A Semantics for step-wise substitution and reduction[*][†]

Fairouz Kamareddine[‡]

December 5, 1995

Abstract

We show the soundness of a λ-calculus $\mathcal{B}$ where de Bruijn indices are used, substitution is explicit, and reduction is step-wise. This is done by interpreting $\mathcal{B}$ in the classical calculus where the explicit substitution becomes implicit and de Bruijn indices become named variables. This is the first flat semantics of explicit substitution and step-wise reduction and the first clear account of exactly when α-reduction is needed.

Keywords: De Bruijn's indices, variable updating, substitution, reduction, soundness.

1 Introduction

Variables play a very demanding role in the reduction and substitution of the λ-calculus. This has lead in many cases to using implicit rather than explicit substitution. Implementations of the λ-calculus provide their own explicit substitution procedures as in HOL [GM 93], Nuprl [Con 86] and Authomath [NGdV 94]. Furthermore, research on theories of explicit substitution has been striving lately ([HL 89], [ACCL 91], [KN 93], [Mel 95], [BBLR 95] and [KR 95]). In this paper, we extend the calculus of [KN 93] (which is influenced by Authomath) giving $\mathcal{B}$, a calculus which uses de Bruijn indices and where reduction and substitution are step-wise and explicit. The species of variable names is cultivated and ordered so that a fine inter-marriage between de Bruijn's indices and variable names takes place. We show the consistency of the fine reduction and explicit substitution of $\mathcal{B}$ in terms of the classical λ-calculus and reflect on the use and necessity of α-conversion.

Basic to our work is the *item notation* (see [KN 96a] for advantages). To write classical terms into item notation, we use $\mathcal{I}$ where $\mathcal{I}(t) \equiv t$ if $t \in V$, $\mathcal{I}(\lambda_{x:t}.t') \equiv (\mathcal{I}(t)\lambda_x)\mathcal{I}(t')$ and $\mathcal{I}(tt') \equiv (\mathcal{I}(t')\delta)\mathcal{I}(t)$ (note the order). Hence, a term t is of the form $s_1 s_2 \ldots s_n t'$ where t' is a variable and s_i for $1 \leq i \leq n$ is an *item* (of the form $(t_i\omega)$ where ω is an operator such as δ

[*]I am grateful for the discussions with Jos Baeten, Henk Barendregt, Erik Barendsen, Inge Bethke, Tijn Borghuis, Herman Geuvers, Jeroen Krabbendam, Rob Nederpelt, Erik Poll and Peter Rodenburgh and for the helpful remarks received from them.

[†]I am grateful to the Department of Mathematics and Computing Science, Eindhoven University of Technology, for their financial support and hospitality from October 1991 to September 1992, and during various short visits since 1993. Furthermore, I am grateful to the Department of Mathematics and Computer Science, University of Amsterdam, and in particluar to Jan Bergstra and Inge Bethke for their hospitality during the preparation of this article. Finally, this work is supported by the EPSRC grant GR/K 25014 and by the ESPRIT Basic Research Action project "Types for Proofs and Programs".

[‡]Department of Computing Science, 17 Lilybank Gardens, University of Glasgow, Glasgow G12 8QQ, Scotland, *email*: fairouz@dcs.glasgow.ac.uk

or λ (with or without a subscript)). When the operators get increased to include substitution (σ), updating (φ) and decreasing (μ) operators, the respresentation of terms remains simple to describe and enables one to define reduction and substitution in a step-wise fashion where at every step it is clear which item moves inside (or over) which one. This step-wise fashion gives explicit substitution and enables local and global reduction as shown in [KN 93].

We provide a method which takes any term of the λ-calculus with named variables and implicit substitution, Λ, into $\mathcal{B}$ such that all α-equivalent terms in Λ are mapped into a unique element of $\mathcal{B}$. The other direction however, of mapping elements of $\mathcal{B}$ into elements of Λ is more difficult. This is because in $\mathcal{B}$, the λ's do not have variable names as subscripts and so we have to look for such subscripts in a way that no free variables in the term get bound. Moreover, a term in $\mathcal{B}$ represents a whole class of terms in Λ (α-equivalent terms). In translating $\mathcal{B}$ to Λ, we avoid α-conversion in Λ and associate to each term of $\mathcal{B}$ a unique term of Λ rather than an arbitrary element of the α-equivalence class. Now, having such a translation $[\![\cdot]\!]$ from $\mathcal{B}$ to Λ, we show that the variable updating, the substitution and the reduction rules in $\mathcal{B}$ are sound by showing that if $t \to t'$ where $\to$ is either σ-, or φ- or μ-reduction (excluding σ- or μ-generation and σ-transition, see below), then $[\![t]\!] \equiv [\![t']\!]$. Hence the rules which accommodate variable updating and substitution result in syntactically equal terms. We shall moreover, show that if $t \to t'$ where the reduction includes σ- or μ-generation, then $[\![t]\!] =_{\overline{\alpha\beta}} [\![t']\!]$. That is, the rules which actually reduce β-redexes in $\mathcal{B}$ are nothing more than the β rule in Λ. Finally if $\to$ is σ-transition then $[\![t]\!] =_{\overline{\alpha}} [\![t']\!]$. Like this, we provide a *flat* semantics where most reduction steps are mapped to syntactical equality and not to a corresponding reduction. This semantics shows that our reduction and substitution rules are a refinement of those of the classical calculus.

We believe that our approach is the first to be so precise about variable manipulation, substitution and reduction. There is never a confusion of which variable is the one manipulated and hence a machine can easily carry out our reduction strategies and translate the terms using variables in a straightforward manner. This approach should be considered in implementations of the λ-calculus. Our work here might look too involved, but we have actually carried out the hard part of manipulating variables once and for all.

2 Basic Notation

We take $I\!N$ to be the set of natural numbers, i.e. ≥ 0, $I\!P$ to be the set of positive natural numbers, i.e. > 0, $\mathbb{Z}$ to be the set of integers and take $i, j, m, n, \ldots$ to range over numbers. We let $\mathcal{F} = \{x_1, x_2, \ldots\}$ be an ordered set whose elements are all distinct and call the left infinite list of λs as drawn in Figure 1, the *free variable list* $\mathcal{F}$. We let V, the set of variables of Λ, be

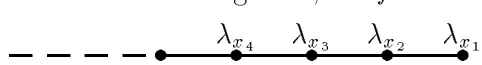


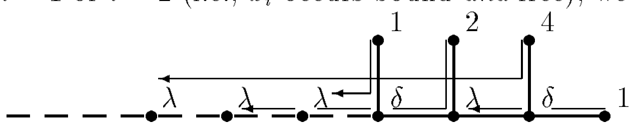
Figure 1: The free variable list $\mathcal{F}$

$\{\varepsilon\} \cup \mathcal{F}$ where ε can be looked at as a special variable or as a constant and is never used as a subscript for λ.[1] We take $\Xi = \{\varepsilon\} \cup I\!P$ to be the set of variables of $\mathcal{B}$ and let $v, v', v'', v_1, v_2, \ldots$ range over $\mathcal{F} \cup \Xi$.[2] We take $\Omega_\Lambda = \{\delta\} \cup \{\lambda_v; v \in \mathcal{F}\}$ and $\Omega_{\mathcal{B}} = \{\delta, \lambda, \sigma, \varphi, \mu\}$ to be the sets

[1] ε is added because it enables us to generalise the calculus. By taking all types of variables after λ to be ε, we obtain the type free λ-calculus ([KN 93]). ε has further uses such as the $\Box$ in [Bar 92].

[2] Note that $\varepsilon \in \Xi$ because no variable in $\mathcal{B}$ is a subscript of a λ.

of *operators* of Λ and $\mathcal{B}$ respectively. We let $\omega, \omega', \omega_1, \ldots$ range over $\Omega_\Lambda \cup \Omega_\mathcal{B}$ and use Ω to range over subsets of $\Omega_\mathcal{B}$. We let $t, t_1, \ldots$ range over terms of Λ and $\mathcal{B}$. We take $FV(t)$ and $BV(t)$ to be defined as usual and to represent the free and bound variables of t in Λ and $\mathcal{B}$; we assume that ε is neither free nor bound. For $r \in \{\alpha, \beta, \sigma, \varphi, \mu, \beta'', \beta'\}$, we assume that $\to_r$ is compatible (see [Bar 84]), call the reflexive transitive closure of $\to_r$, $\twoheadrightarrow_r$ and let $=_r$ the least equivalence relation closed under $\twoheadrightarrow_r$. $=$ is the least equivalence relation closed under $\twoheadrightarrow_\alpha$ and $\twoheadrightarrow_\beta$. We use $\equiv$ to be syntactic identity and when $t = t'$ in Λ, we write $\vdash_\Lambda t = t'$. We assume familiarity with de Bruijn indices. For example, for $i \neq 3, i \in I\!\!P$, $(\lambda_{x_i:x_2}.(x_i x_3))x_1$ or $(x_1\delta)(x_2\lambda_{x_i})(x_3\delta)x_i$ is written $(\lambda_1.14)1$ or $(1\delta)(2\lambda)(4\delta)1$ (see Figure 2) where the free variable list is used to account for the free variables x_1, x_2 and x_3. To translate $(x_1\delta)(x_2\lambda_{x_i})(x_3\delta)x_i$ when $i = 1$ or $i = 2$ (i.e., x_i occurs bound *and* free), we rename x_i to x_j for $j > 3$.



$(\lambda_{x_4:x_2}.x_4x_3)x_1$

$(x_1\delta)(x_2\lambda_{x_4})(x_3\delta)x_4$

$(1\delta)(2\lambda)(4\delta)1$

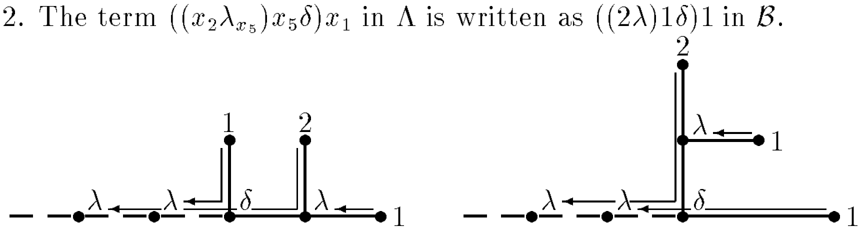
Figure 2: A tree with de Bruijn's indices

Terms of Λ and $\mathcal{B}$ are given by the following syntax:

$$\Lambda ::= V \mid I_\Lambda \Lambda \qquad \text{where } I_\Lambda ::= (\Lambda \Omega_\Lambda)$$
$$\mathcal{B} ::= \Xi \mid I_\mathcal{B} \mathcal{B} \qquad \text{where } I_\mathcal{B} ::= (\mathcal{B} \Omega)$$

We may write $\mathcal{B}^{\lambda\delta}$ when $\Omega = \{\lambda, \delta\}$, and call those terms $\Omega_{\lambda\delta}$-**terms**. Later on we increase Ω by adding σ, φ and μ. μ-terms will only be used with $\Omega_{\lambda\delta}$-terms. Example 2.1 shows terms both in Λ and $\mathcal{B}$. The translation between Λ and $\mathcal{B}$ will be given in Sections 3 and 5.

Ex 2.1 (The de Bruijn trees of these lambda terms are given in Figure 3.)
1. In $\mathcal{B}$, both $(x_1\delta)(x_2\lambda_{x_5})x_5$ and $(x_1\delta)(x_2\lambda_{x_3})x_3$ are denoted as $(1\delta)(2\lambda)1$. Note however, that $(x_1\delta)(x_2\lambda_{x_5})x_5 \not\equiv (x_1\delta)(x_2\lambda_{x_3})x_3$ for example, unless (α) is assumed in Λ.
2. The term $((x_2\lambda_{x_5})x_5\delta)x_1$ in Λ is written as $((2\lambda)1\delta)1$ in $\mathcal{B}$.



$(1\delta)(2\lambda)1$ | $((2\lambda)1\ \delta)1$

$(x_1\delta)(x_2\lambda_{x_5})x_5$ | $((x_2\lambda_{x_5})x_5\ \delta)x_1$

$(\lambda_{x_5:x_2} . x_5)x_1$ | $x_1(\lambda_{x_5:x_2} . x_5)$

Figure 3: de Bruijn trees with explicit free variable lists and reference numbers

Now, we define a number of concepts of Λ and $\mathcal{B}$ that will be used in the rest of the paper.

Def 2.2 *((main) items, (main) segments, ω-items, $\delta\lambda$-segments, body, weight, nl)*

- *If ω is an operator and t is a term then $(t\omega)$ is an* **item** *called* ω-**item**. *We use $s, s_1, s_i, \ldots$ to range over items.*

- *A concatenation of zero or more items is a* **segment**. *We use* $\overline{s}$, $\overline{s_1}$, $\overline{s_i}\ldots$ *to range over segments and write* $\emptyset$ *for the empty segment. A* **reducible** *or* **$\delta\lambda$-segment**, *is a δ-item next to a λ-item. If* $\overline{s} \equiv s_1 s_2 \ldots s_n$, *we call* $s_1, s_2, \ldots, s_n$ *the* **main items** *of* $\overline{s}$.
- *Each term t is the concatenation of zero or more items and a variable:* $t \equiv s_1 s_2 \ldots s_n v$. $s_1, s_2, \ldots, s_n$ *are called the* **main items** *and* $s_1 s_2 \ldots s_n$ *is the* **body** *of t; a concatenation of adjacent main items,* $s_m \ldots s_{m+k}$, *is called a* **main segment**.
- *The* **weight** *of a segment or a term is the number of its main items.*
- *We define* $nl(v) = \emptyset$ *if v is a variable,* $nl((t_1\omega)t_2) = nl(t_1) + nl(t_2)$ *if* $\omega \neq \lambda$ *and* $nl((t_1\lambda)t_2) = nl(t_1) + 1 + nl(t_2)$.

