# Explicit Substitutions Calculi with One Step Eta-reduction Decided Explicitly

DANIEL VENTURA, *Grupo de Teoria da Computação, Departamento de Matemática, Universidade de Brasília, Brasília D.F., Brasil.* *ventura@mat.unb.br*

MAURICIO AYALA-RINCÓN, *Grupo de Teoria da Computação, Departamento de Matemática, Universidade de Brasília, Brasília D.F., Brasil.* *ayala@mat.unb.br*

FAIROUZ KAMAREDDINE, *School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, Scotland.* *fairouz@macs.hw.ac.uk*

## Abstract

It has long been argued that the notion of substitution in the $\lambda$-calculus needs to be made explicit. This resulted in many calculi have been developed in which the computational steps of the substitution operation involved in $\beta$-contractions have been atomised. In contrast to the great variety of developments for making explicit formalisations of the Beta rule, less work has been done for giving explicit definitions of the conditional Eta rule. In this paper constructive Eta rules are proposed for both the $\lambda\sigma$- and the $\lambda s_e$-calculi of explicit substitutions. Our results can be summarised as follows: 1) we introduce constructive and explicit definitions of the Eta rule in the $\lambda\sigma$- and the $\lambda s_e$-calculi, 2) we prove that these definitions are correct and preserve basic properties such as subject reduction. In particular, we show that the explicit definitions of the eta rules coincide with the Eta rule for pure $\lambda$-terms and that moreover, their application is decidable in the sense that Eta redices are effectively detected (and contracted). The formalisation of these Eta rules involves the development of specific calculi for explicitly checking the condition of the proposed Eta rules while constructing the Eta contractum.

*Keywords*: Lambda Calculus, Eta reduction, Explicit Substitutions, Subject Reduction

## 1 Introduction

Well-behaved calculi of explicit substitutions are a useful bridge between the formal study of the $\lambda$-calculus and its real implementations. Since $\beta$-contractions depend on the definition of the operation of substitution which in many papers is informally given, most computational environments develop in an *ad-hoc* way an explicit notion of substitution.

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 normalisation, etc. This is a non trivial task; for instance, the $\lambda\sigma$-calculus [1] was reported to break the last property after some years of its introduction [13]: this implies that infinite derivations starting from well-typed $\lambda$-terms are possible in this calculus.

The $\lambda\sigma$-calculus was enlarged with a non constructive Eta rule in [8], where by constructive Eta rule we understand a definition of Eta which specifies how to algorithmically detect and compute eta-redices and their eta-contracta. This enlarged calculus was used in [7] for treating higher-order unification (HOU) problems. For the $\lambda s_e$ (introduced in [11]), a similar extension is presented in [3]. When restricted to well-typed terms normalised w.r.t. the associated substitution calculus, the presentations of these rules preserve the subject reduction property (for short SR). Despite the necessity to guarantee SR, in [7] and [3] this restriction was not given as part of the definitions.

In this paper we formalise adequate unrestricted Eta rules in the simply typed versions of $\lambda\sigma$ and $\lambda s_e$. These definitions involve a constructive treatment of the generation of the Eta-contracta while deciding simultaneously whether the Eta rule is applicable; in other words, Eta redices are effectively detected and contracted simultaneously. For doing this, we introduce well-behaved rewriting calculi that are applied to check the conditions of the Eta rules. The $\lambda\sigma$-calculus is extended with a set of substitutions $\eta_i^j$, used by the rewriting rules of the new calculus to detect free occurrences and update free indices. The rewriting calculus is shown to be convergent for a subset of this enlarged set of expressions, in which the original syntax of $\lambda\sigma$ is included. Similarly, the $\lambda s_e$-calculus is extended with an operator $\eta$, for detections and updates, and the rewriting calculus is shown to be convergent for the whole enlarged set of terms. Furthermore, we prove that the proposed constructive Eta rules preserve SR.

Related works include the proposed implementations of Eta rules for $\lambda\sigma$ and $\lambda s_e$, presented in [5] and [2] respectively, that can be considered as informal versions of constructive Eta rules. Also, in [6] the $\eta$-contraction in the $\lambda\upsilon$-calculus is turned explicit through an unconditional Eta rule and an extension of the associated substitution calculus $\upsilon$, and in [10] the $\eta$-expansion, rather than $\eta$-contraction, was formulated to a general scheme for explicit substitution calculi. The $\eta$-expansion is relevant in HOU; in fact, in Huet's HOU method the "$\eta$-rule" is defined as the Eta expansion, which makes the method more efficient. Nonetheless, in Huet's method the $\eta$-expansion is restricted and it is guided by the types of the terms, permitting the application of many rewriting steps of Huet's $\eta$-rule in a unique step. In contrast to the importance of $\eta$-expansion in Huet's method, the separate application of $\eta$-contraction is of principal interest in HOU via explicit substitutions[1], and it motivates the study of an explicit $\eta$.

In Section 2, we present the motivation to formalise a constructive Eta rule in the

---

[1] The normalisation 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 $\beta\eta$-normalised (we use $\eta$-contraction!); secondly, these normal forms are converted to $\eta$-*long normal forms*. This corresponds to $\eta$-expansions which are normally done in a unique step according to the type of the term.

An analogous situation occurs in $\lambda s_e$. The practical evidence can be found in [5] where *weak $\eta$-normal forms* are defined which avoids the complete construction of $\eta$-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 specialised dec-application rules based on adequate application of *weak long/head normal forms*; the rule of normalisation is dropped but the $(\eta\beta)$ normalisation is used as part of the rule of expansion of applications exp-app.

$\lambda$-calculus in de Bruijn notation and we introduce the necessary background on the $\lambda\sigma$- and $\lambda s_e$-calculi. Simply typed versions of both calculi are presented in Section 3. In Section 4, we enlarge $\lambda\sigma$ with a constructive Eta rule which is shown to be well-defined and to preserve SR. In Section 5, the same is done for $\lambda s_e$. In Section 6, we conclude and present future work.

## 2   Background

We assume familiarity with the simply typed $\lambda$-calculus, $TA_\lambda$ (cf [9]) and its version in de Bruijn's notation. A standard rewriting notation as in [4] is adopted. Then, for a rewriting system $R$, $\to_R$ denotes the application of one of its rules and $\to_r$ the application of a specific rule $r$. The reflexive, transitive closure of $R$ is denoted by $\to_R^*$, $\to_R^+$ denotes its transitive closure and $\to_R^i$ a number $i$ of applications.

### 2.1   Motivation: non constructive definitions of $\eta$-reduction

The usual and non constructive definition of $\eta$-reduction given in the literature for the $\lambda$-calculus in de Bruijn notation is given by the rewriting rule

$$\lambda.(a\,\underline{1}) \to_\eta b \quad \text{if } b^+ = a$$

where $b^+$ denotes the *lifting* of $b$, where lifting is an operator which increases by one the free indices in its argument $b^2$. Although this definition of $\eta$-reduction appears recurrently in the literature, it has the drawback that it does not specify how to build $b$ from $a$. This kind of non constructive definition of $\eta$ has been adapted to several variants of the $\lambda$-calculus as the two explicit substitutions calculi we treat here.

In order to give a constructive Eta rule for $\lambda$-terms (in de Bruijn notation), we use a counterpart of the lifting operator, as is done in [14].

DEFINITION 2.1 (i-dash)
Let $a$ be a $\lambda$-term in de Bruijn notation. 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. $\underline{n}^{-i} = \begin{cases} \underline{n-1}, & \text{if } n > i \\ \text{undefined}, & \text{if } n = i \\ \underline{n}, & \text{if } n < i. \end{cases}$

The **dash** of a term $a$ is its 1-dash, denoted as $a^-$. Provided $a^-$ is well-defined, $(a^-)^+ = a$. This happens when $a$ has no free occurrences of $\underline{1}$. This gives rise to an adequate definition of $\eta$-reduction:

DEFINITION 2.2 (Constructive (Eta) for $\lambda$)
The constructive **Eta rule** for $\lambda$-terms in de Bruijn notation is given by

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

Observe that the $\eta$-contractum is constructed at the same time the condition for $\eta$-contraction is verified.

---

[2] For our purposes the updating operator for the $\lambda$-calculus in de Bruijn notation, denoted as $U_i^j$ (see [11]), is unnecessary. This updating operator is used when formalising the Beta rule: in a Beta contraction of a redex of the form $(\lambda.a\ b)$, free de Bruijn indices in $\lambda.a$ should be decremented and free indices of $b$ may be incremented and this is controlled by a meta-substitution and (the super and subscripts of) the updating operator ([11]). For the Eta rule lifting is enough, since only incrementing indices by one is necessary. Lifting corresponds to $U_0^2$.

## 2.2    The $\lambda\sigma$-Calculus

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

DEFINITION 2.3 (Set $\Lambda_\sigma$ of $\lambda\sigma$-expressions)
The syntax of the $\lambda\sigma$-calculus is given by the set $\Lambda_\sigma$ of $\lambda\sigma$-expressions, which contains the following terms and substitutions:

**Terms**    $a ::= \underline{1} \mid (a\ a) \mid \lambda.a \mid a[s]$          **Substitutions**    $s ::= id \mid\ \uparrow\ \mid a.s \mid s \circ s$

The intuitive semantics for substitution $s$ is a set or a list of objects of the form $b/\underline{i}$ indicating that the index $\underline{i}$ should be changed to the term $b$: $s(i) = b$. The identity $id$ represents the substitution $\{\underline{1}/\underline{1}, \underline{2}/\underline{2}, \dots\}$, the shift $\uparrow$ represents the substitution $\{\underline{i+1}/\underline{i} \mid \forall \underline{i}\}$ and $\circ$ represents the composition of substitutions. Thus, the shift operator $\uparrow$ in the $\lambda\sigma$ corresponds to the lifting operator $^{+}$ (see [14]). Let the substitution $\uparrow \circ \cdots \circ \uparrow$, which is the composition of $n$ shift operators, be denoted by $\uparrow^n$ and $id$ be denoted by $\uparrow^0$. Then, $\underline{1}[\uparrow^n]$ codifies the de Bruijn index $\underline{n+1}$. The **closure** $a[s]$ represents the application of the substitution $s$ to $a$, which should give the term $a$ replacing all free occurrences of indices $\underline{i}$ in $a$ by $s(i)$; thus, $\underline{i}[s]$ should give $s(i)$. The **cons of $a$ in $s$**, $a.s$, represents the substitution $\{a/\underline{1}, s(i)/\underline{i+1}\}$. The $\beta$-reduction of $(\lambda.a\ b)$ 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 updated (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 $\eta$-reduction, as presented in [7].

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

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

Without (Eta), this system is equivalent to the one given in [1] originally. The associated **substitution calculus**, denoted as $\boldsymbol{\sigma}$, is the one induced by all the rules except (Beta) and (Eta), and its equality is denoted as $=_{\boldsymbol{\sigma}}$.

DEFINITION 2.4 ($\sigma$-normal form)
Given a $\lambda\sigma$-expression $b$, we let $\sigma(b)$ denote its $\sigma$-normal form.

Normalised $\lambda\sigma$-terms with respect to the $\sigma$-calculus, $\sigma$-nf for short, are terms whose closure subterms $a[s]$ are of the form $\underline{1}\,[\uparrow^n]$, where $n \in \mathbb{N}^* = \mathbb{N}\backslash\{0\}$. All other subterms are of the form $\underline{1}$, $\lambda.a$ or $(a\ b)$.

## 2.3  The $\lambda s_e$-Calculus

In contrast to the $\lambda\sigma$-calculus, the $\lambda s_e$-calculus, given in [11], has a sole sort of objects and introduces two operators $\sigma$ and $\varphi$, for substitution and updating.

DEFINITION 2.5 (Set $\Lambda_s$ of $\lambda s_e$-terms)
The syntax of the $\lambda s_e$-calculus, is given by the set $\Lambda_s$ which contains the following $\lambda s_e$-terms where $n, i, j \in \mathbb{N}^*$ and $k \in \mathbb{N}$:

**Terms**    $a ::= \underline{n} \mid (a\ a) \mid \lambda.a \mid a\,\sigma^i a \mid \varphi_k^j a$

Here, $a\,\sigma^i b$ 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 indices in $a$. The term $\varphi_k^j\,a$ represents $j-1$ applications of the $k$-lift to $a$; i.e., $a^{+k^{(j-1)}}$. Table 2 gives the rules of the $\lambda s_e$-calculus augmented with the rule (Eta), as introduced in [3].

