}$. Thus, by IH, $\langle b, i\rangle \vdash True$.
If $i \geq j$, then $\langle b\,\sigma^j c, i\rangle \vdash \langle b, i+1\rangle \vee \langle c, i-j+1\rangle$. by Lemma 5.5, if $N_{s_e}(b\,\sigma^j c)$ has free occurrences of $\underline{i}$, then $N_{s_e}(b)$ has free occurrences of $\underline{i+1}$ or $N_{s_e}(b)$ has free occurrences of $\underline{j}$ and $N_{s_e}(c)$ has free occurrences of $\underline{i-j+1}$. Thus, by IH, $\langle a, i\rangle \vdash True$. Observe that even if $N_{s_e}(b)$ has no free occurrences of $\underline{j}$, $\langle a, i\rangle$ is still assigned $True$.
5) $a = \varphi^j_k\, b$: if $i \leq k$, then $\langle \varphi^j_k\, b, i\rangle \vdash \langle b, i\rangle$. By Lemma 5.1, if $N_{s_e}(\varphi^j_k\, b)$ has free occurrences of $\underline{i}$, then $N_{s_e}(b)$ has too. Thus, by IH, $\langle b, i\rangle \vdash True$.
If $k < i < k+j$, then $\langle \varphi^j_k\, b, i\rangle \vdash False$. By Lemma 5.2, $N_{s_e}(\varphi^j_k\, b)$ has no free occurrences of $\underline{i}$.
If $i \geq k+j$, then $\langle \varphi^j_k\, b, i\rangle \vdash \langle b, i-j+1\rangle$. By Lemma 5.3, if $N_{s_e}(\varphi^j_k\, b)$ has free occurrences of $\underline{i}$, then $N_{s_e}(b)$ has free occurrences of $\underline{i-j+1}$. Thus, by IH, $\langle b, i-j+1\rangle \vdash True$. $\square$

Proof. **(Lemma 7)** By structural induction on a.

1) $a = \underline{n}$: from $\langle \underline{n}, i\rangle \vdash False$ one has that $n \neq i$. If $n < i$, then $\underline{n}\,\eta^i \to \underline{n}$. If $n > i$, then $\underline{n}\,\eta^i \to \underline{n-1}$.
2) $a = (b\ c)$: one has that $(b\ c)\,\eta^i \to (b\,\eta^i\ c\,\eta^i)$. From $\langle (b\ c), i\rangle \vdash False$ one has that $\langle b, i\rangle \vdash False$ and $\langle c, i\rangle \vdash False$. Thus, by IH, neither $N_{R_{\eta s_e}}(b\,\eta^i)$ nor $N_{R_{\eta s_e}}(c\,\eta^i)$ have occurrences of η.
3) $a = \lambda_B.b$: one has $(\lambda_B.b)\,\eta^i \to \lambda_B.b\,\eta^{i+1}$. From $\langle \lambda_B.b, i\rangle \vdash False$ one has that $\langle b, i+1\rangle \vdash False$. Thus, by IH, one has that $N_{R_{\eta s_e}}(b\,\eta^{i+1})$ has no occurrence of η.
4) $a = b\,\sigma^j c$: one has $\langle b\,\sigma^j c, i\rangle \vdash False$.
If $i < j$, then $(b\,\sigma^j c)\,\eta^i \to (b\,\eta^i)\,\sigma^{j-1} c$ and $\langle b, i\rangle \vdash False$. By IH, $N_{R_{\eta s_e}}(b\,\eta^i)$ has no occurrence of η.
If $i \geq j$, then $(b\,\sigma^j c)\,\eta^i \to (b\,\eta^{i+1})\,\sigma^j(c\,\eta^{i-j+1})$, $\langle b, i+1\rangle \vdash False$ and $\langle c, i-j+1\rangle \vdash False$. Thus, by IH, neither $N_{R_{\eta s_e}}(b\,\eta^{i+1})$ nor $N_{R_{\eta s_e}}(c\,\eta^{i-j+1})$ have occurrences of η.
5) $a = \varphi^j_k\, b$: one has that $\langle \varphi^j_k\, b, i\rangle \vdash False$.
If $i \leq k$, then $(\varphi^j_k\, b)\,\eta^i \to \varphi^j_{k-1}(b\,\eta^i)$ and $\langle b, i\rangle \vdash False$. By IH, $N_{R_{\eta s_e}}(b\,\eta^i)$ has no occurrences of η.
If $k < i < k+j$, then $(\varphi^j_k\, b)\,\eta^i \to \varphi^{j-1}_k\, b$ and since b is a λs_e-term, it has no occurrences of η.
If $i \geq k+j$, then $(\varphi^j_k\, b)\,\eta^i \to \varphi^j_k(b\,\eta^{i-j+1})$ and $\langle b, i-j+1\rangle \vdash False$. By IH, $N_{R_{\eta s_e}}(b\,\eta^{i-j+1})$ has no occurrences of η. $\square$

Proof. **(Lemma 8)** By induction on the structure of a. We write $N(a)$ for the $R_{\eta s_e}$-normal form of a.