Ex 2.3 Let $t \equiv (\varepsilon\lambda)((1\delta)(\varepsilon\lambda)1\delta)(2\lambda)1$ and $\overline{s} \equiv (\varepsilon\lambda)((1\delta)(\varepsilon\lambda)1\delta)(2\lambda)$. The main items of t and $\overline{s}$ are $(\varepsilon\lambda)$, $((1\delta)(\varepsilon\lambda)1\delta)$ and (2λ), being a λ-, a δ-, and a λ-item. $((1\delta)(\varepsilon\lambda)1\delta)(2\lambda)$ is a main ($\delta\lambda$-) segment of t and $\overline{s}$. Also, $\overline{s}$ is a $\lambda\delta\lambda$-segment, which is a main segment of t. Note that $weight(t)$ is not necessarily the same as $nl(t)$ (which counts the number of λs in t). For example, $weight(((1\lambda)2\lambda)3) = 1$ whereas $nl(((1\lambda)2\lambda)3) = 2$.

Def 2.4 *(Substitution in Λ) If t, t' are terms in Λ and v is a variable in $\mathcal{F}$, we define the result of substituting t' for all the free occurrences of v in t as follows:*

$$t[v := t'] =_{df} \begin{cases} t' & \text{if } t \equiv v \\ t & \text{if } t \equiv v' \not\equiv v \text{ or } t \equiv \varepsilon \\ (t_2[v := t']\delta)t_1[v := t'] & \text{if } t \equiv (t_2\delta)t_1 \\ (t_2[v := t']\lambda_v)t_1 & \text{if } t \equiv (t_2\lambda_v)t_1 \\ (t_2[v := t']\lambda_{v'})t_1[v := t'] & \text{if } t \equiv (t_2\lambda_{v'})t_1, v \not\equiv v', \\ & (v \notin FV(t_1) \text{ or } v' \notin FV(t')) \\ (t_2[v := t']\lambda_{v''})t_1[v' := v''][v := t'] & \text{if } t \equiv (t_2\lambda_{v'})t_1, v \not\equiv v', v \in FV(t_1), \\ & v' \in FV(t'), v'' \text{is the first variable} \\ & \text{in } \mathcal{F} \text{ which does not occur in } (t\delta)t' \end{cases}$$

The (α) and (β) axioms in Λ are defined as follows:

(α)	$(t\lambda_v)t' \to_\alpha (t\lambda_{v'})t'[v := v']$ where $v' \notin FV(t')$
(β)	$(t''\delta)(t\lambda_v)t' \to_\beta t'[v := t'']$

3 Translating Λ in $\mathcal{B}$

We enumerate $\mathcal{F}$ via $\dagger$, so that: $\dagger x_1 = 1, \dagger x_2 = 2, \dagger x_3 = 3, \ldots$ and define, for $v \in \mathcal{F}$, $\dagger\lambda_v$ to be λ, $\dagger\delta$ to be δ and $\dagger\varepsilon$ to be ε. We need the following notions:

Def 3.1 *($term_i$) We define $term_i$ to be a partial function which takes non empty segments of Λ and returns terms of Λ as follows:*
$term_1((t_1\omega_1)\overline{s}) =_{df} t_1$, *and* $term_i((t_1\omega_1)\overline{s}) =_{df} term_{i-1}(\overline{s})$, *for* $i \geq 2, \overline{s} \neq \emptyset$.

Def 3.2 *(lam_i) lam_i takes $\overline{s}$ of Λ and returns $(\lambda_{v_1})(\lambda_{v_2})\ldots(\lambda_{v_k})$ obtained by removing all the main δ-items from the first $(i-1)$ main-items of $\overline{s}$ and by removing all the t's from the main λ-items $(t\lambda_v)$ of these $(i-1)$ main-items. lam_i is defined as follows:*

$$\begin{array}{lll} lam_i(\overline{s}) & =_{df} & \emptyset \\ lam_i((t\lambda_v)\overline{s}) & =_{df} & (\lambda_v)lam_{i-1}(\overline{s}) \quad \text{for } i \geq 2 \text{ and } weight(\overline{s}) \geq i-2 \\ lam((t\delta)\overline{s}) & =_{df} & lam(\overline{s}) \quad \text{for } i \geq 2 \text{ and } weight(\overline{s}) \geq i-2 \end{array}$$

Let $Seq_{i=1}^{i=n}(t_i\omega_i)$ be $(t_1\omega_1)(t_2\omega_2)\dots(t_n\omega_n)$, $n \geq 0$. The translation from Λ into $\mathcal{B}$ is as follows:

Def 3.3 *(b) For $t, t_1, t_2 \in \Lambda, v, v' \in \mathcal{F}, \overline{s}$ segment of Λ, we define b as follows:*

$$\begin{array}{lll|lll} b(t) & =_{df} & b'(t,\emptyset) & b'(v,\overline{s}(\lambda_v)) & =_{df} & 1 \\ b(\overline{s}) & =_{df} & \texttt{body}(b(\overline{s}\varepsilon)) & b'(v,\overline{s}(\lambda_{v'})) & =_{df} & 1 + b'(v,\overline{s}) \ \textit{if } v' \not\equiv v \\ b'(\varepsilon,\overline{s}) & =_{df} & \varepsilon & b'((t_1\lambda_v)t_2,\overline{s}) & =_{df} & (b'(t_1,\overline{s})\lambda)b'(t_2,\overline{s}(\lambda_v)) \\ b'(v,\emptyset) & =_{df} & \dagger v \ (\textit{note } v \not\equiv \varepsilon) & b'((t_1\delta)t_2,\overline{s}) & =_{df} & (b'(t_1,\overline{s})\delta)b'(t_2,\overline{s}) \end{array}$$

Here $b'(v,\overline{s})$ finds the de Bruijn number corresponding to v whithin context $\overline{s}$ (see Ex 3.5). $b'((t_1\lambda_v)t_2,\overline{s})$ translates t_1 with respect to $\overline{s}$ and t_2 with respect to $\overline{s}(\lambda_v)$.

Lem 3.4 *If $\overline{s_1}, \overline{s_2}$ are segments of $\Lambda, v \in \mathcal{F} \cup \{\varepsilon\}$, then*
$b'(\overline{s_1}v,\overline{s_2}) = Seq_{i=1}^{i=n}(b'(term_i(\overline{s_1}),\overline{s_2}lam_i(\overline{s_1}))\dagger op_i(\overline{s_1}))b'(v,\overline{s_2}lam_{n+1}(\overline{s_1}))$, for $n = weight(\overline{s_1})$.

Proof: *By induction on the length of $\overline{s_1}$.* □

So, if $t \equiv (t_1\omega_1)(t_2\omega_2)\dots(t_n\omega_n)v \equiv \overline{s_1}v \in \Lambda$, then $b'(t,\overline{s_2}) = (t'_1 \dagger \omega_1)(t'_2 \dagger \omega_2)\dots(t'_n \dagger \omega_n)v'$ where $t'_i \equiv b'(t_i,\overline{s_2}lam_i(\overline{s_1}))$ and $v' \equiv b'(v,\overline{s_2}lam_{n+1}(body(t)))$. Hence, t and $b'(t,\overline{s_2})$ have the same trees, except that λ's lose their subscripts and variables are replaced by correct indices found by tracing the λ's. That is why, in t'_i, we had to attach all the λs preceding t'_i.

Ex 3.5
1. $b((x_1\lambda_{x_4})(x_2\lambda_{x_3})x_4) \equiv (b'(x_1,\emptyset)\lambda)(b'(x_2,(\lambda_{x_4}))\lambda)b'(x_4,(\lambda_{x_4})(\lambda_{x_3})) \equiv (\dagger x_1\lambda)(3\lambda)2 \equiv (1\lambda)(3\lambda)2.$
2. $b((x_1\delta)(x_2\lambda_{x_4})(x_3\delta)x_4) \equiv (1\delta)(2\lambda)(4\delta)1.$
3. $b(((x_3\lambda_{x_4})x_4\delta)x_1) \equiv b'((x_3\lambda_{x_4})x_4,\emptyset)\delta)b'(x_1,\emptyset) \equiv ((b'(x_3,\emptyset)\lambda)b'(x_4,(\lambda_{x_4}))\delta)1 \equiv ((3\lambda)1\delta)1$

Lem 3.6 *For any t in Λ, $b(t)$ is well defined.*

Proof: *By induction on $t \in \Lambda$.* □

b is not injective: $b((x_1\lambda_{x_2})x_2) \equiv b((x_1\lambda_{x_3})x_3)$ but $(x_1\lambda_{x_2})x_2 \not\equiv (x_1\lambda_{x_3})x_3$. b however is surjective (see Lem 5.16). The following lemma is informative about b.

Lem 3.7 *If t, t' are terms in Λ such that $t =_\alpha t'$ then $b(t) \equiv b(t')$.*

Proof: *By induction on $t =_\alpha t'$.* □

4 Axioms of $\mathcal{B}$

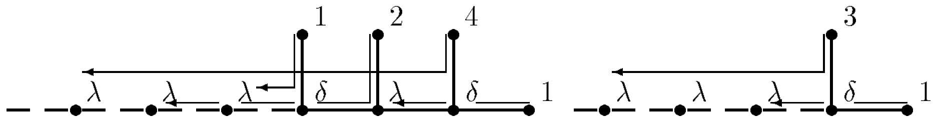
$(x_1\delta)(x_2\lambda_{x_4})(x_3\delta)x_4$ β-reduces to $(x_3\delta)x_1$. Using de Bruijn's indices, this is $(1\delta)(2\lambda)(4\delta)1$ reduces to $(3\delta)1$. In fact, if you look at Figure 4, you see that what is happening is that the $\delta\lambda$-segment $(1\delta)(2\lambda)$ has been cut off the tree, and the 4 has been decreased to 3 as we have lost one λ. The 1 in $(4\delta)1$ is replaced by the 1 of (1δ) giving $(3\delta)1$. We could say that when contracting $(t_1\delta)(t_2\lambda)$ in $(t_1\delta)(t_2\lambda)t$, all free variables in t must be decreased by 1 and all variables in t that are bound by the λ of $(t_2\lambda)$ must be replaced by t_1. This can be tricky however, for assume we take $t \equiv (\varepsilon\lambda)t'$ and write the rule as:

$$(t_1\delta)(t_2\lambda)t \to_\beta t[1 := t_1, 2 := 1, 3 := 2, \dots] \text{ (where substitution is simultaneous)}$$

then replacing $((\varepsilon\lambda)t')[1 := t_1, 2 := 1, 3 := 2, \dots]$ by $(\varepsilon\lambda)t'[1 := t_1, 2 := 1, 3 := 2, \dots]$ would not work. It should be: $(\varepsilon\lambda)t'[1 := 1, 2 := t_1[1 := 2, 2 := 3, \dots], 3 := 2, \dots]$.

Based on this observation, we need to increment variables (via φ) correctly in a term.

Rem 4.1 *(Compatibility)* Let $r \in \{\sigma, \varphi, \mu\}$. We introduce $\to_r$ as a relation between segments, although it is meant to be a relation between terms. Rule $\overline{s} \to_r \overline{s'}$ states that $t \to_r t'$ when a segment $\overline{s}$ occurs in t, where t' is the result of the replacement of $\overline{s}$ by $\overline{s'}$ in t.



$(\lambda_{x_i:x_2}.x_i x_3)x_1$ $\qquad$ $x_1 x_3$

$(x_1\delta)(x_2\lambda_{x_i})(x_3\delta)x_i$ $\qquad$ $(x_3\delta)x_1$

$(1\delta)(2\lambda)(4\delta)1$ $\qquad$ $(3\delta)1$

Figure 4: β-reduction in our notation

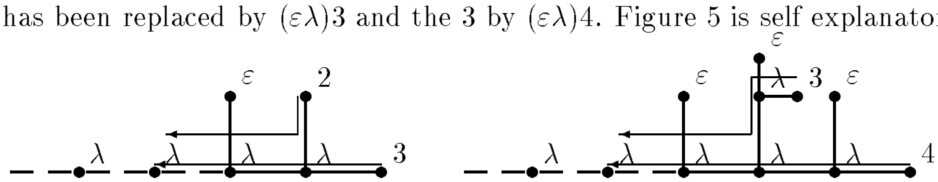
4.1 φ-reduction

We index φ with two parameters $k \geq 0$ and $i \geq 1$. $\forall i \geq 1$, let $\varphi^{(i)}$ denote $\varphi^{(0,i)}$ and φ denote $\varphi^{(1)}$. The intention of the superscripts when $(\varphi^{(k,i)})$ travels through t_1 is the following:

- i preserves the **increment** for $FV(t_1)$ and does not increase when passing other λ's.
- k counts the λ's that are internally passed by in t_1 (k = '**threshold**') and increases when passing another λ. Only variables $> k$ are increased, as the rest are bound.

Updating means all free variables in t_1 increase with an amount of i; k identifies the free variables in t_1. Updating variables by looking at the tree is easy: count the λ's you have gone through before a free variable and increase the free variable by that number.