TABLE 2. The rewriting system of the $\lambda s_e$-calculus

| | | | |
|---|---|---|---|
| $(\lambda.a\ b)$ | $\longrightarrow$ | $a\,\sigma^1 b$ | ($\sigma$-generation) |
| $(\lambda.a)\,\sigma^i b$ | $\longrightarrow$ | $\lambda.(a\,\sigma^{i+1}b)$ | ($\sigma$-$\lambda$-transition) |
| $(a_1\ a_2)\,\sigma^i b$ | $\longrightarrow$ | $((a_1\,\sigma^i b)\ (a_2\,\sigma^i b))$ | ($\sigma$-app-transition) |
| $\underline{n}\,\sigma^i b$ | $\longrightarrow$ | $\begin{cases} \underline{n-1} & \text{if } n > i \\ \varphi_0^i\,b & \text{if } n = i \\ \underline{n} & \text{if } n < i \end{cases}$ | ($\sigma$-destruction) |
| $\varphi_k^i\,(\lambda.a)$ | $\longrightarrow$ | $\lambda.(\varphi_{k+1}^i\,a)$ | ($\varphi$-$\lambda$-transition) |
| $\varphi_k^i\,(a_1\ a_2)$ | $\longrightarrow$ | $((\varphi_k^i\,a_1)\ (\varphi_k^i\,a_2))$ | ($\varphi$-app-transition) |
| $\varphi_k^i\,\underline{n}$ | $\longrightarrow$ | $\begin{cases} \underline{n+i-1} & \text{if } n > k \\ \underline{n} & \text{if } n \leq k \end{cases}$ | ($\varphi$-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 $i \leq j$ | ($\sigma$-$\sigma$-transition) |
| $(\varphi_k^i\,a)\,\sigma^j b$ | $\longrightarrow$ | $\varphi_k^{i-1}\,a$   if $k < j < k+i$ | ($\sigma$-$\varphi$-transition 1) |
| $(\varphi_k^i\,a)\,\sigma^j b$ | $\longrightarrow$ | $\varphi_k^i\,(a\,\sigma^{j-i+1}b)$   if $k+i \leq j$ | ($\sigma$-$\varphi$-transition 2) |
| $\varphi_k^i\,(a\,\sigma^j b)$ | $\longrightarrow$ | $(\varphi_{k+1}^i\,a)\,\sigma^j(\varphi_{k+1-j}^i\,b)$   if $j \leq k+1$ | ($\varphi$-$\sigma$-transition) |
| $\varphi_k^i\,(\varphi_l^j\,a)$ | $\longrightarrow$ | $\varphi_l^j\,(\varphi_{k+1-j}^i\,a)$   if $l+j \leq k$ | ($\varphi$-$\varphi$-transition 1) |
| $\varphi_k^i\,(\varphi_l^j\,a)$ | $\longrightarrow$ | $\varphi_l^{j+i-1}\,a$   if $l \leq k < l+j$ | ($\varphi$-$\varphi$-transition 2) |
| $\lambda.(a\ \underline{1})$ | $\longrightarrow$ | $b$   if $a =_{s_e} \varphi_0^2\,b$ | (Eta) |

$=_{\boldsymbol{s_e}}$ denotes the equality for the associated **substitution calculus, denoted as $\boldsymbol{s_e}$**, induced by all the rules except ($\sigma$-generation) and (Eta).

DEFINITION 2.6 ($s_e$-normal form)
Given a $\lambda s_e$-term $b$, we let $s_e(b)$ denote its $s_e$-normal form.

## 3   Simple Type Systems

In this paper we work only with simple type systems in the so-called Curry-style or implicit typing, where in terms of the form $\lambda.a$ we do not specify the type of the bound index ($\underline{1}$).

DEFINITION 3.1 (Simple types, Contexts)
The syntax of the **simple types** and **contexts** is given by:

   **Types**   $A ::= K \mid A \to A$        **Contexts**   $\Gamma ::= nil \mid A.\Gamma$

$K$ ranges over **type variables**. A **type assignment system** $S$ is a set of rules which allows some terms of a given system to be associated with a type. A **context** gives the necessary information used by $S$ rules to associate a type to a term. In the simply typed $\lambda$-calculus [9], the typeable terms are strongly normalising. For a term $a$, $\Gamma \vdash a : A$ denotes that $a$ has type $A$ in context $\Gamma$. The contexts for $\lambda$-terms in de Bruijn notation are sequences of types.

NOTATION 3.2 ($\Gamma_{<i}, \Gamma_{>i}$)
Let $\Gamma = A_1. \cdots .A_n.nil$ and $i \in \mathbb{N}$. Then we use $\Gamma_{<i}$ to denote $A_1. \cdots .A_{i-1}$ and we use $\Gamma_{>i}$ to denote $A_{i+1}. \cdots .A_n.nil$. We define $\Gamma_{\leq i}$ and $\Gamma_{\geq i}$ similarly. Note that $\Gamma_{\leq 0}.\Gamma = \Gamma_{<0}.\Gamma = \Gamma$.

Given a type system $S$, the **Subject Reduction Property** states that any computation allowed on terms does not change its type.

### 3.1   Simply Typed $\lambda\sigma$

The typed version of $\lambda\sigma$ is presented in Curry style, instead of the Church style given in [7]. Thus, the syntax of $\lambda\sigma$-terms and the rules are the same as the untyped version.

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, due to its semantics, can be viewed as a list of terms. Consequently, its type is a context. The notation $s \triangleright \Gamma$ denotes that the object $s$ of sort substitution has type $\Gamma$.

DEFINITION 3.3 (The System $TA_{\lambda\sigma}$)
$TA_{\lambda\sigma}$ is given by the following typing rules:

$$(\text{var}) \qquad A.\Gamma \vdash \underline{1} : A \qquad\qquad (\text{lambda}) \qquad \frac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda.b : A \to B}$$

$$(\text{app}) \quad \frac{\Gamma \vdash a : A \to B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\ b) : B} \qquad\qquad (\text{clos}) \quad \frac{\Gamma \vdash s \triangleright \Gamma' \quad \Gamma' \vdash a : A}{\Gamma \vdash a[s] : A}$$

$$(\text{id}) \qquad \Gamma \vdash id \triangleright \Gamma \qquad\qquad (\text{shift}) \qquad A.\Gamma \vdash\ \uparrow\ \triangleright \Gamma$$

$$(\text{cons}) \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash s \triangleright \Gamma'}{\Gamma \vdash a.s \triangleright A.\Gamma'} \qquad\qquad (\text{comp}) \quad \frac{\Gamma \vdash s'' \triangleright \Gamma'' \quad \Gamma'' \vdash s' \triangleright \Gamma'}{\Gamma \vdash s' \circ s'' \triangleright \Gamma'}$$

Observe that the name of the typing rules begin with lower-case letters, while the rewriting rules with upper-case letters. We have verified that this version of $\lambda\sigma$ in Curry style has the same properties as the version of $\lambda\sigma$ in Church style given in [7].

For instance, confluence (CR) and weak normalisation property (WN) hold for this version. Furthermore, $\sigma$ is noetherian and CR.

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

THEOREM 3.4 (SR for $\lambda\sigma$ without the Eta rule [1])
If $\Gamma \vdash_{TA_{\lambda\sigma}} a : A$ and $a \rightarrow_{\lambda\sigma} a'$, then $\Gamma \vdash_{TA_{\lambda\sigma}} a' : A$. Analogously, if $\Gamma \vdash_{TA_{\lambda\sigma}} s \triangleright \Gamma'$ and $s \rightarrow_{\lambda\sigma} s'$, then $\Gamma \vdash_{TA_{\lambda\sigma}} s' \triangleright \Gamma'$.

## 3.2 Simply Typed $\lambda s_e$

DEFINITION 3.5 (The System $TA_{\lambda s_e}$)
$TA_{\lambda s_e}$ is given by the following typing rules.

(Var) $\qquad\qquad A.\Gamma \vdash \underline{1} : A$ $\qquad$ (Varn) $\qquad \dfrac{\Gamma \vdash \underline{n} : B}{A.\Gamma \vdash \underline{n+1} : B}$

(Lambda) $\qquad \dfrac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda.b : A \rightarrow B}$ $\qquad$ (App) $\qquad \dfrac{\Gamma \vdash a : A \rightarrow B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\ b) : B}$

(Sigma) $\qquad \dfrac{\Gamma_{\geq i} \vdash b : B \quad \Gamma_{<i}.B.\Gamma_{\geq i} \vdash a : A}{\Gamma \vdash a\,\sigma^i b : A}$ $\qquad$ (Phi) $\qquad \dfrac{\Gamma_{\leq k}.\Gamma_{\geq k+i} \vdash a : A}{\Gamma \vdash \varphi_k^i\,a : A}$

As for $\lambda\sigma$, the typed version of $\lambda s_e$ is presented in Curry style. We have verified that the Curry style version has CR and WN as does the Church style version of [3].

For the simply typed $\lambda s_e$ without the rule (Eta), SR holds.

THEOREM 3.6 (SR for $\boldsymbol{\lambda s_e}$ without the Eta rule [12])
If $\Gamma \vdash_{T\lambda s_e} a : A$ and $a \rightarrow_{\lambda s_e} a'$, then $\Gamma \vdash_{T\lambda s_e} a' : A$.

## 4 The $\lambda\sigma$-Calculus with a constructive Eta rule

The conditional rule $c\eta$ introduced in [8] has the same form as the rule $(Eta)$ in Table 1. In [8] there is no restriction of $c\eta$ application despite the fact that an attempt to extend the rule to any $\lambda\sigma$-term was explicitly expressed: "The ground $\lambda\sigma$-terms in $\sigma$-nf are exactly the $\lambda$-terms so we may apply the $\eta$-reduction of $\lambda$-calculus to those terms. We want to extend this relation to the whole set of $\lambda\sigma$-terms". The $\lambda\sigma$-calculus defined in [7] has the same rules as presented in Table 1, and inherits the Eta rule as defined in [8]. Observe that in [7] some properties about $\lambda\sigma$, defined as $\{(Beta),(Eta)\} \cup \sigma$, refer to papers where $\lambda\sigma$ is defined without an Eta rule. Indeed, without the restriction that both $a$ and $b$ must be in $\sigma$-nf, this rule $(Eta)$ is not deterministic and SR may be violated as we illustrate below.

Suppose the application of $(Eta)$ in Table 1 is restricted to terms where $a$ is a $\sigma$-nf. If furthermore $b$ is not restricted to be in $\sigma$-nf, then using the $\sigma$-equality in the condition given by the definition of $(Eta)$, one has infinite possibilities of reduction since, for any $c[\uparrow] =_\sigma a$ and any $\lambda\sigma$-substitution $s$, we have $\underline{1}\,[c.s][\uparrow] =_\sigma a$. For instance, $\lambda.(\underline{2}\ \underline{1}) \longrightarrow_{Eta} \underline{1}\,[\underline{1}.(\lambda.(\underline{1}\ \underline{1}).\uparrow)]$, where $\underline{2}$ abbreviates $\underline{1}[\uparrow]$. But the substitution $(\lambda.(\underline{1}\ \underline{1}).\uparrow)$ is not typeable because $\lambda.(\underline{1}\ \underline{1})$ is not typeable in the simply typed $\lambda$-calculus. Thus, without restrictions on both $a$ and $b$ this rule violates not only SR, but even the preservation of typeability.

Furthermore, a typed version of Eta without any restriction on the term $b$ gives non noetherianity. To obtain the type version of Eta we include " If $\Gamma \vdash a : A$ then

$\Gamma \vdash b[\uparrow] : A$" to the side condition for the $(Eta)$ rule in Table 1. Let $\Gamma \vdash \lambda.(a\ \underline{1}) : B$, where $\Gamma$ is not *nil*. If there is $b$, where $a =_\sigma b[\uparrow]$ (take for example $a = \underline{1}[\uparrow]$, $b = \underline{1}$) and $\Gamma \vdash b : B$, then $\Gamma \vdash \underline{2}[(\lambda.(a\ \underline{1})).b.\uparrow] : B$. Consequently, $\lambda.(a\ \underline{1})$ can be reduced by the typed Eta rule to $\underline{2}[(\lambda.(a\ \underline{1})).b.\uparrow]$, allowing an infinite reduction.