1) $a = \underline{n}$: Let $\Gamma \vdash \underline{n} : A$. By $\langle \underline{n}, i\rangle \vdash_{COFV} False$, $n \neq i$. If $n < i$, by (η-destruction), $\underline{n}\,\eta^i \to \underline{n}$ and $\Gamma_{<i}.\Gamma_{>i} \vdash \underline{n} : A$. If $n > i$, by (η-destruction), $\underline{n}\,\eta^i \to \underline{n-1}$ and by (Varn), $\Gamma_{>i} \vdash \underline{n-i} : A$ and hence by $i-1$ applications of (Varn), $\Gamma_{<i}.\Gamma_{>i} \vdash \underline{n-1} : A$
2) $a = (b\ c)$: Let $\Gamma \vdash (b\ c) : A$. By (η-app-transition), $N((b\ c)\,\eta^i) = (N(b\,\eta^i)\ N(c\,\eta^i))$. By (App), $\Gamma \vdash b : B \to A$ and $\Gamma \vdash c : B$. By IH, $\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^i) : B \to A$ and $\Gamma_{<i}.\Gamma_{>i} \vdash N(c\,\eta^i) : B$. Thus, by (App), $\Gamma_{<i}.\Gamma_{>i} \vdash (N(b\,\eta^i)\ N(c\,\eta^i)) : A$.
3) $a = \lambda_B.b$: Let $\Gamma \vdash \lambda_B.b : A$. By ($\eta$-$\lambda$-transition), $N((\lambda_B.b)\,\eta^i) = \lambda_B.N(b\,\eta^{i+1})$. By (Lambda), $B.\Gamma \vdash b : C$, where $A = B \to C$. By IH, $B.\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^{i+1}) : C$. By (Lambda), $\Gamma_{<i}.\Gamma_{>i} \vdash \lambda_B.N(b\,\eta^{i+1}) : A$.
4) $a = b\,\sigma^j c$: Let $\Gamma \vdash b\,\sigma^j c : A$. By (Sigma), $\Gamma_{\geq j} \vdash c : B$ and $\Gamma_{<j}.B.\Gamma_{\geq j} \vdash b : A$.
   If $i < j$, by (η-σ-transition 1), $N((b\,\sigma^j c)\,\eta^i) = (N(b\,\eta^i))\,\sigma^{j-1}c$. By IH, $\Gamma_{<i}.(\Gamma_{<j})_{>i}.B.\Gamma_{\geq j} \vdash N(b\,\eta^i) : A$. By (Sigma), $\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^i)\,\sigma^{j-1}c : A$.
   If $i \geq j$, by (η-σ-transition 2), $N((b\,\sigma^j c)\,\eta^i) = N(b\,\eta^{i+1})\,\sigma^j N(c\,\eta^{i-j+1})$. By IH, $\Gamma_{<j}.B.(\Gamma_{\geq j})_{<i-j+1}.\Gamma_{>i} \vdash N(b\,\eta^{i+1}) : A$ and $(\Gamma_{\geq j})_{<i-j+1}.\Gamma_{>i} \vdash N(c\,\eta^{i-j+1}) : B$. By (Sigma) $\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^{i+1})\,\sigma^j N(c\,\eta^{i-j+1}) : A$.
5) $a = \varphi^j_k\,b$: Let $\Gamma \vdash \varphi^j_k\,b : A$. By (Phi), $\Gamma_{\leq k}.\Gamma_{\geq k+j} \vdash b : A$.
   If $i \leq k$, by (η-φ-transition 1), $N((\varphi^j_k\,b)\,\eta^i) = \varphi^j_{k-1}\,N(b\,\eta^i)$. By IH, $\Gamma_{<i}.(\Gamma_{\leq k})_{>i}.\Gamma_{\geq k+j} \vdash N(b\,\eta^i) : A$. Thus, by (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi^j_{k-1}\,N(b\,\eta^i) : A$.
   If $k < i < k+j$, by (η-φ-transition 2), $N((\varphi^j_k\,b)\,\eta^i) = \varphi^{j-1}_k\,b$. Thus, by (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi^{j-1}_k\,b : A$.
   If $k < i$ and $k+j \leq i$, by (η-φ-transition 3), $N((\varphi^j_k\,b)\,\eta^i) = \varphi^j_k\,N(b\,\eta^{i-j+1})$.
   By IH, $\Gamma_{\leq k}.(\Gamma_{\geq k+j})_{<i-j+1-k}.\Gamma_{>i} \vdash N(b\,\eta^{i-j+1}) : A$. By (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi^j_k\,N(b\,\eta^{i-j+1}) : A$. $\square$