Ex 4.2 Replacing in $(\varepsilon\lambda)(2\lambda)3$, the 2 and the 3 by $(\varepsilon\lambda)2$ results in $(\varepsilon\lambda)((\varepsilon\lambda)3\lambda)(\varepsilon\lambda)4$. I.e. the 2 has been replaced by $(\varepsilon\lambda)3$ and the 3 by $(\varepsilon\lambda)4$. Figure 5 is self explanatory.



$\lambda_{x_2:\varepsilon}.\lambda_{x_3:x_1}.x_1$ $\qquad$ $\lambda_{x_2:\varepsilon}.\lambda_{x_3:(\lambda_{x_4:\varepsilon}.x_1)}.\lambda_{x_4:\varepsilon}.x_1$

$(\varepsilon\lambda_{x_2})(x_1\lambda_{x_3})x_1$ $\qquad$ $(\varepsilon\lambda_{x_2})((\varepsilon\lambda_{x_4})x_1\lambda_{x_3})(\varepsilon\lambda_{x_4})x_1$

$(\varepsilon\lambda)(2\lambda)3$ $\qquad$ $(\varepsilon\lambda)((\varepsilon\lambda)3\lambda)(\varepsilon\lambda)4$

Figure 5: Substitution in our notation

The definition below formalises the updating process.

Def 4.3 *(φ-reduction) Let $k \in I\!N, i \in I\!P, v \in \Xi$ and t an $\Omega_{\lambda\delta}$-term.*

(φ-transition rules:)
$$(\varphi^{(k,i)})(t\lambda) \to_\varphi ((\varphi^{(k,i)})t\lambda)(\varphi^{(k+1,i)})$$
$$(\varphi^{(k,i)})(t\delta) \to_\varphi ((\varphi^{(k,i)})t\delta)(\varphi^{(k,i)})$$

(φ-destruction rules:)
$$(\varphi^{(k,i)})v \to_\varphi v+i \quad \textit{if } v > k$$
$$(\varphi^{(k,i)})v \to_\varphi v \quad \textit{if } v \leq k \textit{ or } v \equiv \varepsilon$$

Ex 4.4 In substituting $(\varepsilon\lambda)2$ for 2 in $(\varepsilon\lambda)(2\lambda)3$, we compensate for the preceding λ in $(\varepsilon\lambda)(2\lambda)3$. We substitute $(\varphi^{(0,1)})(\varepsilon\lambda)2$ for this 2:
$(\varepsilon\lambda)((\varphi^{(0,1)})(\varepsilon\lambda)2\lambda)3 \to_\varphi (\varepsilon\lambda)(((\varphi^{(0,1)})\varepsilon\lambda)(\varphi^{(1,1)})2\lambda)3 \twoheadrightarrow_\varphi (\varepsilon\lambda)((\varepsilon\lambda)3\lambda)3$
Similarly, in the substitution of $(\varepsilon\lambda)2$ for 3 in $(\varepsilon\lambda)(2\lambda)3$, we compensate for two extra λs:
$(\varepsilon\lambda)(2\lambda)(\varphi^{(0,2)})(\varepsilon\lambda)2 \twoheadrightarrow_\varphi (\varepsilon\lambda)(2\lambda)(\varepsilon\lambda)4$.

4.2 σ-reduction

σ-items can move through the branches of the term, step-wise, from one node to an adjacent one, until they reach a leaf of the tree. At the leaf, if appropriate, a σ-item (or a *substitution item*) can cause the desired substitution effect. We use σ as an *indexed* operator: $\sigma^{(1)}, \sigma^{(2)}, \ldots$. The intended meaning of a σ-item $(t'\sigma^{(i)})$ is: t' is a candidate to be substituted for one or more occurrences of a certain variable; i selects the appropriate occurrences.

Def 4.5 *(one-step σ-reduction) Let $i \in I\!P, v \in \Xi, t_1, t_2$ $\Omega_{\lambda\delta}$-terms.*

(σ-generation rule:) $\quad (t_1\delta)(t_2\lambda) \to_\sigma (t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})$

(σ-transition rules:)

$$(t_1\sigma^{(i)})(t_2\lambda) \to_\sigma ((t_1\sigma^{(i)})t_2\lambda)((\varphi)t_1\sigma^{(i+1)})$$
$$(t_1\sigma^{(i)})(t_2\delta) \to_\sigma ((t_1\sigma^{(i)})t_2\delta)(t_1\sigma^{(i)})$$

(σ-destruction rules:)

$$(t_1\sigma^{(i)})i \to_\sigma t_1$$
$$(t_1\sigma^{(i)})v \to_\sigma v \text{ if } v \not\equiv i$$

Note that our σ-transition rules do not allow for σ-items to "pass" other σ-items. The following shows that σ-reduction reaches all occurrences to be substituted.

Lem 4.6 *In $(t_1\delta)(t_2\lambda)t_3$, σ-reduction substitutes t_1 for all occurrences of the variables bound by the λ of $(t_2\lambda)$ in t_3. I.e., there is a path for global β-reduction.*

Proof: *The proof is by an easy induction on t_3 in $(t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})t_3$.* □

Lem 4.7 *In $(t_1\sigma^{(i)})t_2$, σ-reduction substitutes t_1 for all occurrences of variables in t_2 which are bound by the same λ being the i-th entry (from the right) in the free variable list of t_2. Moreover, the (φ)s look after the updating of t_2.*

Proof: *By induction on t_2, noting that during propagation, when the σ-item passes a λ, the superscript of σ is incremented, keeping track of the variable to be substituted for.* □

Ex 4.8

1. $(2\sigma^{(1)})(4\delta)1 \to_\sigma ((2\sigma^{(1)})4\delta)(2\sigma^{(1)})1 \twoheadrightarrow_\sigma (4\delta)2.$
2. $((3\delta)2\sigma^{(1)})(1\lambda)1 \to_\sigma (((3\delta)2\sigma^{(1)})1\lambda)((\varphi)(3\delta)2\sigma^{(2)})1 \twoheadrightarrow_{\sigma\varphi} ((3\delta)2\lambda)((4\delta)3\sigma^{(2)})1 \to_\sigma ((3\delta)2\lambda)1.$
3. $((3\delta)2\sigma^{(4)})(1\lambda)1 \to_\sigma (((3\delta)2\sigma^{(4)})1\lambda)((\varphi)(3\delta)2\sigma^{(5)})1 \twoheadrightarrow_{\sigma\varphi} (1\lambda)((4\delta)3\sigma^{(5)})1 \to_\sigma (1\lambda)1.$
4. $(1\delta)(2\lambda)(3\lambda)2 \to_\sigma (1\delta)(2\lambda)((\varphi)1\ \sigma^{(1)})(3\lambda)2 \to_\sigma$
   $(1\delta)(2\lambda)(((\varphi)1\ \sigma^{(1)})3\lambda)((\varphi)(\varphi)1\ \sigma^{(2)})2 \twoheadrightarrow_{\sigma,\varphi}$
   $(1\delta)(2\lambda)(((\varphi)1\ \sigma^{(1)})3\lambda)3 \to_\sigma (1\delta)(2\lambda)(3\lambda)3$

The following shows that the bond between variables and their binding λ's is maintained.

Lem 4.9 *If $\overline{s}(t_1\delta)(t_2\lambda)t \to_\sigma \overline{s}(t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})t$ then in $\overline{s}(t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})t$, all variable occurrences are bound by the same λ's which bound them in $\overline{s}(t_1\delta)(t_2\lambda)t$.*

Proof: *left to the reader.* □

To get local substitution, one adds to Def 4.5 the σ-destruction rule: $(t_1\sigma^{(i)})t \to_\sigma t$ ([KN 93]).

4.3 β-reduction

In the σ-generation rule, the reducible segment may be "without customers" and so σ-generation is undesirable since it leads to useless efforts. Hence we restrict σ-generation to those cases where the main λ of the reducible segment binds at least one variable. When this is *not* the case, we speak of a **void $\delta\lambda$-segment** (which may be removed by replacing it by $(\mu^{(1)})$). This can be compared to the application of a constant function to some argument; the result is always the (unchanged) body of the function. The meaning of $(\mu^{(i)})t$ is: decrease by 1 all variables in $t > i$. Variables $\leq i$ in t are bound by some λs in t and hence should not be decreased. Now the μ rules are defined as follows:

Def 4.10 *(μ-reduction) Let $t_1, t_2, t \in \Omega_{\lambda\delta}$-terms, $v \in \Xi$ and $i \in I\!P$.*

(μ-generation rule:)	$(t_1\delta)(t_2\lambda)t \to_\mu (\mu^{(1)})t$ *if* $(t_1\delta)(t_2\lambda)$ *is void in* t
(μ-transition rules:)	$(\mu^{(i)})(t\lambda) \to_\mu ((\mu^{(i)})t\lambda)(\mu^{(i+1)})$ $(\mu^{(i)})(t\delta) \to_\mu ((\mu^{(i)})t\delta)(\mu^{(i)})$
(μ-destruction rules:)	$(\mu^{(i)})v \to_\mu v$ *if* $v \equiv \varepsilon$ *or* $v < i$ $(\mu^{(i)})v \to_\mu v-1$ *if* $i < v$

Note in the second μ-destruction rule that $v > 1$ as $i \geq 1$. Note moreover that we never reach the case where we get $(\mu^{(i)})i$ (see Lem 4.13).

Def 4.11 *(One-step β-reduction $\to_{\beta'}$) One-step β-reduction of an $\Omega_{\lambda\delta}$-term is the combination of one σ-generation from a $\delta\lambda$-segment $\overline{s}$, the transition of the generated σ-item through the appropriate subterm in a global manner, followed by a number of σ-destructions, and updated by φ-items until again an $\Omega_{\lambda\delta}$-term is obtained. Finally, we replace the now void segment $\overline{s}$ by $(\mu^{(1)})t$ and we use the μ-reduction rules to dispose completely of μ in $(\mu^{(1)})t$.*

Ex 4.12 $(4\delta)(\lambda)(1\lambda)(1\lambda)3 \to_{\beta'} (4\lambda)(1\lambda)6$:

$$
\begin{array}{lll}
(4\delta)(\lambda)(1\lambda)(1\lambda)3 & \to_\sigma & (4\delta)(\lambda)((\varphi)4\sigma^{(1)})(1\lambda)(1\lambda)3 \\
 & \twoheadrightarrow_{\sigma,\varphi} & (4\delta)(\lambda)(5\lambda)(1\lambda)7 \\
 & \to_\mu & (\mu^{(1)})(5\lambda)(1\lambda)7 \\
 & \to_\mu & ((\mu^{(1)})5\lambda)(\mu^{(2)})(1\lambda)7 \\
 & \to_\mu & (4\lambda)(\mu^{(2)})(1\lambda)7 \\
 & \twoheadrightarrow_\mu & (4\lambda)(1\lambda)(\mu^{(3)})7 \\
 & \to_\mu & (4\lambda)(1\lambda)6
\end{array}
$$

The following Lemma is needed when discussing the semantics of μ-reduction:

Lem 4.13 *If t is an $\Omega_{\lambda\delta}$-term and $t \twoheadrightarrow_\mu t'$ then for all $(\mu^{(i)})t''$ subterm of t' with t'' an $\Omega_{\lambda\delta}$-term, we have that i does not refer to any free variable of t''. In particular, if $t \twoheadrightarrow_\mu t'$ then we never find in t', $(\mu^{(i)})i$ as a subterm.*

Proof: *By induction on $\twoheadrightarrow_\mu$.* □

5 Translating $\mathcal{B}$ in Λ

Here, we have to be careful. For example, we can translate $(\lambda)2$ as any of $(\lambda_{x_i})x_1$ for $i \neq 1$ but not as $(\lambda_{x_1})x_1$. The following example gives another case where we have to be careful:

Ex 5.1 $t \equiv ((1\delta)2\lambda)(1\lambda)3$ has for any $i, j \neq 1$, $((x_1\delta)x_2\lambda_{x_i})(x_i\lambda_{x_j})x_1$ as a corresponding Λ-term. Now the subterm $(1\lambda)3$ of t should be considered relative to a free variable list extended with λ_{x_i}: $\ldots, \lambda_{x_4}, \lambda_{x_3}, \lambda_{x_2}, \lambda_{x_1}, \lambda_{x_i}$, and hence corresponds with $(x_i\lambda_{x_j})x_1$ for $j \neq 1$.

To avoid choosing wrong subscripts of λ's, we work at a mid-level $\overline{\Lambda}$, between $\mathcal{B}$ and Λ. In $\overline{\Lambda}$, subscripts of λ's will be in a list $\updownarrow = x', x'', \ldots$ such that $\mathcal{F} \cap \updownarrow = \emptyset$. We assume all elements of $\updownarrow$ are distinct. We take $\Theta = \updownarrow \cup \mathcal{F}$, let $\theta, \theta_1, \theta_2, \theta', \cdots$ range over Θ, and $X, X', X_1, X_2, \ldots$ range over $\updownarrow$. We call elements of $\mathcal{F}$ free variables and elements of $\updownarrow$ bound variables.

Def 5.2 *($\overline{\Lambda}$) Terms of $\overline{\Lambda}$ are defined similarly to those of Λ except that all bound variables are indexed by elements from $\updownarrow$ (all free variables are in $\mathcal{F} \cup \updownarrow$).*

Examples of terms of $\overline{\Lambda}$ are ε, $(x_1\lambda_{x'})x'$ and $(x_1\lambda_{x'})(x'\delta)x''$. Bound and free variables, α, β, and $\equiv$ defined for Λ can be easily extended to $\overline{\Lambda}$. We use $FV(t)$ and $BV(t)$ for the free and bound variables of t in $\overline{\Lambda}$. We use $\overline{\alpha}$, $\overline{\beta}$ for α and β in $\overline{\Lambda}$.