In addition, supposing $a$ and $b$ are restricted to be $\sigma$-normal, notice that adding restrictions such as $b$ is well-typed is not enough for guaranteeing that the computations for deciding the Eta condition must preserve SR. In fact, the reduction $\lambda(\underline{2}\ \underline{1}) \longrightarrow_\eta \underline{1}$ can be decided by inferring that $\underline{1}[\uparrow] =_\sigma \underline{2}$ by a $\sigma$-conversion 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 satisfiability of the rule's condition should be decided.

### 4.1   A calculus for explicitly checking the Eta condition in $\lambda\sigma$

The definition of Eta for the $\lambda\sigma$-calculus in Table 1 is inherited from the usual non constructive definition of $\eta$-reduction given in the literature for the $\lambda$-calculus in de Bruijn notation as presented in the motivation of section 2. In fact the condition of the rule Eta suggests searching for some term $b$ such that $b[\uparrow] =_\sigma a$ instead of building $b$ from $a$.

For a constructive definition of $\eta$-reduction in $\lambda\sigma$ some relevant aspects have to be considered. Given a potential $\eta$-redex $\lambda(a\ \underline{1})$ where $a$ is a $\sigma$-nf, then for a dummy symbol $\diamond$, $\sigma(a[\diamond.\uparrow])$ would answer whether the term is an $\eta$-redex marking any free occurrence of $\underline{1}$ in $a$ with $\diamond$. Then one would have

$$\lambda.(a\ \underline{1}) \longrightarrow a[\diamond.id] \quad \text{if} \quad a \text{ is } \sigma-\text{nf and } \sigma(a[\diamond.\uparrow]) \in \Lambda_\sigma \qquad (Eta_1)$$

Since a $\sigma$-normalisation is necessary to verify the condition for $\eta$-reduction, one would also have

$$\lambda.(a\ \underline{1}) \longrightarrow b \quad \text{if} \quad a \text{ is } \sigma-\text{nf and } b \equiv \sigma(a[\diamond.id]) \in \Lambda_\sigma \qquad (Eta_2)$$

In the conditional rule $(Eta_2)$, the $\eta$-contractum is constructed at the same time the condition for reduction is verified, as is done for the pure $\lambda$-calculus in the Bruijn notation [14]. Thus, $\eta$-contraction is done in one step whereas the verification of the condition for reduction and the construction of the $\eta$-contractum are made explicit.

The extension of $(Eta_2)$ to any $\lambda\sigma$-term has to be made carefully. The implementation of such an extension is presented in [2], using the syntax of the $\lambda\sigma$-calculus enlarged with $\diamond$, which belongs to the sort of terms. While constructing $b$, an attempt to avoid reductions not related to the $\eta$-contraction, is made by a restriction on the application of $\sigma$-rules to substitutions with occurrences of $\diamond$. A verification of whether $\diamond$ occurs in the normal form of $a[\diamond.id]$ with respect to this restricted calculus is carried out, to decide whether it is the $\eta$-contractum. However, some rules do not have any restriction on their application, otherwise some "rubbish" would remain after normalisation. For instance, here is an example taken from [2]:
$(\underline{1}[\uparrow^2])[\diamond.id] \longrightarrow_{Clos} \underline{1}[\uparrow^2 \circ (\diamond.id)] \longrightarrow_{AssEnv} \underline{1}[\uparrow \circ (\uparrow \circ (\diamond.id))] \longrightarrow_{ShiftCons} \underline{1}[\uparrow \circ id]$,
where (IdR) is needed giving $\underline{1}[\uparrow \circ id] \longrightarrow_{IdR} \underline{1}[\uparrow]$.

Allowing unrestricted applications of this rule permits reductions of similar redices which are not related to $\eta$-reduction.

TABLE 3. $\eta_{\lambda\sigma}$: the rewriting system for $\eta$-reduction in $\lambda\sigma$

| | | | |
|---|---|---|---|
| $(a\ b)[\eta_j^i]$ | $\longrightarrow$ | $(a[\eta_j^i]\ b[\eta_j^i])$ if $i = j$ | ($\eta$-App) |
| $\underline{1}[\eta_j^i]$ | $\longrightarrow$ | $\begin{cases} \underline{1}[\uparrow^{i-j}] & \text{if } 1 < j < i \\ \underline{1} & \text{if } 1 < j = i \end{cases}$ | ($\eta$-VarCons) |
| $(\lambda.a)[\eta_j^i]$ | $\longrightarrow$ | $\lambda.(a[\eta_{j+1}^{i+1}])$ if $i = j$ | ($\eta$-Abs) |
| $(a[t])[\eta_j^i]$ | $\longrightarrow$ | $a[t\circ\eta_j^i]$ if $i = j$ | ($\eta$-Clos) |
| $a[\eta_0^i]$ | $\longrightarrow$ | $\begin{cases} a & \text{if } i = 1 \\ a[\uparrow^{i-1}] & \text{otherwise} \end{cases}$ | ($\eta$-Id) |
| $(s_1 \circ s_2) \circ \eta_j^i$ | $\longrightarrow$ | $s_1 \circ (s_2 \circ \eta_j^i)$ | ($\eta$-AssEnv) |
| $\uparrow \circ\ \eta_j^i$ | $\longrightarrow$ | $\begin{cases} \uparrow^i & \text{if } j = 0 \\ \eta_{j-1}^i & \text{otherwise} \end{cases}$ | ($\eta$-ShiftCons) |
| $(a.s) \circ \eta_j^i$ | $\longrightarrow$ | $a[\eta_j^i].(s \circ \eta_j^i)$ if $i = j$ | ($\eta$-MapEnv) |

Our approach at expliciting the Eta-contracta building while deciding the condition for Eta-reduction consists in extending the syntax and the rules of the $\lambda\sigma$-calculus. First, the syntax of the $\lambda\sigma$-calculus is enlarged with the symbol $\eta_j^i$, for $i \in \mathbb{N}^*$ and $j \in \mathbb{N}$, which belongs to the sort of substitutions. The symbol $\eta_j^i$ encapsulates the mechanism of the substitution used in [2].

DEFINITION 4.1 (Set $\Lambda_{\sigma\eta}$)
For $i \in \mathbb{N}^*$ and $j \in \mathbb{N}$, $\Lambda_{\sigma\eta}$ is the set generated by

**Terms** $\quad a ::= \underline{1} \,|\, (a\ a) \,|\, \lambda.a \,|\, a[s]$ $\qquad$ **Substitutions** $\quad s ::= id \,|\, \uparrow \,|\, a.s \,|\, s \circ s \,|\, \eta_i^j$

Second, we introduce extra rules that deal with the new symbols $\eta_j^i$ constructively expliciting Eta reduction. In Table 3 we introduce the rewriting system $\eta_{\lambda\sigma}$ used to detect free occurrences of $\underline{1}$. Here, $\eta_{\lambda\sigma}$ is used for detecting free occurrences of $\underline{1}$ in $a$ starting from the term $a[\eta_1^1]$ and reducing it to an $\eta_{\lambda\sigma}$-normal form. The substitution $\eta_1^1$ corresponds to the substitution $\diamond.id$. Then, $\eta_0^i$ corresponds to $\uparrow^{i-1}$, $\eta_1^i$ corresponds to $\diamond.\uparrow^{i-1}$ and $\eta_j^i$ corresponds to $(\underline{n}.\cdots.\underline{i-1}.\diamond.\uparrow^{i-1})$, where $1 < j \leq i$, $n = i - (j-1)$ and $\diamond$ is the $j^{\text{th}}$ term. Analogously to $\sigma$, the rewriting rules of $\eta_{\lambda\sigma}$ propagate the substitution $\eta_j^i$ into the structure of $\lambda\sigma$-terms. Verification of rule applicability is done by arithmetic constraints on $\eta$ subscripts and superscripts. For instance, the former reduction becomes $(\underline{1}[\uparrow^2])[\eta_1^1] \longrightarrow_{\eta\text{-Clos}} \underline{1}[\uparrow^2 \circ \eta_1^1] \longrightarrow_{\eta\text{-AssEnv}} \underline{1}[\uparrow\circ(\uparrow\circ\eta_1^1)] \longrightarrow_{\eta\text{-ShiftCons}} \underline{1}[\uparrow\circ\eta_0^1] \longrightarrow_{\eta\text{-Id}} \underline{1}[\uparrow]$.

Our aim is to generalise rule ($Eta_2$), cutting down the restriction of applicability only to terms in $\sigma$-nf. In order to do that, we propagate the substitution $\eta_i^i$ into the term structure until it finds a closure $a[t]$. Then, the detection of free occurrences of $\underline{i}$ is done on substitution $t$. Since by the semantics of $\lambda\sigma$-substitutions $s$ can be viewed as a list of terms, $s$ will be said to have a free occurrence of $\underline{i}$ if any of these terms has a free occurrence of $\underline{i}$. In other words, the construction of the $\eta$-contractum is aborted if $\sigma(a[t])$ may have a free occurrence of $\underline{i}$. This approach avoids the analysis of whether the $\sigma$-nf would actually have any free occurrence of $\underline{i}$, which could demand the same amount of work in computation steps as the $\sigma$-normalisation itself.

EXAMPLE 4.2

Let $a = \underline{3}\,[id] = (\underline{1}[\uparrow^2])[id]$. Then, for $a[\eta_1^1]$ by ($\eta$-Clos), one has

$$(\underline{3}\,[id])[\eta_1^1] \to \underline{3}\,[id \circ \eta_1^1]$$

which is an $\eta_{\lambda\sigma}$-normal form.

Observe that when deciding whether $\lambda.(\underline{3}\,[id]\ \underline{1})$ is an $\eta$-redex, stopping the propagation of the symbol $\eta_1^1$ is correct. This is the case since the closure $\underline{3}[id]$ has in fact occurrences of $\underline{1}$: $id$ which correspond to the substitution $\underline{1}.\underline{2}.\underline{3}\ldots$. Reaching an $\eta_{\lambda\sigma}$-normal form of $\underline{3}\,[id \circ \eta_1^1]$ that is not a pure $\lambda\sigma$-term is the way to detect these occurrences. Otherwise, the process cannot be considered *clean* (as explained in [2]) because in principle, rules not involved directly in the $\eta$-contraction are applied.

In order to define a new (*Eta*) rule using $\eta_{\lambda\sigma}$ to verify the condition for reduction, we need to prove some basic properties for this rewriting system.

Observe that $\eta_{\lambda\sigma}$ is not confluent over $\Lambda_{\sigma\eta}$. For $(\underline{1}\,[\eta_2^2])\eta_1^1$ one has

$$\underline{1}\,[\eta_1^1] \leftarrow (\underline{1}\,[\eta_2^2])[\eta_1^1] \to \underline{1}\,[\eta_2^2 \circ \eta_1^1]$$

and both $\underline{1}\,[\eta_1^1]$ and $\underline{1}\,[\eta_2^2 \circ \eta_1^1]$ are $\eta_{\lambda\sigma}$-nf. Thus, we need to prove that there are no such subterms during the verification of free occurrence and we need to choose a proper subset of $\Lambda_{\sigma\eta}$ where $\eta_{\lambda\sigma}$ is confluent.

LEMMA 4.3

Let $a[\eta_i^i] \to_{\eta_{\lambda\sigma}}^+ c$, where $a$ is a $\lambda\sigma$-term. For any subexpression $c'$ of $c$, if $c' = b[\eta_j^{i'}]$, then $b$ is a $\lambda\sigma$-term and if $c' = s \circ \eta_j^{i'}$, then $s$ is a $\lambda\sigma$-substitution.

*Proof.*    By induction on the number $n$ of reduction steps. For $n = 1$, a straightforward analysis of the rules gives the result. Suppose the statement is true for some $n$. Let $a[\eta_i^i] \to^{n+1} c$ and $b$ be the expression such that $a \to^n b$ and $b \to c$. By the induction hypothesis (IH), one has that for any subexpression $b'$ of $b$, if $b' = d[\eta_j^{i'}]$, then $d$ is a $\lambda\sigma$-term and if $b' = s \circ \eta_j^{i'}$, then $s$ is a $\lambda\sigma$-substitution. Then, we have to analyse the last rule applied on $b$:

- If $b \to c$ by ($\eta$-App), one has that one occurrence of $(b_1\ b_2)[\eta_i^i]$ in $b$ is replaced by $(b_1[\eta_i^i]\ b_2[\eta_i^i])$, where $b_1$ and $b_2$ are $\lambda\sigma$-terms. Thus, the result holds for $c$. A similar argument can be used if the last rule applied is ($\eta$-Abs), ($\eta$-Clos), ($\eta$-AssEnv) or ($\eta$-MapEnv).

- If $b \to c$ by either ($\eta$-VarCons) or ($\eta$-Id), then one occurrence of $\eta_j^i$ in $b$ is destroyed. Thus, the result holds for $c$.

- Let $b \to c$ by ($\eta$-ShiftCons). If the redex in $b$ is of the form $\uparrow \circ \eta_0^{i'}$, then the occurrence of $\eta_0^{i'}$ is destroyed. Otherwise, $\uparrow \circ \eta_j^{i'}$ is replaced by $\eta_{j-1}^{i'}$. If $u \circ \eta_{j-1}^{i'}$ is a subexpression of $c$, then $u$ is a $\lambda\sigma$-substitution, otherwise we would have $a[\eta_i^i] \to^i d$, for $i < n$, where $(u \circ \uparrow) \circ \eta_j^{i'}$ is a subexpression of $d$ and $(u \circ \uparrow)$ is not a $\lambda\sigma$-substitution. If $d[\eta_{j-1}^{i'}]$ is a subexpression of $c$, then $d$ is a $\lambda\sigma$-term, otherwise we would have $a[\eta_i^i] \to^i e$, for $i < n$, where either $(d[\uparrow])[\eta_j^{i'}]$ or $(d[\uparrow \circ u])[\eta_{j'}^{i'}]$, for $j' > j$, is a subexpression of $e$ and neither $d[\uparrow]$ nor $d[\uparrow \circ u]$ is a $\lambda\sigma$-term.    □

Since we observed that $\eta_{\lambda\sigma}$ is not confluent over $\Lambda_{\sigma\eta}$, in what follows, we will introduce the sub-language $\Im$ of terms of $\Lambda_{\sigma\eta}$ over which $\eta_{\lambda\sigma}$ is confluent. The sub-language $\Im$ will be used to decide $\eta$-redices.

DEFINITION 4.4 (Set $\mathfrak{I}$)
Let $\mathfrak{I} \subset \Lambda_{\sigma\eta}$ be the set of expressions (subterms and substitutions) obtained as the closure, under the rules of $\eta_{\lambda\sigma}$, of the set of all $a[\eta_i^i]$, where $a$ is a $\lambda\sigma$-term. In particular, an **$\eta_{\lambda\sigma}$-normal form** of $c \in \mathfrak{I}$ is denoted as **$\eta_{\lambda\sigma}(c)$**.

In order to show that $\eta_{\lambda\sigma}$ effectively decides and contracts $\eta$-redices, we need to show its confluence and termination on $\mathfrak{I}$. The next lemma establishes confluence of $\eta_{\lambda\sigma}$ on $\mathfrak{I}$.

LEMMA 4.5
$\eta_{\lambda\sigma}$ is confluent on $\mathfrak{I}$.

*Proof.* Note that $\eta_{\lambda\sigma}$ is left-linear. The critical pairs for $\eta_{\lambda\sigma}$ arise when one has redices of the form $(a[\eta_j^i])[\eta_{i'}^{i'}]$ or $(s \circ \eta_j^i) \circ \eta_{j'}^{i'}$. However, by Lemma 4.3, these redices do not occur in any $c \in \mathfrak{I}$. Hence, $\eta_{\lambda\sigma}$ is non-overlapping for $\mathfrak{I}$. Thus, by orthogonality, $\eta_{\lambda\sigma}$ is confluent. $\square$

The proof of $\eta_{\lambda\sigma}$ termination on $\mathfrak{I}$ needs the following definition and lemmas.

DEFINITION 4.6
Let $a$ be a $\lambda\sigma$-expression. Define $\| \cdot \| : \Lambda_\sigma \to \mathbb{N}$ by:

$$
\begin{aligned}
\|(a\ b)\| &= \|a\| + \|b\| & \|\underline{1}\| &= 0 \\
\|\lambda.a\| &= \|a\| & \|id\| &= 0 \\
\|a[s]\| &= \|a\| + \|s\| & \|\uparrow\| &= 0 \\
\|s \circ t\| &= \|s\| + \|t\| & \|a.t\| &= 1 + \|a\| + \|t\|
\end{aligned}
$$

LEMMA 4.7
Let $j \leq i$, where $i \in \mathbb{N}^*$ and $j \in \mathbb{N}$, and let $s$ be a $\lambda\sigma$-substitution such that $\|s\| = 0$. Then, $(s \circ \eta_j^i) \to_{\eta_{\lambda\sigma}}^* s'$, where $s'$ is a normal form w.r.t. $\eta_{\lambda\sigma}$ and is either a $\lambda\sigma$-substitution or a substitution with occurrence of $\eta_{j'}^i$, where $j' \leq j$. Particularly, if $s' = \eta_{j'}^i$, then $j' < j$.

*Proof.* By induction on the structure of $s$.

1) $s = id$: $(id \circ \eta_j^i)$ is a normal form w.r.t. $\eta_{\lambda\sigma}$.
2) $s = \uparrow$: By ($\eta$-ShiftCons) one has that $(\uparrow \circ \eta_j^i)$ reduces to $\uparrow^i$ or to $\eta_{j-1}^i$.
3) $s = u \circ t$: By ($\eta$-AssEnv) one has that $(u \circ t) \circ \eta_j^i \to u \circ (t \circ \eta_j^i)$. By IH, $(t \circ \eta_j^i) \to^* t'$, where $t'$ is a $\lambda\sigma$-substitution or a substitution with an occurrence of $\eta_{j'}^i$, for $j' \leq j$. If $t' = \eta_{j'}^i$, then one has the result by IH on $u \circ t'$. Otherwise, $u \circ t'$ is a normal form w.r.t. $\eta_{\lambda\sigma}$. $\square$

LEMMA 4.8
Let $i \in \mathbb{N}^*$, and $a$ be a $\lambda\sigma$-term such that $\|a\| = 0$. Then, $a[\eta_i^i] \to_{\eta_{\lambda\sigma}}^* a'$, where $a'$ is a normal form w.r.t. $\eta_{\lambda\sigma}$.

*Proof.* By induction on the structure of $a$. Normal forms are w.r.t. $\eta_{\lambda\sigma}$.

1) $a = \underline{1}$: If $i = 1$, then $\underline{1}[\eta_1^1]$ is a normal form. Otherwise, by ($\eta$-VarCons), $\underline{1}[\eta_i^i] \to \underline{1}$.
2) $a = (b\ c)$: By ($\eta$-App), $(b\ c)[\eta_i^i] \to (b[\eta_i^i]\ c[\eta_i^i])$. By IH, the result holds for $b[\eta_i^i]$ and $c[\eta_i^i]$.

3) $a = \lambda.b$: By ($\eta$-Abs), $(\lambda.b)[\eta_i^i] \to \lambda.(b[\eta_{i+1}^{i+1}])$. By IH, the result holds for $b[\eta_{i+1}^{i+1}]$.

4) $a = b[s]$: By ($\eta$-Clos), $(b[s])[\eta_i^i] \to b[s \circ \eta_i^i]$. One has $\|b[s]\| = \|b\| + \|s\| = 0$. Then, by Lemma 4.7, $(s \circ \eta_i^i) \to^* s'$ where $s'$ is a normal form. If $s' = \eta_{i'}^i$ and $i' \neq 1$ then, by Lemma 4.7, $i' < i$, so only the rules ($\eta$-VarCons) and ($\eta$-Id) can be applied to $a[s']$, giving a normal form. Otherwise, $a[s']$ is a normal form. □

LEMMA 4.9

Let $i \in \mathbb{N}^*$, and let $a$ be a $\lambda\sigma$-term. Then, $\eta_{\lambda\sigma}$ is terminating for $a[\eta_i^i]$.

*Proof.* By induction on the structure of $a$ with subinduction on $\|\cdot\|$, having Lemmas 4.7 and 4.8 as induction base (IB). Normal forms are w.r.t. $\eta_{\lambda\sigma}$.

1) $a = \underline{1}$: Since $\|\underline{1}\| = 0$, by Lemma 4.8 the result holds.

2) $a = (b\ c)$: By ($\eta$-App), $(b\ c)[\eta_i^i] \to (b[\eta_i^i]\ c[\eta_i^i])$. By IH on the structure, the result holds for $b[\eta_i^i]$ and $c[\eta_i^i]$.

3) $a = \lambda.b$: By ($\eta$-Abs), $(\lambda.b)[\eta_i^i] \to \lambda.(b[\eta_{i+1}^{i+1}])$. By IH on the structure, the result holds for $b[\eta_{i+1}^{i+1}]$.

4) $a = b[s]$: By ($\eta$-Clos), $(b[s])[\eta_i^i] \to b[s \circ \eta_i^i]$. If $\|b[s]\| = 0$, then by Lemma 4.8 the result holds. Otherwise:
   - If $\|s\| = 0$, then Lemma 4.7 can be applied. Then $b[s \circ \eta_i^i] \to^* b[s']$, where $s'$ is a normal form
   - Otherwise, $s = c.t$ or $s = u \circ t$. If $s = c.t$, then by ($\eta$-MapEnv) one has that $b[(c.t) \circ \eta_i^i] \to b[c[\eta_i^i].(t \circ \eta_i^i)]$. As $\|c\|, \|t\| < \|s\|$, by IH on $\|\cdot\|$ one has $c[\eta_i^i] \to^* c'$ and $(t \circ \eta_i^i) \to^* t'$, where $c'$ and $t'$ are normal forms. Observe that $b[c'.t']$ is also a normal form. If $s = u \circ t$, then by ($\eta$-AssEnv) one has that $b[(u \circ t) \circ \eta_i^i] \to b[u \circ (t \circ \eta_i^i)]$. If $\|u\|, \|t\| > 0$, the result holds by IH on $\|\cdot\|$. Otherwise, at least one of the substitutions has $\|\cdot\|$ greater than 0. Using induction on the structure of the substitution $s$, where $\|s\| > 0$, the result holds. Then, one has that $b[u \circ (t \circ \eta_i^i)] \to^* b[s']$, where $s'$ is a normal form as described in Lemma 4.7

   Thus, for $\|b[s]\| > 0$, either $b[s']$ is a normal form or the only applicable rules are ($\eta$-VarCons) and ($\eta$-Id), which gives a normal form. □

Finally, here is the termination of $\eta_{\lambda\sigma}$ on $\mathfrak{I}$.

LEMMA 4.10

$\eta_{\lambda\sigma}$ is terminating for $\mathfrak{I}$.

*Proof.* By Lemma 4.9 the result holds. Otherwise, one would have a subexpression of $b$, where $a[\eta_i^i] \to^* b$ for some $\lambda\sigma$-term $a$ and some $i \in \mathbb{N}^*$, with a non terminating reduction. □

Using $\eta_{\lambda\sigma}$ we can define the eta rule (Eta) as follows.

DEFINITION 4.11 (Constructive (Eta) for $\boldsymbol{\lambda\sigma}$)

Let $a$ be a $\lambda\sigma$-term. The rule **(Eta)** is given by

$$\lambda.(a\ \underline{1}) \to \eta_{\lambda\sigma}(a[\eta_1^1]), \qquad \text{if } \eta_{\lambda\sigma}(a[\eta_1^1]) \text{ is a } \lambda\sigma\text{-term.} \qquad \text{(Eta)}$$

With this definition, $a =_\sigma b[\uparrow]$ where $b = \eta_{\lambda\sigma}(a[\eta_1^1])$. Thus, "$b$" is obtained from $a$ in a deterministic way. The next lemma establishes the correctness of the side condition of the above definition of Eta.