Def 5.3 *(Substitution in $\overline{\Lambda}$) If t, t' are terms in $\overline{\Lambda}$ and if $v \in \mathcal{F} \cup \updownarrow$, then $t[v := t']'$ is exactly defined as in Def 2.4 except that, $[v := t']$ is replaced everywhere by $[v := t']'$, $[v' := v'']$ is replaced by $[v' := v'']'$ and in the last clause, $\mathcal{F}$ is replaced by $\updownarrow$.*

If $t \in \overline{\Lambda}$ translates $t' \in \mathcal{B}$, then $FV(t) \subseteq \mathcal{F}$ and $BV(t) \subseteq \updownarrow$ by Lem 5.42. In this case, t can be mapped to Λ by replacing its $\mathcal{I}$-variables by variables in $\mathcal{F}$ which do not occur in t:

Def 5.4 *(Translating $\overline{\Lambda}$ in Λ via τ) If t is a term in $\overline{\Lambda}$ such that $FV(t) \subseteq \mathcal{F}$ and $BV(t) \subseteq \updownarrow$ then we translate t to t' by first looking for the biggest free variable in t, x_i for $i \in I\!\!P$, and for the smallest bound variable in t. We replace all the occurrences of this bound variable by x_{i+1}. Then we replace the second smallest bound variable by x_{i+2} and so on until no variables from $\updownarrow$ appear in t. We call the translation of the $\overline{\Lambda}$-term t in Λ, $\tau(t)$.*

Ex 5.5 $(\lambda)2$ and $((1\delta)2\lambda)(1\lambda)3$ translate in $\overline{\Lambda}$ as $(\lambda_{x'})x_1$ and $((x_1\delta)x_2\lambda_{x'})(x'\lambda_{x''})x_1$ as we shall see. Now, these terms in $\overline{\Lambda}$ are transformed into terms of Λ in a *unique* way as follows:
The greatest variable of $\mathcal{F}$ in $(\lambda_{x'})x_1$ is x_1, hence x' gets replaced by x_2, giving $(\lambda_{x_2})x_1$.
The greatest variable of $\mathcal{F}$ in $((x_1\delta)x_2\lambda_{x'})(x'\lambda_{x''})x_1$ is x_2, hence all occurrences of x', x'' get replaced by x_3, x_4 respectively giving $((x_1\delta)x_2\lambda_{x_3})(x_3\lambda_{x_4})x_1$.

As Λ and $\overline{\Lambda}$ are similar, we avoid the trivial step of translating between Λ and $\overline{\Lambda}$ and show the soundness in $\overline{\Lambda}$. This simplification does not affect the results of this paper.

5.1 Variables and lists

We assume the usual basic list operations such as concatenation $+$ and *head* and *tail*, *hd* and *tl*. For $i \in I\!\!P$, we take $hd^1 =_{df} hd$ and $hd^{i+1} =_{df} hd \circ hd^i$, and we define tl^i similarly. Moreover, the set of operators $\setminus, \subset, \subseteq$ and $\in$ are also applicable for lists and we will mix sets and lists at will. We take $\overline{v}, \overline{v'}, \overline{v_1}, \overline{v_2}, \ldots$ to range over (finite and infinite) lists.

Def 5.6 *Every list is written as the sum of its ordered elements from right to left. If* $\overline{v} = \ldots + \theta_2 + \theta_1$ *and* $m \geq 1$*, we define* $\overline{v}_{\geq m} = \ldots + \theta_{m+1} + \theta_m$ *to be the left part of* $\overline{v}$ *starting at* m*, and* $\overline{v}_{<m} = \theta_{m-1} + \theta_{m-2} + \ldots + \theta_1$ *to be the right part of* $\overline{v}$ *ending before* m*. Note that* $\overline{v}_{\geq m} = tl^{m-1}(\overline{v})$*,* $\overline{v}_{<1}$ *is the empty list and* $\overline{v}_{<2} = hd(\overline{v})$*.*

Ex 5.7 $\mathcal{F} = \ldots + x_2 + x_1$, $\updownarrow = \ldots + x'' + x'$, $\mathcal{F}_{\geq m} = \ldots x_{m+1} + x_m$ and $\mathcal{F}_{<m} = x_{m-1} + \ldots + x_1$.

Def 5.8 *We take* $\psi \notin \Theta$ *to be a special symbol whose meaning will be clear below. We write* ψ^1 *as* ψ *and* ψ^0 *as the empty string* $\emptyset$*.* ψ^n *will be* $\underbrace{\psi + \cdots + \psi}_{n}$*.*

- *For a set* $\mathcal{A}$*,* $\mathcal{L}(\mathcal{A}) = \{B; B$ *is a finite list of distinct elements of* $\mathcal{A}\}$*.*
- $\mathcal{L}_\infty(\overline{v}) = \{\overline{v}_{\geq i}; i \in \mathbb{P}\}$*,* $\mathcal{L}_{sp} = \{\mathcal{F}_{\geq m} + \overline{v}; m \in \mathbb{P}, \overline{v} \in \mathcal{L}(\Theta \cup \{\psi\})\}$*,*
  $\mathcal{L}^{-1}(\Theta) = \{\overline{v}; \overline{v} \in \mathcal{L}_{sp} \wedge \overline{v}$ *is* ψ*-free}, and* $\mathcal{L}_\psi = \mathcal{L}_{sp} \cup \mathcal{L}(\Theta \cup \{\psi\})$
- $\forall \overline{v} \in \mathcal{L}(\Theta \cup \{\psi\}), \theta \in \Theta$*, let* $||\emptyset|| = 0, ||\overline{v} + \psi|| = ||\overline{v}|| - 1$ *and* $||\overline{v} + \theta|| = ||\overline{v}|| + 1$*.*
- *For a segment* $\overline{s}$*, let* $sl(\emptyset) = \emptyset$*,* $sl((t_1\delta)\overline{s'}) = sl(\overline{s'})$ *and* $sl((t\lambda_\theta)\overline{s'}) = \theta + sl(\overline{s'})$*.*

Note that $\updownarrow \notin \mathcal{L}(\updownarrow)$ and that $\emptyset \in \mathcal{L}(\mathcal{A})$ for every set $\mathcal{A}$. We write $|\overline{v}|$ for the length of $\overline{v}$.

Lem 5.9 *For all* $\overline{v} \in \mathcal{L}(\Theta \cup \{\psi\})$*,* $||\overline{v}|| \leq |\overline{v}|$*. Moreover, if* $\overline{v} \in \mathcal{L}(\Theta)$ *then* $||\overline{v}|| = |\overline{v}|$*.*

Def 5.10 *(comp) For all* $\overline{v} \in \mathcal{L}_\psi, \theta \in \Theta, n \in \mathbb{P}$*:*

$$\begin{aligned}
comp_1(\overline{v} + \theta) &=_{df} \theta \\
comp_{n+1}(\overline{v} + \theta) &=_{df} comp_n(\overline{v}) \\
comp_n(\overline{v} + \theta + \psi^{i+1}) &=_{df} comp_n(\overline{v} + \psi^i), \quad i \in \mathbb{N}
\end{aligned}$$

The idea of *comp* is to select the appropriate named variable, given a list of (different) named variables. We write $comp_n(\overline{v})\downarrow$, when $comp_n(\overline{v})$ is defined.

Lem 5.11 *For all* $\overline{v} \in \mathcal{L}(\Theta \cup \{\psi\}), n \in \mathbb{P}$*, if* $n \leq ||\overline{v}||$ *then* $comp_n(\overline{v})\downarrow \wedge comp_n(\overline{v}) \in \overline{v}$*.*
Proof: *By induction on* $|\overline{v}|$ *noting that if* $||\overline{v}|| \geq 1$ *then* $\exists \theta \in \Theta$ *such that* $\theta \in \overline{v}$*.* □

Cor 5.12 *For all* $\overline{v} \in \mathcal{L}(\Theta), n \in \mathbb{P}$*, if* $n \leq |\overline{v}|$ *then* $comp_n(\overline{v})\downarrow \wedge comp_n(\overline{v}) \in \overline{v}$*.*
Proof: *Obvious, using lemmas 5.9 and 5.11.* □

Lem 5.13 *For all* $\overline{v} \in \mathcal{L}_{sp}, n \in \mathbb{P}, i \in \mathbb{N}$*, we have* $comp_n(\overline{v})\downarrow \wedge comp_n(\overline{v}) \in \overline{v}$*. Moreover,* $comp_n(\overline{v} + \psi^i) = comp_{n+i}(\overline{v})$*.*
Proof: *The first is by induction on* n*. The second is easy.* □

Note that the only case where $comp_n(\overline{v})$ is undefined is when $n > ||\overline{v}||$.

Lem 5.14 *For all* $\overline{v'} \in \mathcal{L}_{sp}, \overline{v} \in \mathcal{L}(\Theta \cup \{\psi\}), \theta \in \Theta, n \in \mathbb{P}$*, and* $i \in \mathbb{N}$*, we have:*

1. *If* $n > ||\overline{v}|| \geq 0$ *then* $comp_n(\overline{v'} + \overline{v}) \equiv comp_{n-||\overline{v}||}(\overline{v'})$*.*
2. *If* $n > ||\overline{v}|| \geq 0$ *then* $comp_n(\overline{v'} + \psi^i + \overline{v}) \equiv comp_{n+i}(\overline{v'} + \overline{v})$*.*
3. *If* $n \leq ||\overline{v}||$ *then* $comp_n(\overline{v'} + \overline{v}) \equiv comp_n(\overline{v})$*.*
4. $comp_n(\overline{v'} + \theta + \psi + \overline{v}) \equiv comp_n(\overline{v'} + \overline{v})$*.*

Proof: *1 and 3, by induction on* $|\overline{v}|$ *using Lem 5.13. 2, using Lem 5.13 and 1. For 4:*
Case $n \leq ||\overline{v}||$ *or* $n > ||\overline{v}|| \geq 0$*, use the definition of comp and cases 1+3 above.*
Case $n > ||\overline{v}||$ *and* $||\overline{v}|| < 0$ *then by induction on* $|\overline{v}|$*.* □

5.2 The inverse function e

Def 5.15 *(e) Let* $t, t_1, t_2 \in \mathcal{B}^{\lambda\delta}$, $\overline{s}$ *be a segment of* $\overline{\Lambda}$ *consisting of items of the form* (λ_X) *for* $X \in \updownarrow, l \in \mathcal{L}_\infty(\updownarrow), j \in I\!\!P, v \in \Xi, X \in \updownarrow$. e *takes* $\Omega_{\lambda\delta}$*-terms into terms in* $\overline{\Lambda}$ *as follows:*

$$
\begin{array}{lll}
e(t) & =_{df} & c(t, \emptyset, \updownarrow) \\
c(v, \overline{s}, l) & =_{df} & d(v, \overline{s}) \\
c((t_1\delta)t_2, \overline{s}, l) & =_{df} & (c(t_1, \overline{s}, l)\delta)c(t_2, \overline{s}, tl^{nl(t_1)}(l)) \\
c((t_1\lambda)t_2, \overline{s}, l) & =_{df} & (c(t_1, \overline{s}, l)\lambda_{hd^{1+nl(t_1)}(l)})c(t_2, \overline{s}(\lambda_{hd^{1+nl(t_1)}(l)}), tl^{1+nl(t_1)}(l)) \\
d(j, \emptyset) & =_{df} & x_j \\
d(\varepsilon, \overline{s}) & =_{df} & \varepsilon \\
d(1, \overline{s}(\lambda_X)) & =_{df} & X \\
d(n, \overline{s}(\lambda_X)) & =_{df} & d(n-1, \overline{s}) \text{ if } n > 1
\end{array}
$$

d associates with each de Bruijn's index, the right variable in $\mathcal{F} \cup \updownarrow$ which should replace it.

Lem 5.16 *e is well defined and* $b \circ \tau \circ e(t) \equiv t$ *for any* $t \in \mathcal{B}^{\lambda\delta}$

Ex 5.17

$$
\begin{array}{ll}
e(((2\lambda)2\lambda)1) & \equiv c(((2\lambda)2\lambda)1, \emptyset, \updownarrow) \\
& \equiv (c((2\lambda)2, \emptyset, \updownarrow)\lambda_{x''})c(1, (\lambda_{x''}), \{x''', x^{iv}, \ldots\}) \\
& \equiv ((c(2, \emptyset, \updownarrow)\lambda_{x'})c(2, (\lambda_{x'}), \{x'', x''', \ldots\})\lambda_{x''})d(1, (\lambda_{x''})) \\
& \equiv ((d(2, \emptyset)\lambda_{x'})d(2, (\lambda_{x'}))\lambda_{x''})x'' \\
& \equiv ((x_2\lambda_{x'})d(1, \emptyset)\lambda_{x''})x'' \\
& \equiv ((x_2\lambda_{x'})x_1\lambda_{x''})x''
\end{array}
$$

(The first λ becomes $\lambda_{x''}$ and not $\lambda_{x'}$, as there is one λ in $(2\lambda)2$; i.e. $nl((2\lambda)2) = 1$, so $\lambda_{hd^{1+nl((2\lambda)2)}(\updownarrow)} = \lambda_{hd^2(\updownarrow)} = \lambda_{x''}$.) This $\overline{\Lambda}$-term is replaced by $((x_2\lambda_{x_3})x_1\lambda_{x_4})x_4$ in Λ.