LEMMA 4.12

Let $i \in \mathbb{N}^*$ and $a$ be a $\lambda\sigma$-term in $\sigma$-normal form. Then the term $\eta_{\lambda\sigma}(a[\eta_i^i])$ is a $\lambda\sigma$-term, whenever $a$ has no free occurrences of $\underline{i}$. In this case, one has that $\eta_{\lambda\sigma}(a[\eta_i^i])$ corresponds to $a$, with all the free indices greater than $i$ decremented by one. Otherwise, $\eta_{\lambda\sigma}(a[\eta_i^i]) \notin \Lambda_\sigma$.

*Proof.*   The proof is by induction on the structure of $a$. Normal forms are w.r.t. $\eta_{\lambda\sigma}$.

- $\boldsymbol{a = \underline{1}}$: if $i = 1$, then $\underline{1}[\eta_1^1]$ is a normal form. If $i > 1$, then by ($\eta$-VarCons) one obtains $\underline{1}[\eta_i^i] \to \underline{1}$.
- $\boldsymbol{a = \underline{1}[\uparrow^n]}$: by ($\eta$-Clos), $(\underline{1}[\uparrow^n])[\eta_i^i] \to \underline{1}[\uparrow^n \circ \eta_i^i]$.
  - If $n < i - 1$, then by ($\eta$-AssEnv) and ($\eta$-ShiftCons) one obtains, for $i - n > 1$, $\underline{1}[\uparrow^n \circ \eta_i^i] \to^{2n} \underline{1}[\eta_{i-n}^i]$. So, by ($\eta$-VarCons), $\underline{1}[\eta_{i-n}^i] \to \underline{1}[\uparrow^{i-(i-n)}] = \underline{1}[\uparrow^n]$.
  - If $n = i - 1$, then $i > 1$. By ($\eta$-AssEnv) and ($\eta$-ShiftCons), $\underline{1}[\uparrow^n \circ \eta_i^i] \to^{2n} \underline{1}[\eta_1^i]$.
  - If $n = i$, then by ($\eta$-AssEnv) and ($\eta$-ShiftCons), $\underline{1}[\uparrow^n \circ \eta_i^i] \to^{2i} \underline{1}[\eta_0^i]$. Then, by ($\eta$-Id), if $i = 1$ one has $\underline{1}[\eta_0^1] \to \underline{1}$, otherwise one has $\underline{1}[\eta_0^i] \to \underline{1}[\uparrow^{i-1}]$.
  - If $n > i$, then by ($\eta$-AssEnv) and ($\eta$-ShiftCons), $\underline{1}[\uparrow^n \circ \eta_i^i] \to^{2i} \underline{1}[\uparrow^{n-i} \circ \eta_0^i]$. Then, by ($\eta$-AssEnv) and ($\eta$-ShiftCons), $\underline{1}[\uparrow^{n-i} \circ \eta_0^i] \to^2 \underline{1}[\uparrow^{n-i-1} \circ \uparrow^i]$.
- $\boldsymbol{a = (b\ c)}$: by ($\eta$-App) one has $(b\ c)[\eta_i^i] \to (b[\eta_i^i]\ c[\eta_i^i])$. By the induction hypothesis (IH), if either $b$ or $c$ have free occurrences of $\underline{i}$, then either $\eta_{\lambda\sigma}(b[\eta_i^i]) \notin \Lambda_\sigma$ or $\eta_{\lambda\sigma}(c[\eta_i^i]) \notin \Lambda_\sigma$. Otherwise, $\eta_{\lambda\sigma}(b[\eta_i^i])$ and $\eta_{\lambda\sigma}(c[\eta_i^i])$ are $\lambda\sigma$-terms of the desired form.
- $\boldsymbol{a = \lambda.b}$: by ($\eta$-Abs), $(\lambda.b)[\eta_i^i] \to \lambda.(b[\eta_{i+1}^{i+1}])$. By IH, $\eta_{\lambda\sigma}(b[\eta_{i+1}^{i+1}]) \notin \Lambda_\sigma$ if there are free occurrences of $\underline{i+1}$, else, $\eta_{\lambda\sigma}(b[\eta_{i+1}^{i+1}])$ is the desired $\lambda\sigma$-term.   □

Taking $i = 1$ in the previous lemma, one has that for $\sigma$-normalised $\lambda\sigma$-terms, the rule (Eta) correctly implements $\eta$-reduction.

Observe that, if a $\sigma$-normal form term $a$ has a free occurrence of $\underline{i}$, then $\eta_{\lambda\sigma}(a[\eta_i^i])$ has an occurrence of $\eta_j^i$, where $j \leq i$. Then, we can extend the definition of free index to $\lambda\sigma$-substitutions $s$ by: if $\eta_{\lambda\sigma}(s \circ \eta_i^i)$ has an occurrence of $\eta_j^i$, where $j \leq i$, then $s$ is said to have a free occurrence of $\underline{i}$. Using this extension, we can present the following formal definition of free occurrences.

DEFINITION 4.13

Let $a$ be a $\lambda\sigma$-term. If $\eta_{\lambda\sigma}(a[\eta_i^i])$ is not a $\lambda\sigma$-term, we say that $a$ **has a free occurrence of** $\underline{i}$.

## 4.2   The constructive Eta rule for $\lambda\sigma$ preserves SR

Notice that the rules of the system $\eta_{\lambda\sigma}$ are similar to those of the $\sigma$-calculus, with a substitution $\eta_j^i$ having a particular semantics. To verify if $\eta_{\lambda\sigma}$ has the subject reduction property we have to give a type inference rule for $\eta_j^i$ which is related to its semantics.

DEFINITION 4.14 (Typing judgement *(eta)* for $\boldsymbol{\eta_j^i}$)

Let $j \leq i$, where $j \in \mathbb{N}$ and $i \in \mathbb{N}^*$. Then, given a context $\Gamma$, one has that

$$(eta) \qquad \Gamma_{<i}.\Gamma_{>i} \vdash \eta_j^i \triangleright \Gamma_{>(i-j)}$$

Note that for $i = j$ we have $\Gamma_{>(i-j)} = \Gamma_{>0} = \Gamma$. From this definition, one has a rule for inferring the type of $\eta_j^i$, that depends on the given context and has a semantics which is related to this context only.

LEMMA 4.15
Let $a$ be a $\lambda\sigma$-term, such that $\Gamma \vdash a : A$. If $a$ has no free occurrences of $\underline{i}$ and $a[\eta_i^i] \to_{\eta\lambda\sigma}^* a'$, then $\Gamma_{<i}.\Gamma_{>i} \vdash a' : A$. Particularly, $\Gamma_{<i}.\Gamma_{>i} \vdash \eta_{\lambda\sigma}(a[\eta_i^i]) : A$.

*Proof.*   By inspection of the system rules, one by one, we have the following:

- **($\eta$-App)**: Let $\Gamma \vdash (b\ c)[\eta_i^i] : A$. By $(clos)$ one has that $\Gamma \vdash \eta_i^i \triangleright \Gamma'$ and $\Gamma' \vdash (b\ c) : A$. By $(app)$ one has that $\Gamma' \vdash b : B \to A$ and $\Gamma' \vdash c : B$. By $(clos)$ one has $\Gamma \vdash b[\eta_i^i] : B \to A$ and $\Gamma \vdash c[\eta_i^i] : B$. Thus, by $(app)$, $\Gamma \vdash (b[\eta_i^i]\ c[\eta_i^i]) : A$.

- **($\eta$-VarCons)**: Let $\Gamma \vdash \underline{1}[\eta_j^i] : A$. By $(clos)$ one has that $\Gamma \vdash \eta_j^i \triangleright \Gamma'$ and $\Gamma' \vdash \underline{1} : A$. By $(eta)$ one has $\Gamma' = (\Gamma_{<i}.D.\Gamma_{\geq i})_{>(i-j)}$. There are two cases:
  - If $1 < i = j$, then $\Gamma' = \Gamma_{<i}.D.\Gamma_{\geq i}$. By $(var)$ one has that $\Gamma_{<i} = A.\Gamma''$. Thus, by $(var)$, $\Gamma \vdash \underline{1} : A$.
  - If $1 < j < i$, let $\Gamma = A_1.A_2.\cdots.A_{i-1}.\Gamma_{\geq i}$. Then $\Gamma' = A_{(i-j)+1}.\cdots.A_{i-1}.D.\Gamma_{\geq i}$, where $(i-j)+1 \leq i-1$. By $(var)$ one has that $A_{(i-j)+1} = A$. By $(shift)$ and $(comp)$ one has $\Gamma \vdash \uparrow^{(i-j)} \triangleright A_{(i-j)+1}.\cdots.A_{i-1}.\Gamma_{\geq i}$. By $(var)$ one has that $A_{(i-j)+1}.\cdots.A_{i-1}.\Gamma_{\geq i} \vdash \underline{1} : A_{(i-j)+1} = A$. Thus, by $(clos)$, $\Gamma \vdash \underline{1}[\uparrow^{(i-j)}] : A$.
  **($\eta$-Abs)**: Let $\Gamma \vdash (\lambda.b)[\eta_i^i] : A$. By $(clos)$ one has $\Gamma \vdash \eta_i^i \triangleright \Gamma'$ and $\Gamma' \vdash \lambda.b : A$. By $(eta)$ one has $\Gamma' = \Gamma_{<i}.D.\Gamma_{\geq i}$ for some type $D$. By $(lambda)$ one has $C.\Gamma_{<i}.D.\Gamma_{\geq i} \vdash b : B$, where $A = C \to B$. By $(eta)$ one has that $C.\Gamma \vdash \eta_{i+1}^{i+1} \triangleright C.\Gamma_{<i}.D.\Gamma_{\geq i}$ and, by $(clos)$, $C.\Gamma \vdash b[\eta_{i+1}^{i+1}] : B$. Thus, by $(lambda)$, $\Gamma \vdash \lambda.(b[\eta_{i+1}^{i+1}]) : A$.

- **($\eta$-Clos)**: Let $\Gamma \vdash (b[s])[\eta_i^i] : A$. By $(clos)$ one has that $\Gamma \vdash \eta_i^i \triangleright \Gamma'$ and $\Gamma' \vdash b[s] : A$. By $(clos)$ one has $\Gamma' \vdash s \triangleright \Gamma''$ and $\Gamma'' \vdash b : A$. By $(comp)$ one has that $\Gamma \vdash s \circ \eta_i^i \triangleright \Gamma''$. Thus, by $(clos)$, $\Gamma \vdash b[s \circ \eta_i^i] : A$.

- **($\eta$-Id)**: Let $\Gamma \vdash a[\eta_0^i] : A$. By $(clos)$ one has that $\Gamma \vdash \eta_0^i \triangleright \Gamma'$ and $\Gamma' \vdash a : A$. By $(eta)$, $\Gamma' = (\Gamma_{<i}.D.\Gamma_{\geq i})_{>i} = \Gamma_{\geq i}$. There are two cases:
  - If $i = 1$, then $\Gamma_{\geq 1} = \Gamma$. Thus $\Gamma \vdash a : A$.
  - If $i > 1$, then by $(shift)$ and $(comp)$ one has that $\Gamma \vdash \uparrow^{i-1} \triangleright \Gamma_{\geq i}$. Thus, by $(clos)$, $\Gamma \vdash a[\uparrow^{i-1}] : A$.
  **($\eta$-AssEnv)**: Let $\Gamma \vdash (u \circ v) \circ \eta_j^i \triangleright \Gamma'$. By $(comp)$ one has that $\Gamma \vdash \eta_j^i \triangleright \Gamma''$ and $\Gamma'' \vdash u \circ v \triangleright \Gamma'$. By $(comp)$ one has $\Gamma'' \vdash v \triangleright \Gamma'''$ and $\Gamma''' \vdash u \triangleright \Gamma'$. Then, by $(comp)$, one has that $\Gamma \vdash v \circ \eta_j^i \triangleright \Gamma'''$. Thus, by $(comp)$, $\Gamma \vdash u \circ (v \circ \eta_j^i) \triangleright \Gamma'$.