Ex 5.18

$$
\begin{array}{ll}
e((\lambda)(1\lambda)(1\delta)3) & \equiv c((\lambda)(1\lambda)(1\delta)3, \emptyset, \updownarrow) \\
& \equiv (c(\varepsilon, \emptyset, \updownarrow)\lambda_{x'})c((1\lambda)(1\delta)3, (\lambda_{x'}), \{x'', x''', \ldots\}) \\
& \equiv (d(\varepsilon, \emptyset)\lambda_{x'})(c(1, (\lambda_{x'}), \{x'', x''', \ldots\})\lambda_{x''})c((1\delta)3, (\lambda_{x'})(\lambda_{x''}), \{x''', \ldots\}) \\
& \equiv (\varepsilon\lambda_{x'})(d(1, (\lambda_{x'}))\lambda_{x''})(c(1, (\lambda_{x'})(\lambda_{x''}), \{x''', \ldots\})\delta)c(3, (\lambda_{x'})(\lambda_{x''}), \{x''', \ldots\}) \\
& \equiv (\varepsilon\lambda_{x'})(x'\lambda_{x''})(d(1, (\lambda_{x'})(\lambda_{x''}))\delta)d(3, (\lambda_{x'})(\lambda_{x''})) \\
& \equiv (\lambda_{x'})(x'\lambda_{x''})(x''\delta)d(2, (\lambda_{x'})) \\
& \equiv (\lambda_{x'})(x'\lambda_{x''})(x''\delta)d(1, \emptyset) \\
& \equiv (\lambda_{x'})(x'\lambda_{x''})(x''\delta)x_1
\end{array}
$$

Finally, we replace x' and x'' of $\updownarrow$ by x_2 and x_3 respectively obtaining $(\lambda_{x_2})(x_2\lambda_{x_3})(x_3\delta)x_1$.

e does not take into account φ-, σ- and μ-items. It is difficult to provide the translation of φ-items without watching what happens in the lists $\mathcal{F}$ and $\updownarrow$. For example:

Ex 5.19 $(\varphi^{(1,2)})(1\delta)(2\lambda)3$ of $\mathcal{B}$ should be: $(x_1\delta)(x_4\lambda_{x'})x_4$ in $\overline{\Lambda}$ and $(x_1\delta)(x_4\lambda_{x_5})x_4$ in Λ. Due to $(\varphi^{(1,2)})$, we use $\mathcal{F}'$ rather than $\mathcal{F}$ where $\mathcal{F}' = \ldots x_5 + x_4 + x_1$. I.e. the x_2 and x_3 disappear.

5.3 The semantics of $\mathcal{B}$-terms: an initial account

We provide the semantics using lists of variables $\overline{v}$ and $\overline{v'}$ so that $[\![\overline{v};\overline{v'};t]\!]$ finds the meaning of $t \in \mathcal{B}$ using $\overline{v}$ and $\overline{v'}$ to give names to the free and bound variables in t respectively. Moreover, $\overline{v} \cap \overline{v'}$ is taken to be $\emptyset$ in order to avoid binding any free variable. If we were to determine the semantics of $\mathcal{B}^{\lambda\delta}$ only, then it is enough to take $\overline{v} \in \mathcal{L}(\updownarrow)$. With φ however, we need $\overline{v} \in \mathcal{L}_{sp}$. We start first with only finite lists in $\mathcal{L}(\updownarrow)$ and give the semantics of $\mathcal{B}^{\lambda\delta}$ as follows:

Def 5.20 *(λ-, δ-semantics)* $\forall t_1, t_2 \in \mathcal{B}^{\lambda\delta}, \overline{v} \in \mathcal{L}(\updownarrow), \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, n \in \Xi$,

$$\begin{array}{lll}
[\![\overline{v};\overline{v'};(t_1\lambda)t_2]\!] & =_{df} & ([\![\overline{v};\overline{v'};t_1]\!]\lambda_X)[\![\overline{v} + X;\overline{v'}_{\geq i+1};t_2]\!] \textit{ for } i = nl(t_1)+1, X = hd^i(\overline{v'}) \\
[\![\overline{v};\overline{v'};(t_1\delta)t_2]\!] & =_{df} & ([\![\overline{v};\overline{v'};t_1]\!]\delta)[\![\overline{v};\overline{v'}_{\geq i};t_2]\!] \textit{ for } i = nl(t_1)+1 \\
[\![\overline{v};\overline{v'};n]\!] & =_{df} & \begin{cases} comp_n(\overline{v}) & \textit{if } n \leq |\overline{v}| \\ x_{n-|\overline{v}|} & n > |\overline{v}| \\ \varepsilon & \textit{if } n = \varepsilon \end{cases}
\end{array}$$

Ex 5.21 *(see Example 5.17)*

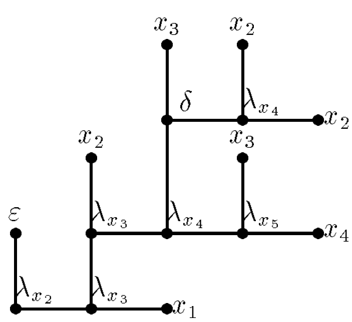
$$\begin{array}{ll}
[\![\emptyset;\updownarrow;((2\lambda)2\lambda)1]\!] & \equiv \\
([\![\emptyset;\updownarrow;(2\lambda)2]\!]\lambda_{x''})[\![x'';\updownarrow_{\geq 3};1]\!] & \equiv \\
(([\![\emptyset;\updownarrow;2]\!]\lambda_{x'})[\![x';\updownarrow_{\geq 2};2]\!]\lambda_{x''})comp_1(x'') & \equiv \\
((x_{2-|\emptyset|}\lambda_{x'})x_{2-|x'|}\lambda_{x''})x'' & \equiv \\
((x_2\lambda_{x'})x_1\lambda_{x''})x'' & \equiv
\end{array}$$

Lem 5.22 *For any* $\overline{v} \in \mathcal{L}(\updownarrow), \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, t \in \mathcal{B}^{\lambda\delta}, FV([\![\overline{v};\overline{v'};t]\!]) \subseteq \overline{v} \cup \mathcal{F}$.
Proof: *By induction on t, recalling that ε is neither free nor bound.* □

Lem 5.23 $\forall \overline{v} \in \mathcal{L}(\updownarrow), \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, t \in \mathcal{B}^{\lambda\delta}$, $[\![\overline{v};\overline{v'};t]\!]$ *is well-defined + unique in* $\overline{\Lambda}$.
Proof: *By induction on $t \in \mathcal{B}^{\lambda\delta}$ using Cor 5.12.* □

Lem 5.24 *For all* $t \in \mathcal{B}^{\lambda\delta}, e(t) \equiv [\![\emptyset;\updownarrow;t]\!]$.
Proof: *Show by induction on t* $\forall t \in \mathcal{B}^{\lambda\delta}, \overline{s} \in \overline{\Lambda}$ *and* $\overline{v} \in \mathcal{L}_\infty(\updownarrow), c(t,\overline{s},\overline{v}) \equiv [\![sl(\overline{s});\overline{v};t]\!]$. □

Ex 5.25 Let $t \equiv (\varepsilon\lambda)((1\lambda)((1\delta)(2\lambda)3\lambda)(2\lambda)2\lambda)3$. Now, the reader can check that:
$e(t) \equiv [\![\emptyset;\updownarrow;t]\!] \equiv (\varepsilon\lambda_{x'})((x'\lambda_{x''})((x''\delta)(x'\lambda_{x'''})x'\lambda_{x'''})(x''\lambda_{x''''})x'''\lambda_{x''})x_1$.
Furthermore, $\tau(e(t)) \equiv (\varepsilon\lambda_{x_2})((x_2\lambda_{x_3})((x_3\delta)(x_2\lambda_{x_4})x_2\lambda_{x_4})(x_3\lambda_{x_5})x_4\lambda_{x_3})x_1$ (see Figure 6).



$t \equiv (\varepsilon\lambda_{x_2})((x_2\lambda_{x_3})((x_3\delta)(x_2\lambda_{x_4})x_2\lambda_{x_4})(x_3\lambda_{x_5})x_4\lambda_{x_3})x_1 \equiv (\varepsilon\lambda)((1\lambda)((1\delta)(2\lambda)3\lambda)(2\lambda)2\lambda)3$

Figure 6: The tree of $\tau(e(t))$

5.4 Extending the initial account

$(\varphi^{(k,i)})t$ means: add i to all free variables $> k$, occurring in t. When we look for $\llbracket \overline{v}; \overline{v'}; (\varphi^{(k,i)})t \rrbracket$, all the variables in $t \leq k$ take the same value as in $\llbracket \overline{v}; \overline{v'}; t \rrbracket$. Those variables $> k$ must not take the values they would have taken in $\llbracket \overline{v}; \overline{v'}; t \rrbracket$. Rather, looking for their corresponding variables in $\overline{v}$, we have to shift still i positions to the left. I.e. if the index is n, where $n > k$ then the variable corresponding to n is not the n^{th} variable from right to left in $\overline{v}$. Rather, it is the $(n+i)^{th}$ variable from the right. For example:

$$\llbracket x''''x'''x''x'; \updownarrow_{\geq 5}; (\varphi^{(1,2)})(1\delta)2 \rrbracket \equiv (x'\delta)x''''$$

For this, we allow a special symbol ψ to become an element of $\overline{v}$. The operational meaning of ψ is: on going left, delete the first named variable. Such a ψ, will not only be used to erase variables but will also say which free variable in $\mathcal{F}$ correponds to the variable in hand.

Ex 5.26 The idea is that:

1. If $|\overline{v}| \geq k+i, \overline{v} = \overline{v_1} \mathbin{+\!\!+} \overline{v_2}$ and $|\overline{v_2}| = k$, then $\llbracket \overline{v}; \overline{v'}; (\varphi^{(k,i)})t \rrbracket = \llbracket \overline{v_1} \mathbin{+\!\!+} \psi^i \mathbin{+\!\!+} \overline{v_2}; \overline{v'}; t \rrbracket$. Hence for $\llbracket x''''x'''x''x'; \updownarrow_{\geq 5}; (\varphi^{(1,2)})2 \rrbracket$, we need $\llbracket x''''x'''x'' \mathbin{+\!\!+} \psi^2 \mathbin{+\!\!+} x'; \updownarrow_{\geq 5}; 2 \rrbracket$. This evaluates to $\llbracket x''''x'''x'' \mathbin{+\!\!+} \psi^2; \updownarrow_{\geq 5}; 1 \rrbracket$. The presence of ψ^2 means ignore $x'''x''$. Therefore the result reduces to $\llbracket x''''; \updownarrow_{\geq 5}; 1 \rrbracket$ which is x''''.

2. For every $n \in I\!N, m \in I\!P, \llbracket \overline{v} \mathbin{+\!\!+} \psi^n; \overline{v'}; m \rrbracket = \llbracket \overline{v}; \overline{v'}; n+m \rrbracket$ and $\llbracket \psi^n; \overline{v'}; m \rrbracket = x_{n+m}$.

Looking at the first part of Example 5.26, we see that we need to have $\overline{v} = \overline{v_1} \mathbin{+\!\!+} \overline{v_2}$ where $|\overline{v_2}| = k$. In other words, we have to go through the list $\overline{v}$ from right to left until we pass the k^{th} element. In order to accommodate this, we introduce an extra argument in the semantic meaning of φ-terms. We will give an example which explains the point even though it is ahead of its time in the section. We believe however, that the reader can still follow it, once point 2 of Example 5.26 is remembered.

Ex 5.27 Notice how we save x' to use it later on:

$$\begin{array}{ll}
\llbracket x''x'; \updownarrow_{\geq 3}; (\varphi^{(1,2)})(1\delta)2 \rrbracket & \equiv \\
\llbracket x''; x'; \updownarrow_{\geq 3}; (\varphi^{(1,2)})(1\delta)2 \rrbracket & \equiv \\
\llbracket x'' \mathbin{+\!\!+} \psi^2 \mathbin{+\!\!+} x'; \updownarrow_{\geq 3}; (1\delta)2 \rrbracket & \equiv \\
(\llbracket x'' \mathbin{+\!\!+} \psi^2 \mathbin{+\!\!+} x'; \updownarrow_{\geq 3}; 1 \rrbracket \delta) \llbracket x'' \mathbin{+\!\!+} \psi^2 \mathbin{+\!\!+} x'; \updownarrow_{\geq 3}; 2 \rrbracket & \equiv \\
(x'\delta) \llbracket x'' \mathbin{+\!\!+} \psi^2; \updownarrow_{\geq 3}; 1 \rrbracket & \equiv \\
(x'\delta) \llbracket x''; \updownarrow_{\geq 3}; 3 \rrbracket & \equiv (x'\delta)x_2
\end{array}$$

We extend lists from elements of $\mathcal{L}(\updownarrow)$ (as in Def 5.20) to elements of $\mathcal{L}_{sp}$. Now our lists include ψ's, bound and free variables, and are denumerably infinite. Now, here is $\llbracket \cdot;\cdot;\cdot \rrbracket_e$, the extended definition of the semantics of λ- and δ-items.