- **($\eta$-ShiftCons)**: Let $\Gamma \vdash \uparrow \circ \eta_j^i \triangleright \Gamma'$. By $(comp)$ one has that $\Gamma \vdash \eta_j^i \triangleright \Gamma''$ and $\Gamma'' \vdash \uparrow \triangleright \Gamma'$. By $(shift)$ one has $\Gamma'' = C.\Gamma'$ and, by $(eta)$, $\Gamma'' = (\Gamma_{<i}.D.\Gamma_{\geq i})_{>(i-j)}$.
  - If $j = 0$, then $C.\Gamma' = \Gamma_{\geq i}$. Thus, by $(shift)$ and $(comp)$, $\Gamma \vdash \uparrow^i \triangleright \Gamma_{>i} = \Gamma'$.
  - Otherwise, $(\Gamma_{<i}.D.\Gamma_{\geq i})_{>i-(j-1)} = ((\Gamma_{<i}.D.\Gamma_{\geq i})_{>(i-j)})_{>1} = (C.\Gamma')_{>1} = \Gamma'$. Thus, by $(eta)$, $\Gamma \vdash \eta_{j-1}^i \triangleright \Gamma'$.
  **($\eta$-MapEnv)**: Let $\Gamma \vdash (b.t) \circ \eta_i^i \triangleright \Gamma'$. By $(comp)$ one has that $\Gamma \vdash \eta_i^i \triangleright \Gamma''$ and $\Gamma'' \vdash b.t \triangleright \Gamma'$. By $(cons)$ one has $\Gamma'' \vdash b : B$ and $\Gamma'' \vdash t \triangleright \Gamma'''$, where $\Gamma' = B.\Gamma'''$. By $(clos)$ one has that $\Gamma \vdash b[\eta_i^i] : B$ and, by $(comp)$, $\Gamma \vdash t \circ \eta_i^i \triangleright \Gamma'''$. Thus, by $(cons)$, $\Gamma \vdash b[\eta_i^i].(t \circ \eta_i^i) \triangleright \Gamma'$.   $\square$

THEOREM 4.16 (SR for the constructive (Eta) in $\lambda\sigma$)
If $\Gamma \vdash \lambda.(a\ \underline{1}) : A$ and $\lambda.(a\ \underline{1}) \to_{Eta} b$, then $\Gamma \vdash b : A$.

*Proof.*     Suppose that $\Gamma \vdash \lambda.(a\,\underline{1}) : A$. 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 $b = \eta_{\lambda\sigma}(a[\eta_1^1])$, where $a$ has no free occurrence of $\underline{1}$. Thus, by Lemma 4.15 with $i = 1$, $\Gamma \vdash b : A$.                                              $\square$

## 5   The $\lambda s_e$-Calculus with a constructive Eta rule

Recall that in Table 2 we gave the definition of (Eta) for $\lambda s_e$ inherited from the usual definition based on lifting as presented in the motivation of section 2. Similarly to the $\lambda\sigma$-calculus, when no restrictions are given on terms, derivations of ill-typed terms may happen. For instance, let $\Gamma \vdash \underline{i} : B$ and $\Gamma \vdash \underline{n} : A$, where $n < i$. Assume $\Gamma_{\geq i} \vdash N : C$, where $C \neq A$. By ($\sigma$-destruction), $\underline{n}\,\sigma^i N \to \underline{n}$. Note that $\underline{n}\,\sigma^i N$ is not typeable. Thus, $\lambda(\underline{n+1}\,\underline{1}) \to \underline{n}\sigma^i N$, because $\varphi_0^2(\underline{n}\sigma^i N) \to \varphi_0^2(\underline{n}) \to \underline{n+1}$ $(\underline{n+1} =_{s_e} \varphi_0^2(\underline{n}\sigma^i N))$.

### 5.1   A calculus for explicitly checking the Eta condition in $\lambda s_e$

In order to give a constructive and explicit definition of Eta, one should define the free occurrences of $\underline{i}$ in a $\lambda s_e$-term. The following calculus detects whether there are free occurrences of a specific index in a term.

DEFINITION 5.1
Let $m, n \in \mathbb{N}^*$. The **Calculus of detection of Occurrences of Free Indices** in $\lambda s_e$, denoted as COFI, is given by the following rules, where $\vee$ denotes the classic disjunction.

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

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

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

With this definition, the free occurrences of indices in $\lambda s_e$ can be formalised.

DEFINITION 5.2 (Free indices in $\boldsymbol{\lambda s_e}$)
If $\langle a, i \rangle \vdash_{COFI} True$, we say $a$ **has a free occurrence of $\underline{i}$**.

Observe that Definition 5.2 formalises the notion of free occurrence of indices in $\lambda$-terms in de Bruijn notation and extends it to any $\lambda s_e$-term. The following lemma states the relation between free occurrences in $\lambda s_e$-terms and in the corresponding terms in $s_e$-nf.

LEMMA 5.3 (Free occurrences in $\boldsymbol{s_e}$-nf)
Let $a$ and $b$ be $\lambda s_e$-terms, in $s_e$-normal form. The following hold:

1. For $i \leq k$, $s_e(\varphi_k^j a)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i}$.
2. For $k < i < k+j$, $s_e(\varphi_k^j a)$ has no free occurrences of $\underline{i}$.

3. For $i \geq k + j$, $s_e(\varphi_k^j a)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i - j + 1}$.

4. For $i < j$, $s_e(a\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i}$.

5. For $i \geq j$, $s_e(a\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff 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 $\underline{i - j + 1}$).

*Proof.*    All the proofs are by induction on the structure of $a$.

1. • $a = \underline{n}$: by ($\varphi$-destruction), $\varphi_k^j \underline{n} \to \underline{n}$. Thus, $s_e(\varphi_k^j \underline{n})$ has a free occurrence of $\underline{i}$ iff $n = i$.

   • $a = (c\ d)$: by ($\varphi$-app-transition), $\varphi_k^j(c\ d) \to (\varphi_k^j c\ \varphi_k^j d)$. By IH, $s_e(\varphi_k^j c)$ has a free occurrence of $\underline{i}$ iff $c$ has a free occurrence of $\underline{i}$. Analogously for $s_e(\varphi_k^j d)$ and $d$. Note that $s_e(\varphi_k^j a) = (s_e(\varphi_k^j c)\ s_e(\varphi_k^j d))$. Thus, $s_e(\varphi_k^j a)$ has a free occurrence of $\underline{i}$ iff either $c$ or $d$ has a free occurrence of $\underline{i}$.

   • $a = \lambda.c$: by ($\varphi$-$\lambda$-transition), $\varphi_k^j(\lambda.c) \to \lambda.(\varphi_{k+1}^j c)$. Since $i + 1 \leq k + 1$, by IH, $s_e(\varphi_{k+1}^j c)$ has a free occurrence of $\underline{i + 1}$ iff $c$ has a free occurrence of $\underline{i + 1}$. Hence, $s_e(\varphi_{k+1}^j a)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i}$.

2. • $a = \underline{n}$: If $n \leq k$, then by ($\varphi$-destruction) one has $\varphi_k^j \underline{n} \to \underline{n}$, where $n \leq k < i$. If $n > k$, then by ($\varphi$-destruction) one has $\varphi_k^j \underline{n} \to \underline{n + j - 1}$, where $n + j - 1 > k + j - 1 \geq i$. Thus $n + j - 1 > i$.

   • $a = (c\ d)$: by ($\varphi$-app-transition), $\varphi_k^j(c\ d) \to (\varphi_k^j c\ \varphi_k^j d)$. Thus, by IH, neither $s_e(\varphi_k^j c)$ nor $s_e(\varphi_k^j d)$ has a free occurrence of $\underline{i}$.

   • $a = \lambda.c$: by ($\varphi$-$\lambda$-transition), $\varphi_k^j(\lambda.c) \to \lambda.(\varphi_{k+1}^j c)$. $k + 1 < i + 1 < (k + 1) + j$, thus, by IH, $s_e(\varphi_k^j c)$ has no free occurrences of $\underline{i + 1}$. Hence, $s_e(\varphi_{k+1}^j a)$ has no free occurrences of $\underline{i}$.

3. • $a = \underline{n}$: If $n \leq k$, then by ($\varphi$-destruction), $\varphi_k^j \underline{n} \to \underline{n}$, where $i \geq k + j > k \geq n$. If $n > k$, then by ($\varphi$-destruction), $\varphi_k^j \underline{n} \to \underline{n + j - 1}$. Thus, $s_e(\varphi_k^j \underline{n})$ has a free occurrence of $\underline{i}$ iff $n = i - j + 1$.

   • $a = (c\ d)$: by ($\varphi$-app-transition), $\varphi_k^j(c\ d) \to (\varphi_k^j c\ \varphi_k^j d)$. By IH, $s_e(\varphi_k^j a) = (s_e(\varphi_k^j c)\ s_e(\varphi_k^j d))$ has a free occurrence of $\underline{i}$ iff either $c$ or $d$ has a free occurrence of $\underline{i - j + 1}$.

   • $a = \lambda.c$: by ($\varphi$-$\lambda$-transition), $\varphi_k^j(\lambda.c) \to \lambda.(\varphi_{k+1}^j c)$. Since $i + 1 \geq (k + 1) + j$, by IH, $s_e(\varphi_{k+1}^j c)$ has a free occurrence of $\underline{i + 1}$ iff $c$ has a free occurrence of $\underline{(i + 1) - j + 1}$. Hence, $s_e(\varphi_{k+1}^j a)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i - j + 1}$.

4. • $a = \underline{n}$: If $n < j$, then by ($\sigma$-destruction), $\underline{n}\,\sigma^j b \to \underline{n}$. Thus, $s_e(\underline{n}\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff $n = i$. If $n = j$, then by ($\sigma$-destruction), $\underline{n}\,\sigma^j b \to \varphi_0^j b$. Since $0 = k < i < k + j$, by item 2, $s_e(\varphi_0^j b)$ has no free occurrences of $\underline{i}$. If $n > j$, then by ($\sigma$-destruction), $\underline{n}\,\sigma^j b \to \underline{n - 1}$, where $n - 1 \geq j > i$.

   • $a = (c\ d)$: by ($\sigma$-app-transition), $(c\ d)\sigma^j b \to ((c\,\sigma^j b)\ (d\,\sigma^j b))$. Thus, by IH, $s_e(a\,\sigma^j b) = (s_e(c\,\sigma^j b)\ s_e(d\,\sigma^j b))$ has a free occurrence of $\underline{i}$ iff either $c$ or $d$ has a free occurrence of $\underline{i}$.

   • $a = \lambda.c$: by ($\sigma$-$\lambda$-transition), $(\lambda.c)\sigma^j b \to \lambda.(c\,\sigma^{j+1}b)$. Since $i + 1 < j + 1$, by IH, $s_e(c\,\sigma^{j+1}b)$ has a free occurrence of $\underline{i + 1}$ iff $c$ has a free occurrence of $\underline{i + 1}$. Hence, $s_e(a\,\sigma^{j+1}b)$ has a free occurrence of $\underline{i}$ iff $a$ has a free occurrence of $\underline{i}$.

5. • $a = \underline{n}$: If $n < j$, then by ($\sigma$-destruction), $\underline{n}\,\sigma^j b \to \underline{n}$, where $n < j \leq i$. If $n = j$, then by ($\sigma$-destruction), $\underline{n}\,\sigma^j b \to \varphi_0^j\, b$. Since $i \geq k + j$ where $k = 0$, by item 3, $s_e(\varphi_0^j\, b)$ has free occurrence of $\underline{i}$ iff $b$ has a free occurrence of $\underline{i - j + 1}$. If $n > j$, then by ($\sigma$-destruction) one has $\underline{n}\,\sigma^j b \to \underline{n-1}$. Thus, $s_e(\underline{n}\,\sigma^j b)$ has a free occurrence of $\underline{i}$ iff $n = i + 1$. Observe that $i + 1 > i \geq j$.

   • $a = (c\ d)$: by ($\sigma$-app-transition), $(c\ d)\sigma^j b \to ((c\,\sigma^j b)\ (d\,\sigma^j b))$. Thus, by IH, $s_e(a\,\sigma^j b) = (s_e(c\,\sigma^j b)\ \ s_e(d\,\sigma^j b))$ 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 $\underline{i - j + 1}$.