Def 5.28 *(Extended λ- and δ-semantics)* $\llbracket \cdot;\cdot;\cdot \rrbracket_e : \mathcal{L}_{sp} \times \mathcal{L}_\infty(\updownarrow) \times \mathcal{B}^{\lambda\delta\sigma\varphi} \mapsto \overline{\Lambda}$:
$\forall t_1, t_2 \in \mathcal{B}^{\lambda\delta}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, n \in I\!P$,

$$\begin{array}{ll}
\llbracket \overline{v}; \overline{v'}; (t_1\lambda)t_2 \rrbracket_e & =_{df} (\llbracket \overline{v}; \overline{v'}; t_1 \rrbracket_e \lambda_X) \llbracket \overline{v} \mathbin{+\!\!+} X; \overline{v'}_{\geq i+1}; t_2 \rrbracket_e \textit{ for } i = nl(t_1)+1, X = hd^i(\overline{v'}) \\
\llbracket \overline{v}; \overline{v'}; (t_1\delta)t_2 \rrbracket_e & =_{df} (\llbracket \overline{v}; \overline{v'}; t_1 \rrbracket_e \delta) \llbracket \overline{v}; \overline{v'}_{\geq i}; t_2 \rrbracket_e \textit{ for } i = nl(t_1)+1 \\
\llbracket \overline{v}; \overline{v'}; n \rrbracket_e & =_{df} comp_n(\overline{v}) \\
\llbracket \overline{v}; \overline{v'}; \varepsilon \rrbracket_e & =_{df} \varepsilon
\end{array}$$

Lem 5.29 *Let* $\overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow), (\overline{v} +\!\!+ \theta) \cap \overline{v'} = \emptyset, \theta \in \Theta, n, m \in \mathbb{P}$ *and* $k \in \mathbb{N}$.

1.	$\llbracket \overline{v} +\!\!+ \theta; \overline{v'}; 1 \rrbracket_e$	$\equiv$	θ
2.	$\llbracket \overline{v}; \overline{v'}; n+k \rrbracket_e$	$\equiv$	$\llbracket \overline{v} +\!\!+ \psi^k; \overline{v'}; n \rrbracket_e$
3.	$\llbracket \overline{v} +\!\!+ \theta; \overline{v'}; n+1 \rrbracket_e$	$\equiv$	$\llbracket \overline{v}; \overline{v'}; n \rrbracket_e$
4.	$\llbracket \mathcal{F}_{\geq m} +\!\!+ \psi^k; \overline{v'}; n \rrbracket_e$	$\equiv$	$x_{n+k+m-1}$
5.	$\llbracket \overline{v}; \overline{v'}; n \rrbracket_e$	$\in$	$\overline{v}$
6.	*If* $n \neq m$	*then*	$\llbracket \overline{v}; \overline{v'}; n \rrbracket_e \not\equiv \llbracket \overline{v}; \overline{v'}; m \rrbracket_e$

Proof: *Easy, using Lem 5.13 and the definition of comp.* □

Lem 5.30 $\forall \overline{v'} \in \mathcal{L}_{sp}, \overline{v} \in \mathcal{L}(\Theta \cup \{\psi\}), \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v'} +\!\!+ \overline{v}) \cap \overline{v''} = \emptyset, \theta \in \Theta\ n, i \in \mathbb{P}$:

1. *If* $n > ||\overline{v}|| \geq 0$ *then* $\llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v'}; \overline{v''}; n - ||\overline{v}|| \rrbracket_e$

2. *If* $n > ||\overline{v}|| \geq 0$ *then* $\llbracket \overline{v'} +\!\!+ \psi^i +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n+i \rrbracket_e$.

3. *If* $n \leq ||\overline{v}||$ *then* $\llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv comp_n(\overline{v})$

4. $\llbracket \overline{v'} +\!\!+ \theta +\!\!+ \psi +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e$

Proof: *This follows from Lem 5.14.* □

Cor 5.31 $\forall \overline{v'} \in \mathcal{L}_{sp},\ \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v'} +\!\!+ \overline{v}) \cap \overline{v''} = \emptyset,\ n, i \in \mathbb{P},\ \overline{v} \in \mathcal{L}(\Theta)$:

1. *If* $n > |\overline{v}|$ *then* $\llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v'}; \overline{v''}; n - |\overline{v}| \rrbracket_e$

2. *If* $n > |\overline{v}|$ *then* $\llbracket \overline{v'} +\!\!+ \psi^i +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n+i \rrbracket_e$.

3. *If* $n \leq |\overline{v}|$ *then* $\llbracket \overline{v'} +\!\!+ \overline{v}; \overline{v''}; n \rrbracket_e \equiv comp_n(\overline{v})$

Proof: *Obvious by lemmas 5.9 and 5.30.* □

Rem 5.32 Note that if $\overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}(\Theta \cup \{\psi\}), \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v'} +\!\!+ \overline{v}) \cap \overline{v''} = \emptyset, n, i \in \mathbb{P}, ||\overline{v'}|| < 0$, then even though $n > ||\overline{v'}||$, it is not necessarily the case that:

1. $\llbracket \overline{v} +\!\!+ \overline{v'}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v}; \overline{v''}; n - ||\overline{v'}|| \rrbracket_e$

2. $\llbracket \overline{v} +\!\!+ \psi^i +\!\!+ \overline{v'}; \overline{v''}; n \rrbracket_e \equiv \llbracket \overline{v} +\!\!+ \overline{v'}; \overline{v''}; n+i \rrbracket_e$

For example, $\llbracket \mathcal{F} +\!\!+ \psi^5 x'; \updownarrow_{\geq 2}; 1 \rrbracket_e \equiv x'$ whereas $\llbracket \mathcal{F}; \updownarrow_{\geq 2}; 1 - ||\psi^5 x'|| \rrbracket_e \equiv \llbracket \mathcal{F}; \updownarrow_{\geq 2}; 5 \rrbracket_e \equiv x_5$.

Lem 5.33 *For all* $\overline{v} \in \mathcal{L}(\updownarrow), \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, t \in \mathcal{B}^{\lambda\delta},\ \llbracket \overline{v}; \overline{v'}; t \rrbracket \equiv \llbracket \mathcal{F} +\!\!+ \overline{v}; \overline{v'}; t \rrbracket_e$.
Proof: *Show* $\forall n \in \mathbb{P} \cup \{\varepsilon\}$: $\llbracket \overline{v}; \overline{v'}; n \rrbracket \equiv \llbracket \mathcal{F} +\!\!+ \overline{v}; \overline{v'}; n \rrbracket_e$ *and then use induction on* t. □

5.5 The semantics of σ- and φ-terms

Def 5.34 *(σ-semantics)* $\forall t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, i \in \mathbb{P}$:
$\llbracket \overline{v}; \overline{v'}; (t_1 \sigma^{(i)}) t_2 \rrbracket_e =_{df} \llbracket \overline{v}; \overline{v'}; t_2 \rrbracket_e [\llbracket \overline{v}; \overline{v'}; i \rrbracket_e := \llbracket \overline{v}; \overline{v'}_{\geq 1+nl(t_2)}; t_1 \rrbracket_e]'$

Def 5.35 *(φ-semantics)*
$\forall t \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}(\Theta), \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v} +\!\!\!+ \theta) \cap \overline{v''} = \emptyset, \theta \in \Theta, i \in I\!\!P, k \in I\!\!N$:

$$\begin{array}{ll}
[\![\overline{v}; \overline{v''}; (\varphi^{(k,i)})t]\!]_e & =_{df} [\![\overline{v}; \emptyset; \overline{v''}; (\varphi^{(k,i)})t]\!] \\
[\![\overline{v}; \overline{v'}; \overline{v''}; (\varphi^{(0,i)})t]\!] & =_{df} [\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; t]\!]_e \\
[\![\overline{v} +\!\!\!+ \theta; \overline{v'}; \overline{v''}; (\varphi^{(k+1,i)}t]\!] & =_{df} [\![\overline{v}; \theta +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})t]\!] \\
[\![\overline{v} +\!\!\!+ \theta +\!\!\!+ \psi^{k+1}; \overline{v'}; \overline{v''}; t]\!] & =_{df} [\![\overline{v} +\!\!\!+ \psi^k; \overline{v'}; \overline{v''}; t]\!]
\end{array}$$

Note here that $\overline{v''}$ does not play a role because we do not have bound variables that we are trying to replace by variable names. What the $\overline{v'}$ does however is to save the first k variables of $\overline{v}$ which are actually the variables in t which should not be updated because they are $\leq k$. Once the first k variables of $\overline{v}$ have been saved in $\overline{v'}$, we remove the first i variables from the resulting $\overline{v}$. Hence in the end, we get the correct list from which we find the meaning of t.

Ex 5.36

$$\begin{array}{lll}
1. & [\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; (\varphi^{(2,3)})3]\!]_e & = [\![\mathcal{F} +\!\!\!+ x'; \emptyset; \updownarrow_{\geq 2}; (\varphi^{(2,3)})3]\!] \\
 & & = [\![\mathcal{F}; x'; \updownarrow_{\geq 2}; (\varphi^{(1,3)})3]\!] \\
 & & = [\![\mathcal{F}_{\geq 2}; x_1 +\!\!\!+ x'; \updownarrow_{\geq 2}; (\varphi^{(0,3)})3]\!] \\
 & & = [\![\mathcal{F}_{\geq 2} +\!\!\!+ \psi^3 +\!\!\!+ x_1 +\!\!\!+ x'; \updownarrow_{\geq 2}; 3]\!]_e = x_5 \\
2. & [\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; (\varphi^{(2,3)})1]\!]_e & = x' \\
3. & [\![\mathcal{F}; \updownarrow_{\geq 2}; (\varphi^{(1,2)})(\varphi^{(0,1)})1]\!] & = x_4
\end{array}$$

Now the following lemma is basic about φ-items.

Lem 5.37 *Let* $t \in \mathcal{B}^{\lambda\delta\sigma\varphi}$, $\overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}(\Theta), \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v} +\!\!\!+ \overline{v'}) \cap \overline{v''} = \emptyset$, $i \in I\!\!P$.
$[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(|\overline{v'}|,i)})t]\!]_e \equiv [\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; t]\!]_e$.
Proof: *Easy. First prove by induction on* $|\overline{v'}|$ *that if* $\overline{v} \in \mathcal{L}_{sp}, \overline{v'}, \overline{v_1} \in \mathcal{L}(\Theta)$ *such that* $(\overline{v} +\!\!\!+ \overline{v'} +\!\!\!+ \overline{v_1}) \cap \overline{v''} = \emptyset$ *then* $[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v_1}; \overline{v''}; (\varphi^{(|\overline{v'}|,i)})t]\!] \equiv [\![\overline{v}; \overline{v'} +\!\!\!+ \overline{v_1}; \overline{v''}; (\varphi^{(0,i)})t]\!]$ □

The following lemma opens the road to working with lists which do not contain ψ.

Lem 5.38 $\forall \overline{v'} \in \mathcal{L}_{sp}, \overline{v} \in \mathcal{L}(\Theta \cup \{\psi\}), \overline{v_1} \in \mathcal{L}_\infty(\updownarrow), (\overline{v'} +\!\!\!+ \theta +\!\!\!+ \overline{v}) \cap \overline{v_1} = \emptyset, \theta \in \Theta$, $n \in I\!\!P$:
$[\![\overline{v'} +\!\!\!+ \theta +\!\!\!+ \psi +\!\!\!+ \overline{v}; \overline{v_1}; t]\!]_e \equiv [\![\overline{v'} +\!\!\!+ \overline{v}; \overline{v_1}; t]\!]_e$
Proof: *By nested induction. We prove by induction on* t *that* $IH_1(t)$ *holds where* $IH_1(t)$ *is:* $[\![\overline{v'} +\!\!\!+ \theta +\!\!\!+ \psi +\!\!\!+ \overline{v}; \overline{v_1}; t]\!]_e \equiv [\![\overline{v'} +\!\!\!+ \overline{v}; \overline{v_1}; t]\!]_e$

- *If* $t = n$, *use case 4 of Lem 5.30.*
- *If* $(t_1\delta)t_2$ *or* $(t_1\lambda)t_2$ *or* $(t_1\sigma^{(i)})t_2$ *where* $IH_1(t_1)$ *and* $IH_1(t_2)$ *hold, easy.*
- *If* $(\varphi^{(k,i)})t$ *and* $IH_1(t)$. *Prove* $IH_2(k)$ *by induction on* k *where* $IH_2(k)$, $\forall \overline{v''} \in \mathcal{L}(\Theta)$ *is:*
  $[\![\overline{v'} +\!\!\!+ \theta +\!\!\!+ \psi +\!\!\!+ \overline{v}; \overline{v''}; \overline{v_1}; (\varphi^{(k,i)})t]\!]_e \equiv [\![\overline{v'} +\!\!\!+ \overline{v}; \overline{v''}; \overline{v_1}; (\varphi^{(k,i)})t]\!]_e$
  - *If* $k = 0$, *use* $IH_1(t)$.
  - *Assume* $IH_2(k)$. *Prove by induction on* $|\overline{v}|$ *that* $IH_3(\overline{v})$ *holds where* $IH_3(\overline{v})$ *is* $[\![\overline{v'} +\!\!\!+ \theta +\!\!\!+ \psi +\!\!\!+ \overline{v}; \overline{v''}; \overline{v_1}; (\varphi^{(k+1,i)})t]\!]_e \equiv [\![\overline{v'} +\!\!\!+ \overline{v}; \overline{v''}; \overline{v_1}; (\varphi^{(k+1,i)})t]\!]_e$:
    * *If* $|\overline{v}| = 0$, *use Def 5.35.*
    * *If* $\overline{v} +\!\!\!+ \theta$ *where* $\theta \in \Theta$ *and* $IH_3(\overline{v})$ *holds, use Def 5.35 and* $IH_2(k)$.
    * *If* $\overline{v} +\!\!\!+ \theta +\!\!\!+ \psi^j$, $\theta \in \Theta, j \in I\!\!P$ *and* $IH_3(\overline{v} +\!\!\!+ \psi^{j-1})$, *use Def 5.35 and* $IH_3(\overline{v} +\!\!\!+ \psi^{j-1})$.
    * *Case* ψ^j *where* $j \in I\!\!P$, *use Def 5.35.* □

The following lemma is very important. It says that all the ψ's can be removed from lists.