   • $a = \lambda.c$: by ($\sigma$-$\lambda$-transition), $(\lambda.c)\sigma^j b \to \lambda.(c\,\sigma^{j+1}b)$. Since $i + 1 \geq j + 1$, thus, by IH, $s_e(c\,\sigma^{j+1}b)$ has a free occurrence of $\underline{i+1}$ iff $c$ has a free occurrence of $\underline{(i+1)+1}$ or $c$ has a free occurrence of $\underline{j+1}$ and $b$ has a free occurrence of $\underline{(i+1) - (j+1) + 1} = \underline{i - j + 1}$. Consequently, $s_e(a\,\sigma^{j+1}b)$ has a free occurrence of $\underline{i}$ if, and only if, $a$ has a free occurrence of $\underline{i+1}$ or $a$ has a free occurrence of $\underline{j}$ and $b$ has a free occurrence of $\underline{i - j + 1}$. □

The next lemma shows that whenever an index occurs freely in $s_e(a)$ then it might occurs freely in $a$.

LEMMA 5.4
If $s_e(a)$ has a free occurrence of $\underline{i}$, then $\langle a, i \rangle \vdash_{COFI} True$.

*Proof.*   By induction on the structure of $a$.

   • $a = \underline{n}$: If $n = i$, then $\langle \underline{n}, i \rangle \vdash True$

   • $a = (b\ c)$: we have $\langle (b\ c), i \rangle \vdash \langle b, i \rangle \vee \langle c, i \rangle$. By IH, if $s_e(b)$ has a free occurrence of $\underline{i}$, then $\langle b, i \rangle \vdash True$. Analogously for $s_e(c)$. Thus if $s_e(b)$ or $s_e(c)$ have a free occurrence of $\underline{i}$, then $\langle (b\ c), i \rangle \vdash True$. Note that $s_e(b\ c) = (s_e(b)\ s_e(c))$.

   • $a = \lambda.b$: we have $\langle \lambda.b, i \rangle \vdash \langle b, i + 1 \rangle$. By IH, if $s_e(b)$ has a free occurrence of $\underline{i+1}$, then $\langle b, i + 1 \rangle \vdash True$. Thus, if $s_e(a) = \lambda.s_e(b)$ has a free occurrence of $\underline{i}$, then $\langle a, i \rangle \vdash True$.

   • $a = b\,\sigma^j c$:
     – if $i < j$, then $\langle b\,\sigma^j c, i \rangle \vdash \langle b, i \rangle$. By Lemma 5.3.4, if $s_e(b\,\sigma^j c)$ has a free occurrence of $\underline{i}$, then $s_e(b)$ has a free occurrence 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 5.3.5, if $s_e(b\,\sigma^j c)$ has a free occurrence of $\underline{i}$, then $s_e(b)$ has a free occurrence of $\underline{i+1}$ or ($s_e(b)$ has a free occurrence of $\underline{j}$ and $s_e(c)$ has a free occurrence of $\underline{i - j + 1}$). Thus, by IH, $\langle a, i \rangle \vdash True$. Observe that even if $s_e(b)$ has no free occurrences of $\underline{j}$, $\langle a, i \rangle$ is still assigned $True$.

   • $a = \varphi_k^j\, b$:
     – if $i \leq k$, then $\langle \varphi_k^j\, b, i \rangle \vdash \langle b, i \rangle$. By Lemma 5.3.1, if $s_e(\varphi_k^j\, b)$ has a free occurrence of $\underline{i}$, then $s_e(b)$ has a free occurrence of $\underline{i}$. Thus, by IH, $\langle b, i \rangle \vdash True$.
     – If $k < i < k + j$, then $\langle \varphi_k^j\, b, i \rangle \vdash False$. By Lemma 5.3.2, $s_e(\varphi_k^j\, b)$ has no free occurrences of $\underline{i}$.
     – If $i \geq k + j$, then $\langle \varphi_k^j\, b, i \rangle \vdash \langle b, i - j + 1 \rangle$. By Lemma 5.3.3, if $s_e(\varphi_k^j\, b)$ has a free occurrence of $\underline{i}$, then $s_e(b)$ has a free occurrence of $\underline{i - j + 1}$. Thus, by IH, $\langle b, i - j + 1 \rangle \vdash True$. □

When $a$ is a $s_e$-nf, the inverse is true.

**LEMMA 5.5**
If $a$ is a $s_e$-nf and $\langle a, i \rangle \vdash_{COFI} True$, then $a$ has a free occurrence of $\underline{i}$.

*Proof.*    Analogue of the proof of Lemma 5.4.                                    □

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

TABLE 4. $\eta_{\lambda s_e}$: the rewriting system for $\eta$-reduction in $\lambda s_e$

| | | | |
|---|---|---|---|
| $(a\ b)\,\eta^i$ | $\longrightarrow$ | $(a\,\eta^i\ b\,\eta^i)$ | ($\eta$-app-transition) |
| $(\lambda.a)\,\eta^i$ | $\longrightarrow$ | $\lambda.a\,\eta^{i+1}$ | ($\eta$-$\lambda$-transition) |
| $\underline{n}\,\eta^i$ | $\longrightarrow$ | $\begin{cases} \underline{n} & \text{if } n < i \\ \underline{n-1} & \text{if } n > i \end{cases}$ | ($\eta$-destruction) |
| $(a\,\sigma^j b)\,\eta^i$ | $\longrightarrow$ | $(a\,\eta^i)\,\sigma^{j-1}b$   if $i < j$ | ($\eta$-$\sigma$-transition 1) |
| $(a\,\sigma^j b)\,\eta^i$ | $\longrightarrow$ | $(a\,\eta^{i+1})\,\sigma^j(b\,\eta^{i-j+1})$   if $i \geq j$ | ($\eta$-$\sigma$-transition 2) |
| $(\varphi_k^j\,a)\,\eta^i$ | $\longrightarrow$ | $\varphi_{k-1}^j(a\,\eta^i)$   if $i \leq k$ | ($\eta$-$\varphi$-transition 1) |
| $(\varphi_k^j\,a)\,\eta^i$ | $\longrightarrow$ | $\varphi_k^{j-1}a$   if $k < i < k+j$ | ($\eta$-$\varphi$-transition 2) |
| $(\varphi_k^j\,a)\,\eta^i$ | $\longrightarrow$ | $\varphi_k^j(a\,\eta^{i-j+1})$   if $i > k$ and $i \geq k+j$ | ($\eta$-$\varphi$-transition 3) |

Observe that the $\eta_{\lambda s_e}$-rules have a similar structure to the rules of the detection of free indices of Definition 5.1, with a simple adaptation for checking and updating the free indices via the $\eta$-destruction rule, and for computing the correct updating of indices via the five transition rules for the $\sigma$ and $\varphi$ operators.

The position of an occurrence of the symbol $\eta$ in a term $a$ is denoted as a sequence of naturals in $\{1, 2\}$ and defined as follows: $\eta$ occurs at position $\varepsilon$ (the empty sequence) in $a\eta^i$; if $\eta$ occurs at position $I$ in $a'$ and $a$ is of the form $(a'\ b)$, $\lambda.a'$, $(a'\,\sigma^j b)$ or $(\varphi_k^j\,a')$, then $\eta$ occurs at position $1.I$ in a, and if $a$ is of the form $(b\ a')$ or $(b\,\sigma^j a')$, then $\eta$ occurs at position $2.I$ in $a$, and if $a$ is of the form $a'\eta^l$, then $\eta$ occurs at position $I$ in $a$ (and also at position $\varepsilon$). A position $I$ is said to be bigger than position $J$, whenever the sequence $I$ is a prefix of the sequence $J$. Notice that the occurrences of $\eta$ in a term $a$ are a multiset of sequences; for example supposing $a$ and $b$ are $\lambda s_e$-terms, a term of the form $(a\eta^l\eta^m\ (\varphi_k^j\,b\eta^n))\eta^r$ has a multiset of occurrences of $\eta$ of the form $\{\{\varepsilon, 1, 1, 2.1\}\}$.

The next lemma establishes the convergence of $\eta_{\lambda s_e}$. Its proof is quite simple. Compare and contrast with the case for $\eta_{\lambda\sigma}$ where we only established convergence for a sub-language $\mathfrak{J}$ and where the proof was much more involved.

**LEMMA 5.6**
$\eta_{\lambda s_e}$ is terminating, confluent and convergent

*Proof.*    $\eta_{\lambda s_e}$ is easily checked to be terminating and confluent. For the former, let $a$ be a term and consider the multiset of positions of occurrences of the symbol $\eta$ in $a$. Notice that one application of any of the rules of $\eta_{\lambda s_e}$ results in a term with its multiset of positions of occurrences of the symbol $\eta$ smaller than the original one. For the later, notice that $\eta_{\lambda s_e}$ is left linear and there is no possible overlapping between left-hand sides of the rules; thus, by orthogonality, confluence holds. The

$\eta_{\lambda s_e}$ normalisation of $a\eta^i$ can be conceived simply as a propagation of the symbol $\eta$ between the finite structure of $a$.　　□

One has the following property of $\eta_{\lambda s_e}$ which guarantees reductions for $\lambda s_e$-terms.

LEMMA 5.7

If $\langle a, i \rangle \vdash_{COFI} False$, then $\eta_{\lambda s_e}(a\,\eta^i)$ has no occurrences of the operator $\eta$.

*Proof.*　　By induction on the structure of $a$.

- $a = \underline{n}$ : 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 $\underline{n}\,\eta^i \to \underline{n-1}$.
- $a = (b\ c)$: 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 $\eta_{\lambda s_e}(b\,\eta^i)$ nor $\eta_{\lambda s_e}(c\,\eta^i)$ have occurrences of $\eta$.
- $a = \lambda.b$: one has $(\lambda.b)\,\eta^i \to \lambda.b\,\eta^{i+1}$. From $\langle \lambda.b, i \rangle \vdash False$ one has that $\langle b, i+1 \rangle \vdash False$. Thus, by IH, one has that $\eta_{\lambda s_e}(b\,\eta^{i+1})$ has no occurrences of $\eta$.
- $a = b\,\sigma^j c$: 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, $\eta_{\lambda s_e}(b\,\eta^i)$ has no occurrences of $\eta$.
  - 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 $\eta_{\lambda s_e}(b\,\eta^{i+1})$ nor $\eta_{\lambda s_e}(c\,\eta^{i-j+1})$ have occurrences of $\eta$.
- $a = \varphi_k^j\,b$: one has that $\langle \varphi_k^j\,b, i \rangle \vdash False$.
  - If $i \leq k$, then $(\varphi_k^j\,b)\,\eta^i \to \varphi_{k-1}^j\,(b\,\eta^i)$ and $\langle b, i \rangle \vdash False$. By IH, $\eta_{\lambda s_e}(b\,\eta^i)$ has no occurrences of $\eta$.
  - If $k < i < k+j$, then $(\varphi_k^j\,b)\,\eta^i \to \varphi_k^{j-1}\,b$ and since $b$ is a $\lambda s_e$-term, it has no occurrences of $\eta$.
  - If $i \geq k+j$, then $(\varphi_k^j\,b)\,\eta^i \to \varphi_k^j\,(b\,\eta^{i-j+1})$ and $\langle b, i-j+1 \rangle \vdash False$. By IH, $\eta_{\lambda s_e}(b\,\eta^{i-j+1})$ has no occurrences of $\eta$.　　□

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

DEFINITION 5.8 (Constructive (Eta) for $\boldsymbol{\lambda s_e}$)

Let $a$ be a $\lambda s_e$-term. The rule **(Eta)** is given by

$$\lambda.(a\ \underline{1}) \to \eta_{\lambda s_e}(a\,\eta^1) \qquad \text{if } \langle a, 1 \rangle \vdash_{COFI} False \qquad \text{(Eta)}$$

By Lemma 5.7, the condition $\langle a, 1 \rangle \vdash_{COFI} False$ guarantees that $\eta_{\lambda s_e}(a\,\eta^1)$ is a $\lambda s_e$-term. It follows an example of $\eta$-reduction.