Lem 5.39 *For all* $\overline{v} \in \mathcal{L}_{sp}, \exists \overline{v'} \in \mathcal{L}_{sp}$ *which is free for* ψ *such that for all* $t \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v''} \in \mathcal{L}_{\infty}(\updownarrow)$ *such that* $\overline{v} \cap \overline{v''} = \emptyset, [\![\overline{v}; \overline{v''}; t]\!]_e \equiv [\![\overline{v'}; \overline{v''}; t]\!]_e$.

Proof: *We can write* $\overline{v}$ *as* $\overline{v_1} +\!\!+ \theta +\!\!+ \overline{v_2}$ *such that* $\theta \in \Theta, \overline{v_1} \in \mathcal{L}_{sp}, \overline{v_2} \in \mathcal{L}(\Theta \cup \{\psi\})$, $\overline{v_1}$ *is free of* ψ *and* $\overline{v_2}$ *has* ψ *as its leftmost element. Now, the proof is by induction on* $|\overline{v_2}|$ *using Lem 5.38. Note moreover, that* $\overline{v'}$ *is independent of* t. *Hence, we may assume from now on that our start lists do not contain* ψ. □

Finally, we give the translation of any term t of $\mathcal{B}^{\lambda\delta\sigma\varphi}$:

Def 5.40 *(The semantic function) Define* $[\![\cdot]\!] : \mathcal{B}^{\lambda\delta\sigma\varphi} \mapsto \overline{\Lambda}$ *such that* $[\![t]\!] =_{df} [\![\mathcal{F}; \updownarrow; t]\!]_e$

Lem 5.41 $[\![\cdot]\!]$ *is well defined. That is, for all* $t \in \mathcal{B}^{\lambda\delta\sigma\varphi}$, $[\![t]\!]$ *is a unique term in* $\overline{\Lambda}$.

Proof: *By induction on* $t \in \mathcal{B}^{\lambda\delta\sigma\varphi}$. □

Now here is our first lemma towards the correctness of our semantics:

Lem 5.42 *For all* $t \in \mathcal{B}^{\lambda\delta\sigma\varphi}$, *we have:*

1. $BV([\![\overline{v}; \overline{v'}; t]\!]_e) \subset \overline{v'}$ *for every* $\overline{v} \in \mathcal{L}_{sp}$ *and* $\overline{v'} \in \mathcal{L}_{\infty}(\updownarrow)$ *such that* $\overline{v} \cap \overline{v'} = \emptyset$.
2. $FV([\![\overline{v}; \overline{v'}; t]\!]_e) \subset \overline{v}$ *for every* $\overline{v} \in \mathcal{L}_{sp}$ *and* $\overline{v'} \in \mathcal{L}_{\infty}(\updownarrow)$ *such that* $\overline{v} \cap \overline{v'} = \emptyset$.
3. $BV([\![t]\!]) \subset \updownarrow$ *and* $FV([\![t]\!]) \subset \mathcal{F}$.

Proof: *1 and 2 are by induction on* t. *3 follows from 1 and 2.* □

Hence, a term $[\![t]\!]$ in $\overline{\Lambda}$ can be translated using Def 5.4 to a term in Λ.

Ex 5.43 (Note that we sometimes combine many steps in one.)

$$\begin{aligned}
[\![(\varphi^{(2,1)})(1\delta)(2\lambda)3]\!] &\equiv [\![\mathcal{F}; \updownarrow; (\varphi^{(2,1)})(1\delta)(2\lambda)3]\!]_e \\
&\equiv [\![\mathcal{F}; \emptyset; \updownarrow; (\varphi^{(2,1)})(1\delta)(2\lambda)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 2}; x_1; \updownarrow; (\varphi^{(1,1)})(1\delta)(2\lambda)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 3}; x_2 +\!\!+ x_1; \updownarrow; (\varphi^{(0,1)})(1\delta)(2\lambda)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 3} +\!\!+ \psi +\!\!+ x_2 +\!\!+ x_1; \updownarrow; (1\delta)(2\lambda)3]\!]_e \equiv (x_1\delta)(x_2\lambda_{x'})x_4
\end{aligned}$$

$$\begin{aligned}
[\![(\varphi^{(2,3)})(\varphi^{(1,2)})(1\delta)(2\delta)3]\!] &\equiv [\![\mathcal{F}; \updownarrow; (\varphi^{(2,3)})(\varphi^{(1,2)})(1\delta)(2\delta)3]\!]_e \\
&\equiv [\![\mathcal{F}_{\geq 2}; x_1; \updownarrow; (\varphi^{(1,3)})(\varphi^{(1,2)})(1\delta)(2\delta)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 3}; x_2 +\!\!+ x_1; \updownarrow; (\varphi^{(0,3)})(\varphi^{(1,2)})(1\delta)(2\delta)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ x_2 +\!\!+ x_1; \updownarrow; (\varphi^{(1,2)})(1\delta)(2\delta)3]\!]_e \\
&\equiv [\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ x_2; x_1; \updownarrow; (\varphi^{(0,2)})(1\delta)(2\delta)3]\!] \\
&\equiv [\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ x_2 +\!\!+ \psi^2 +\!\!+ x_1; \updownarrow; (1\delta)(2\delta)3]\!]_e \\
&\equiv (x_1\delta)([\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ x_2 +\!\!+ \psi^2 +\!\!+ x_1; \updownarrow; 2]\!]_e\delta) \\
&\qquad [\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ x_2 +\!\!+ \psi^2 +\!\!+ x_1; \updownarrow; 3]\!]_e \\
&\equiv (x_1\delta)([\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ \psi; \updownarrow; 1]\!]_e\delta)[\![\mathcal{F}_{\geq 3} +\!\!+ \psi^3 +\!\!+ \psi; \updownarrow; 2]\!]_e \\
&\equiv (x_1\delta)([\![\mathcal{F}_{\geq 7}; \updownarrow; 1]\!]_e\delta)[\![\mathcal{F}_{\geq 7}; \updownarrow; 2]\!]_e \equiv (x_1\delta)(x_7\delta)x_8
\end{aligned}$$

6 The soundness of σ- and φ-reduction

Here, we show that if $t \to t'$ where $\to$ is φ-transition or destruction, or σ-destruction, then $[\![t]\!] \equiv [\![t']\!]$. That is, φ and σ are sound with respect to variable updating and substitution. We show moreover, that if $t \to_\sigma t'$ where $\to$ is σ-generation, then $[\![t]\!] = [\![t']\!]$. That is, σ-generation is a form of β-conversion. Furthermore, σ-transition has α-conversion. That is, if $t \to_\sigma t'$ where $\to_\sigma$ is σ-transition, then $[\![t]\!] =_{\overline{\alpha}} [\![t']\!]$. For this, let us repeat the semantic function:

Def 6.1 *(Semantics of $\mathcal{B}^{\lambda\delta\sigma\varphi}$) $\forall t, t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}(\Theta), \overline{v''} \in \mathcal{L}_\infty(\updownarrow), (\overline{v}+\theta) \cap \overline{v''} = \emptyset, \theta \in \Theta, i, n \in I\!P$ and $k \in I\!N$, we define:*

$$M1.\quad [\![t]\!] =_{df} [\![\mathcal{F}; \updownarrow; t]\!]_e$$

$$
\begin{array}{lll}
M2. & [\![\overline{v}; \overline{v''}; \varepsilon]\!]_e & =_{df} \varepsilon \\
M3. & [\![\overline{v}; \overline{v''}; n]\!]_e & =_{df} comp_n(\overline{v}) \\
M4. & [\![\overline{v}; \overline{v''}; (t_1\lambda)t_2]\!]_e & =_{df} ([\![\overline{v}; \overline{v''}; t_1]\!]_e \lambda_X)[\![\overline{v} +\!\!\!+ X; \overline{v''}_{\geq i+1}; t_2]\!]_e \textit{ for } i = nl(t_1)+1, X = hd^i(\overline{v''}) \\
M5. & [\![\overline{v}; \overline{v''}; (t_1\delta)t_2]\!]_e & =_{df} ([\![\overline{v}; \overline{v''}; t_1]\!]_e \delta)[\![\overline{v}; \overline{v''}_{\geq i}; t_2]\!]_e \textit{ for } i = nl(t_1)+1 \\
M6. & [\![\overline{v}; \overline{v''}; (t_1\sigma^{(i)})t_2]\!]_e & =_{df} [\![\overline{v}; \overline{v''}; t_2]\!]_e[[\![\overline{v}; \overline{v''}; i]\!]_e := [\![\overline{v}; \overline{v''}_{\geq i}; t_1]\!]_e]' \textit{ for } i = nl(t_2)+1 \\
M7. & [\![\overline{v}; \overline{v''}; (\varphi^{(k,i)})t]\!]_e & =_{df} [\![\overline{v}; \emptyset; \overline{v''}; (\varphi^{(k,i)})t]\!]
\end{array}
$$

$$
\begin{array}{lll}
M8. & [\![\overline{v}; \overline{v'}; \overline{v''}; (\varphi^{(0,i)})t_1]\!] & =_{df} [\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; t]\!]_e \\
M9. & [\![\overline{v} +\!\!\!+ \theta; \overline{v'}; \overline{v''}; (\varphi^{(k+1,i)})t_1]\!] & =_{df} [\![\overline{v}; \theta +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})t]\!] \\
M10. & [\![\overline{v} +\!\!\!+ \theta +\!\!\!+ \psi^{k+1}; \overline{v'}; \overline{v''}; t]\!] & =_{df} [\![\overline{v} +\!\!\!+ \psi^k; \overline{v'}; \overline{v''}; t]\!]
\end{array}
$$

Now, the following lemmas inform us about the place of (α) in our system.

Lem 6.2 $\forall n \in I\!P, \overline{v} \in \mathcal{L}_{sp}, \overline{v'}, \overline{v''} \in \mathcal{L}_\infty(\updownarrow),\ \overline{v} \cap \overline{v'} = \overline{v} \cap \overline{v''} = \emptyset \Rightarrow [\![\overline{v}; \overline{v'}; n]\!]_e = [\![\overline{v}; \overline{v''}; n]\!]_e$.

Lem 6.3 $\forall t \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow),\ \overline{v} \cap \overline{v'} = \emptyset \Rightarrow \forall \overline{v''} \in \mathcal{L}_\infty(\overline{v'}),\ [\![\overline{v}; \overline{v'}; t]\!]_e =_{\overline{\alpha}} [\![\overline{v}; \overline{v''}; t]\!]_e$.
Proof: *By induction on t.* □

Now we define the notions of (α-, β-) soundness:

Def 6.4 *Let $\to$ be a reduction rule. We say:*

- *$\to$ is sound if:* $(\forall t, t', \overline{v}, \overline{v'})[t \to t' \Rightarrow [\![\overline{v}; \overline{v'}; t]\!]_e \equiv [\![\overline{v}; \overline{v'}; t']\!]_e]$.
- *$\to$ is α-sound if:* $(\forall t, t', \overline{v}, \overline{v'})[t \to t' \Rightarrow [\![\overline{v}; \overline{v'}; t]\!]_e =_{\overline{\alpha}} [\![\overline{v}; \overline{v'}; t']\!]_e]$.
- *$\to$ is β-sound if:* $(\forall t, t', \overline{v}, \overline{v'})[t \to t' \Rightarrow [\![\overline{v}; \overline{v'}; t]\!]_e =_{\overline{\beta}} [\![\overline{v}; \overline{v'}; t']\!]_e]$.
- *$\to$ is $\alpha\beta$-sound if:* $(\forall t, t', \overline{v}, \overline{v'})[t \to t' \Rightarrow [\![\overline{v}; \overline{v'}; t]\!]_e = [\![\overline{v}; \overline{v'}; t']\!]_e]$.