EXAMPLE 5.9

Let $b = \lambda.(\lambda.((\underline{1}\,\sigma^2\underline{2})\ (\varphi_0^2\,\underline{2}))\ \underline{1})$. Let $a = \lambda.((\underline{1}\,\sigma^2\underline{2})\ (\varphi_0^2\,\underline{2}))$. One has $\langle a, 1 \rangle \vdash_{COFI} False$. Then $b$ is an $\eta$-redex. By the rules of Table 4 we have the following derivation

$$
\begin{array}{llll}
a\eta^1 & \longrightarrow & \lambda.\big(((\underline{1}\,\sigma^2\underline{2})\ (\varphi_0^2\,\underline{2}))\eta^2\big) & (\eta\text{-}\lambda\text{-transition}) \\
& \longrightarrow & \lambda.\big((\underline{1}\,\sigma^2\underline{2})\eta^2\ (\varphi_0^2\,\underline{2})\eta^2\big) & (\eta\text{-app-transition}) \\
& \longrightarrow & \lambda.\big(((\underline{1}\,\eta^3)\sigma^2(\underline{2}\,\eta^1))\ (\varphi_0^2\,\underline{2})\eta^2\big) & (\eta\text{-}\sigma\text{-transition 2}) \\
& \longrightarrow & \lambda.\big((\underline{1}\,\sigma^2(\underline{2}\,\eta^1))\ (\varphi_0^2\,\underline{2})\eta^2\big) & (\eta\text{-destruction}) \\
& \longrightarrow & \lambda.\big((\underline{1}\,\sigma^2\underline{1})\ (\varphi_0^2\,\underline{2})\eta^2\big) & (\eta\text{-destruction}) \\
& \longrightarrow & \lambda.\big((\underline{1}\,\sigma^2\underline{1})\ \varphi_0^2\,(\underline{2}\,\eta^1)\big) & (\eta\text{-}\varphi\text{-transition 3}) \\
& \longrightarrow & \lambda.\big((\underline{1}\,\sigma^2\underline{1})\ \varphi_0^2\,\underline{1}\big) & (\eta\text{-destruction})
\end{array}
$$

Then, by (Eta) defined above, $b \to \lambda.\big((\underline{1}\,\sigma^2\underline{1})\,\varphi_0^2\,\underline{1}\big)$.

The next lemma is necessary to state that the Definition 5.8 coincides with $\eta$-contraction for $\lambda$-terms in de Bruijn notation.

LEMMA 5.10
If $a$ is a term in $s_e$-nf and $\langle a, i \rangle \vdash_{COFI} False$, then $b = \eta_{\lambda s_e}(a\,\eta^i)$ is a $\lambda$-term in de Bruijn notation such that $\varphi_{i-1}^2 b \to_{s_e}^* a$.

*Proof.*

- $a = \underline{n}$: from $\langle \underline{n}, i \rangle \vdash False$ one has that $n \neq i$. If $n < i$, then $\underline{n}\,\eta^i \to \underline{n}$ and, by ($\varphi$-destruction), $\varphi_{i-1}^2\,\underline{n} \to \underline{n}$. If $n > i$, then $\underline{n}\,\eta^i \to \underline{n-1}$ and, by ($\varphi$-destruction), $\varphi_{i-1}^2\,\underline{n-1} \to \underline{(n-1)+2-1} = \underline{n}$.

- $a = (a_1\ a_2)$: one has that both $a_1$ and $a_2$ are in $s_e$-nf and from $\langle (a_1\ a_2), i \rangle \vdash False$ one has $\langle a_1, i \rangle \vdash False$ and $\langle a_2, i \rangle \vdash False$. By ($\eta$-app-transition) one has $(a_1\ a_2)\,\eta^i \to (a_1\eta^i\ a_2\eta^i)$, thus, by IH, $b_1 = \eta_{\lambda s_e}(a_1\eta^i)$ and $b_2 = \eta_{\lambda s_e}(a_2\eta^i)$ are $\lambda$-terms in de Bruijn notation such that $\varphi_{i-1}^2 b_1 \to_{s_e}^* a_1$ and $\varphi_{i-1}^2 b_2 \to_{s_e}^* a_2$, respectively. Since $b = \eta_{\lambda s_e}(a\,\eta_i^i) = (b_1\ b_2)$ and $\varphi_{i-1}^2 b \to (\varphi_{i-1}^2 b_1\ \varphi_{i-1}^2 b_2)$ by ($\varphi$-app-transition), the result holds for $a$ and $b$.

- $a = \lambda.a'$: from $\langle \lambda.a', i \rangle \vdash False$ one has that $\langle a', i+1 \rangle \vdash False$ and, by rule ($\eta$-$\lambda$-transition), $(\lambda.a')\,\eta^i \to \lambda.a'\eta^{i+1}$. Thus, by IH, one has that $b' = \eta_{\lambda s_e}(a'\eta^{i+1})$ is a $\lambda$-term in de Bruijn notation such that $\varphi_i^2 b' \to_{s_e}^* a'$. Since $b = \eta_{\lambda s_e}(a\,\eta_i^i) = \lambda.b'$ and $\varphi_{i-1}^2(\lambda.b') \to \lambda.\varphi_i^2 b'$ by ($\varphi$-$\lambda$-transition), the results holds for $a$ and $b$.

$\square$

By the operational semantics of the updating operator of the $\lambda s_e$, the $s_e$-nf of the term $\varphi_{i-1}^2 b$ is the same as $b$ but incrementing all de Bruijn indices greater than $i$ by one. Thus, taking $i = 1$, the lemma above states that the (Eta) rule given in Definition 5.8 coincides with the $\eta$-reduction for $\lambda$-calculus in de Bruijn notation.

## 5.2    The constructive Eta rule for $\lambda s_e$ preserves SR

The following property of $\eta_{\lambda s_e}$, related to types, is necessary in order to prove SR for the proposed rule (Eta).

LEMMA 5.11
If $\Gamma \vdash a : A$ and $\langle a, i \rangle \vdash_{COFI} False$, then $\Gamma_{<i}.\Gamma_{>i} \vdash \eta_{\lambda s_e}(a\,\eta^i) : A$.

*Proof.*    By induction on the structure of $a$. We write $N(a)$ for the $\eta_{\lambda s_e}$-nf of $a$.

- $a = \underline{n}$: Let $\Gamma \vdash \underline{n} : A$. By $\langle \underline{n}, i \rangle \vdash_{COFI} False$, $n \neq i$. If $n < i$, by ($\eta$-destruction), $\underline{n}\,\eta^i \to \underline{n}$ and $\Gamma_{<i}.\Gamma_{>i} \vdash \underline{n} : A$. If $n > i$, by ($\eta$-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), $\Gamma_{<i}.\Gamma_{>i} \vdash \underline{n-1} : A$

- $a = (b\ c)$: Let $\Gamma \vdash (b\ c) : A$. By the rule ($\eta$-app-transition) one has $N((b\ c)\,\eta^i) = (N(b\,\eta^i)\ N(c\,\eta^i))$. By the rule (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), $\Gamma_{<i}.\Gamma_{>i} \vdash (N(b\,\eta^i)\ N(c\,\eta^i)) : A$.

- $a = \lambda.b$: Let $\Gamma \vdash \lambda.b : A$. By ($\eta$-$\lambda$-transition), $N((\lambda.b)\,\eta^i) = \lambda.N(b\,\eta^{i+1})$. 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), $\Gamma_{<i}.\Gamma_{>i} \vdash \lambda.N(b\,\eta^{i+1}) : A$.

- $a = b\,\sigma^j c$: 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 ($\eta$-$\sigma$-transition 1), $N((b\,\sigma^j c)\,\eta^i) = (N(b\,\eta^i))\,\sigma^{j-1} c$. Then by IH, one has $\Gamma_{<i}.(\Gamma_{<j})_{>i}.B.\Gamma_{\geq j} \vdash N(b\,\eta^i) : A$. By (Sigma), $\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^i)\,\sigma^{j-1} c : A$.
  - If $i \geq j$, by ($\eta$-$\sigma$-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) $\Gamma_{<i}.\Gamma_{>i} \vdash N(b\,\eta^{i+1})\,\sigma^j N(c\,\eta^{i-j+1}) : A$.

- $a = \varphi_k^j b$: Let $\Gamma \vdash \varphi_k^j b : A$. By (Phi), $\Gamma_{\leq k}.\Gamma_{\geq k+j} \vdash b : A$.
  - If $i \leq k$, by ($\eta$-$\varphi$-transition 1), $N((\varphi_k^j b)\,\eta^i) = \varphi_{k-1}^j N(b\,\eta^i)$. Consequently by IH, $\Gamma_{<i}.(\Gamma_{\leq k})_{>i}.\Gamma_{\geq k+j} \vdash N(b\,\eta^i) : A$. Thus, by (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi_{k-1}^j N(b\,\eta^i) : A$.
  - If $k < i < k+j$, by ($\eta$-$\varphi$-transition 2), $N((\varphi_k^j b)\,\eta^i) = \varphi_k^{j-1} b$. Thus, by (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi_k^{j-1} b : A$.
  - If $k < i$ and $k+j \leq i$, by ($\eta$-$\varphi$-transition 3), $N((\varphi_k^j b)\,\eta^i) = \varphi_k^j N(b\,\eta^{i-j+1})$. By IH, one has that $\Gamma_{\leq k}.(\Gamma_{\geq k+j})_{<i-j+1-k}.\Gamma_{>i} \vdash N(b\,\eta^{i-j+1}) : A$. And finally, by (Phi), $\Gamma_{<i}.\Gamma_{>i} \vdash \varphi_k^j N(b\,\eta^{i-j+1}) : A$. $\qquad\square$

THEOREM 5.12 (SR for the constructive (Eta) in $\boldsymbol{\lambda s_e}$)
If $\Gamma \vdash \lambda.(a\,\underline{1}) : A$ and $\lambda.(a\,\underline{1}) \to_{Eta} b$, then $\Gamma \vdash b : A$.

*Proof.* Suppose that $\Gamma \vdash \lambda.(a\,\underline{1}) : A$. 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 $\langle a, 1 \rangle \vdash_{COFI} False$ and $b = \eta_{\lambda s_e}(a\,\eta^1)$. Consequently, by case $i = 1$ of Lemma 5.11, one obtains $\Gamma \vdash b : A$. $\qquad\square$

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

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

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

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

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

# 6  Conclusions and Future Work

We defined constructive explicit Eta rules for the $\lambda\sigma$- and the $\lambda s_e$-calculi which preserve subject reduction. These formalisations involve the presentation of specific

sub-calculi, given as well-behaved rewriting systems, for verifying the condition of Eta in each of these two calculi explicitly. The proposed definitions work in such a way that the construction of the Eta-contractum is given, while the condition of the rule is being checked. Thus, our formalisation is directly implementable from the constructive definition of the Eta rules for the simply-typed version of these calculi. In addition to turning decidable the definitions of Eta in [8, 7] and [3], our work contributes to making the informal implementations of the rules suggested in [5] and [2] more precise. In contrast to the rule Eta introduced in [8] for $\lambda\sigma$ ([3] for $\lambda s_e$), our constructive definition can be applied to non $\sigma$-normal forms (non $s_e$-nf respectively) in such a way that the rules Beta and Eta are put on an equal footing. One could look at explicit substitutions as an active ingredient to distinguish between local and global $\beta$-reduction (and hence as giving a form of explicit $\beta$-reduction). Our work goes a step further in making explicit reduction and we present systems in which both $\beta$- and $\eta$-reductions are explicit.

As future work, it is interesting to compare the efficiency of the implementations of the suggested Eta rules in both calculi.

## Funding

## Acknowledgments

## References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of 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 $\lambda s_e$-Style of Explicit Substitution. *The Journal of the Interest Group in Pure and Applied Logics*, 9(4):489–523, 2001.

[4] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[5] P. Borovanský. Implementation of Higher-Order Unification Based on Calculus of Explicit Substitutions. Volume 1012 of *Lecture Notes in Computer Science*, pages 363–368. 1995.

[6] D. Briaud. An explicit Eta rewrite rule. In *Typed lambda calculi and applications*, volume 902 of *Lecture Notes in Computer Science*, pages 94–108. 1995.

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

[8] T. Hardin. Eta-conversion for the languages of explicit substitutions. In *Algebraic and logic programming*, volume 632 of *Lecture Notes in Computer Science*, pages 306–321. 1992.

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

[10] D. Kesner. Confluence of extensional and non-extensional $\lambda$-calculi with explicit substitutions. *Theoretical Computer Science*, 238(1-2):183–220, 2000.

[11] F. Kamareddine and A. Ríos. Extending a lambda-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, 1997.

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

[13] P.-A. Melliès. Typed $\lambda$-calculi with explicit substitutions may not terminate in Proceedings of TLCA'95. *Lecture Notes in Computer Science*, 902, 1995.

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