Lem 6.5 *φ-transition through a δ-item is sound. I.e.,* $\forall t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v_1} \in \mathcal{L}_{sp}, \overline{v''} \in \mathcal{L}_\infty(\updownarrow), \overline{v_1} \cap \overline{v''} = \emptyset, i \in I\!P,\ k \in I\!N$: $[\![\overline{v_1}; \overline{v''}; (\varphi^{(k,i)})(t_1\delta)t_2]\!]_e \equiv [\![\overline{v_1}; \overline{v''}; ((\varphi^{(k,i)})t_1\delta)(\varphi^{(k,i)})t_2]\!]_e$
Proof: *Assume $\overline{v_1}$ ψ-free (Lem 5.39). Assume also $\overline{v_1} = \overline{v} +\!\!\!+ \overline{v'}$ for $|\overline{v'}| = k$.*

$$
\begin{array}{ll}
([\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; ((\varphi^{(k,i)})t_1\delta)(\varphi^{(k,i)})t_2]\!]_e & \equiv^{j=1+nl(t_1)} \\
([\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})t_1]\!]_e\delta)[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}_{\geq j}; (\varphi^{(k,i)})t_2]\!]_e & \equiv^{Lem\ 5.37} \\
([\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; t_1]\!]_e\delta)[\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}_{\geq j}; t_2]\!]_e & \equiv \\
[\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; (t_1\delta)t_2]\!]_e & \equiv^{Lem\ 5.37} \\
[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})(t_1\delta)t_2]\!]_e &
\end{array}
$$
□

Lem 6.6 *φ-transition through a λ-item is sound. I.e.,* $\forall t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v_1} \in \mathcal{L}_{sp}, \overline{v''} \in \mathcal{L}_\infty(\updownarrow), \overline{v_1} \cap \overline{v''} = \emptyset, i \in \mathbb{P},\ k \in \mathbb{N}$: $[\![\overline{v_1}; \overline{v''}; (\varphi^{(k,i)})(t_1\lambda)t_2]\!]_e \equiv [\![\overline{v_1}; \overline{v''}; ((\varphi^{(k,i)})t_1\lambda)(\varphi^{(k+1,i)})t_2]\!]_e$

Proof: *Similar to Lem 6.5, asume $\overline{v_1}$ is ψ-free and $\overline{v_1} = \overline{v} + \!\!\!+ \overline{v'}$ for $|\overline{v'}| = k$.*

$$\begin{array}{ll}
([\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; ((\varphi^{(k,i)})t_1\lambda)(\varphi^{(k+1,i)})t_2]\!]_e & \equiv^{j=1+nl(t_1), X=hd^j(\overline{v''})} \\
([\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})t_1]\!]_e\lambda_X)[\![\overline{v} +\!\!\!+ \overline{v'} +\!\!\!+ X; \overline{v''}_{\geq j+1}; (\varphi^{(k+1,i)})t_2]\!]_e & \equiv^{Lem\ 5.37} \\
([\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; t_1]\!]_e\lambda_X)[\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'} +\!\!\!+ x; \overline{v''}_{\geq j+1}; t_2]\!]_e & \equiv \\
[\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v''}; (t_1\lambda)t_2]\!]_e & \equiv^{Lem\ 5.37} \\
[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v''}; (\varphi^{(k,i)})(t_1\lambda)t_2]\!]_e & \square
\end{array}$$

Lem 6.7 *φ-destruction is sound:* $\forall \overline{v_1} \in \mathcal{L}_{sp}, \overline{v_2} \in \mathcal{L}_\infty(\updownarrow), \overline{v_1} \cap \overline{v_2} = \emptyset, n, i \in \mathbb{P}, k \in \mathbb{N}$:

1. *If $n > k$ then* $[\![\overline{v_1}; \overline{v_2}; (\varphi^{(k,i)})n]\!]_e \equiv [\![\overline{v_1}; \overline{v_2}; n+i]\!]_e$.
2. *If $n \leq k$ then* $[\![\overline{v_1}; \overline{v_2}; (\varphi^{(k,i)})n]\!]_e \equiv [\![\overline{v_1}; \overline{v_2}; n]\!]_e$.

Proof: *Assume $\overline{v_1}$ is ψ-free and $\overline{v_1} = \overline{v} + \!\!\!+ \overline{v'}$ such that $|\overline{v'}| = k$ and use Lem 5.37 and Cor 5.31:*
1.$[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v_2}; (\varphi^{(k,i)})n]\!]_e \equiv [\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v_2}; n]\!]_e \equiv [\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v_2}; n+i]\!]_e$
2.$[\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v_2}; (\varphi^{(k,i)})n]\!]_e \equiv [\![\overline{v} +\!\!\!+ \psi^i +\!\!\!+ \overline{v'}; \overline{v_2}; n]\!]_e \equiv comp_n(\overline{v'}) \equiv [\![\overline{v} +\!\!\!+ \overline{v'}; \overline{v_2}; n]\!]_e$ $\square$

Lem 6.8 *σ-destruction is sound:* $\forall t \in \mathcal{B}^{\lambda\delta\sigma\varphi}, \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, i, j \in \mathbb{P}$:

1. $[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})i]\!]_e \equiv [\![\overline{v}; \overline{v'}; t]\!]_e$.
2. $[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})j]\!]_e \equiv [\![\overline{v}; \overline{v'}; j]\!]_e$ *if* $j \neq i$.
3. $[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})\varepsilon]\!]_e \equiv \varepsilon$.

Proof: *Note that if $i \neq j$ then $[\![\overline{v}; \overline{v'}; j]\!]_e \not\equiv [\![\overline{v}; \overline{v'}; i]\!]_e$ by Lem 5.29:*
$[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})i]\!]_e \equiv [\![\overline{v}; \overline{v'}; i]\!]_e[[\![\overline{v}; \overline{v'}; i]\!]_e := [\![\overline{v}; \overline{v'}; t]\!]_e]' \equiv [\![\overline{v}; \overline{v'}; t]\!]_e$.
$[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})j]\!]_e \equiv [\![\overline{v}; \overline{v'}; j]\!]_e[[\![\overline{v}; \overline{v'}; i]\!]_e := [\![\overline{v}; \overline{v'}; t]\!]_e]' \equiv [\![\overline{v}; \overline{v'}; j]\!]_e$.
$[\![\overline{v}; \overline{v'}; (t\sigma^{(i)})\varepsilon]\!]_e \equiv [\![\overline{v}; \overline{v'}; \varepsilon]\!]_e[[\![\overline{v}; \overline{v'}; i]\!]_e := [\![\overline{v}; \overline{v'}; t]\!]_e]' \equiv \varepsilon$, *as $\varepsilon \notin \overline{v}$, for every $\overline{v}$.* $\square$

Lem 6.9 *σ-transition is α-sound:* $\forall \overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \overline{v} \cap \overline{v'} = \emptyset, i \in \mathbb{P}, t_1, t_2, t \in \mathcal{B}^{\lambda\delta\sigma\varphi}$:

1. $[\![\overline{v}; \overline{v'}; (t_1\sigma^{(i)})(t_2\lambda)t]\!]_e =_{\overline{\alpha}} [\![\overline{v}; \overline{v'}; ((t_1\sigma^{(i)})t_2\lambda)((\varphi)t_1\sigma^{(i+1)})t]\!]_e$
2. $[\![\overline{v}; \overline{v'}; (t_1\sigma^{(i)})(t_2\delta)t]\!]_e =_{\overline{\alpha}} [\![\overline{v}; \overline{v'}; ((t_1\sigma^{(i)})t_2\lambda)(t_1\sigma^{(i)})t]\!]_e$

The 6.10 *Let r be r'-transition or r'-destruction rule for $r' \in \{\sigma, \varphi\}$. $t \to_r t' \Rightarrow [\![t]\!] \equiv [\![t']\!]$.*

Proof: *Use lemmas 6.5, 6.6, 6.7, 6.8 and 6.9 above. (Note $t, t' \in \mathcal{B}^{\lambda\delta\sigma\varphi}$.)* $\square$

Transition and destruction rules of σ and φ work like substitution and variable updating and so return equivalent terms. σ-generation on the other hand, accommodates β-reduction.

Ex 6.11 $[\![\mathcal{F}; \updownarrow; (2\delta)(3\lambda)1]\!]_e \equiv ([\![\mathcal{F}; \updownarrow; 2]\!]_e\delta)([\![\mathcal{F}; \updownarrow; 3]\!]_e\lambda_{x'})[\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; 1]\!]_e \equiv (x_2\delta)(x_3\lambda_{x'})x'$. Also

$$\begin{array}{ll}
[\![\mathcal{F}; \updownarrow; (2\delta)(3\lambda)((\varphi)2\sigma^{(1)})1]\!]_e & \equiv \\
([\![\mathcal{F}; \updownarrow; 2]\!]_e\delta)([\![\mathcal{F}; \updownarrow; 3]\!]_e\lambda_{x'})[\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; ((\varphi)2\sigma^{(1)})1]\!]_e & \equiv \\
([\![\mathcal{F}; \updownarrow; 2]\!]_e\delta)([\![\mathcal{F}; \updownarrow; 3]\!]_e\lambda_{x'})([\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; 1]\!]_e[[\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; 1]\!]_e := [\![\mathcal{F} +\!\!\!+ x'; \updownarrow_{\geq 2}; (\varphi)2]\!]_e]' & \equiv \\
([\![\mathcal{F}; \updownarrow; 2]\!]_e\delta)([\![\mathcal{F}; \updownarrow; 3]\!]_e\lambda_{x'})(x'[x' := x_2]') & \equiv \\
([\![\mathcal{F}; \updownarrow; 2]\!]_e\delta)([\![\mathcal{F}; \updownarrow; 3]\!]_e\lambda_{x'})x_2 & \equiv \\
(x_2\delta)(x_3\lambda_{x'})x_2 &
\end{array}$$

Of course $(x_2\delta)(x_3\lambda_{x'})x'$ and $(x_2\delta)(x_3\lambda_{x'})x_2$ are not α-equivalent but are β-equivalent:
$(x_2\delta)(x_3\lambda_{x'})x' \to_{\overline{\beta}} x_2$ and $(x_2\delta)(x_3\lambda_{x'})x_2 \to_{\overline{\beta}} x_2$.

Lem 6.12 *σ-generation is $\alpha\beta$-sound. That is, for all $t, t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}$, for all $\overline{v} \in \mathcal{L}_{sp}, \overline{v'} \in \mathcal{L}_\infty(\updownarrow)$, such that $\overline{v} \cap \overline{v'} = \emptyset$, $[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)t]\!]_e = [\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})t]\!]_e$.*

Proof: *Let $i = 1 + nl(t_1), j = 1 + nl(t_2), X = hd^j(\overline{v}_{\geq i}), k = 1 + nl(t)$. Note that $[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)t]\!]_e \equiv ([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)[\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e =_{\overline{\beta}} [\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e[X := [\![\overline{v}; \overline{v'}; t_1]\!]_e]'$. Moreover,*

$$
\begin{array}{ll}
[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)((\varphi)t_1\sigma^{(1)})t]\!]_e & \equiv \\
([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)([\![\overline{v} + X; \overline{v'}_{\geq i+j}; ((\varphi)t_1\sigma^{(1)})t]\!]_e) & \equiv \\
([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)([\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e[X := [\![\overline{v} + x; \overline{v'}_{\geq i+j+k}; (\varphi)t_1]\!]_e]') & =_{\overline{\beta}}^{5.37,\ 5.38} \\
([\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e[X := [\![\overline{v}; \overline{v'}_{\geq i+j+k}; t_1]\!]_e]'[X := [\![\overline{v}; \overline{v'}; t_1]\!]_e]') & =_{\overline{\alpha}}^{Lem\ 6.3} \\
([\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e[X := [\![\overline{v}; \overline{v'}; t_1]\!]_e]'[X := [\![\overline{v}; \overline{v'}; t_1]\!]_e]') & \equiv^{Lem\ 5.42} \\
[\![\overline{v} + X; \overline{v'}_{\geq i+j}; t]\!]_e[X := [\![\overline{v}; \overline{v'}; t_1]\!]_e]' & \square
\end{array}
$$

7 The meaning and soundness of β-reduction

Recall from Def 4.11 that β-reduction was defined as a combination of σ-, φ- and μ-reduction. Hence, as σ- and φ-reduction are sound, all we have left to show here is that μ-reduction is sound. More precisely, we will show that μ-generation is $\alpha\beta$-sound and that μ-destruction and transition are sound. Let us first define the meaning of terms with μ-leading items.

Def 7.1 *(μ-semantics) If t is an $\Omega_{\lambda\delta}$-term, $\overline{v} \in \mathcal{L}^{-1}(\Theta), \overline{v'} \in \mathcal{L}(\Theta), \theta \in \Theta, \overline{v''} \in \mathcal{L}_\infty(\updownarrow)$, $\overline{v} \cap \overline{v''} = \emptyset, i \in I\!\!P$ and i does not refer to any free variable of t, we define:*

$$
\begin{array}{lll}
[\![\overline{v}; \overline{v''}; (\mu^{(i)})t]\!]_e & \equiv & [\![\overline{v}; \emptyset; \overline{v''}; (\mu^{(i)})t]\!] \\
[\![\overline{v}; \overline{v'}; \overline{v''}; (\mu^{(1)})t]\!] & \equiv & [\![\overline{v} + hd(\overline{v''}) + \overline{v'}; \overline{v''}_{\geq 2}; t]\!]_e \\
[\![\overline{v} + \theta; \overline{v'}; \overline{v''}; (\mu^{(i+1)})t]\!] & \equiv & [\![\overline{v}; \theta + \overline{v'}; \overline{v''}; (\mu^{(i)})t]\!]
\end{array}
$$

The provision "i does not refer to a free variable of t" can be assumed due to Lem 4.13; this is the only case we need to define the semantics for. Moreover, it suffice to take $\overline{v} \in \mathcal{L}^{-1}(\Theta)$, because t is an $\Omega_{\lambda\delta}$-term, so we never generate ψ's in the list $\overline{v}$.

Ex 7.2

1. $$
\begin{array}{ll}
[\![(\mu^{(1)})(2\lambda)1]\!] & \equiv \\
[\![\mathcal{F}; \updownarrow; (\mu^{(1)})(2\lambda)1]\!]_e & \equiv \\
[\![\mathcal{F}; \emptyset; \updownarrow; (\mu^{(1)})(2\lambda)1]\!] & \equiv \\
[\![\mathcal{F} + x'; \updownarrow_{\geq 2}; (2\lambda)1]\!]_e & \equiv \\
([\![\mathcal{F} + x'; \updownarrow_{\geq 2}; 2]\!]_e\lambda_{x''})[\![\mathcal{F} + x'; \updownarrow_{\geq 3}; 1]\!]_e & \equiv (x_1\lambda_{x''})x''
\end{array}
$$

2. $$
\begin{array}{ll}
[\![(\mu^{(2)})(1\lambda)1]\!] & \equiv \\
[\![\mathcal{F}; \updownarrow; (\mu^{(2)})(1\lambda)1]\!]_e & \equiv \\
[\![\mathcal{F}; \emptyset; \updownarrow; (\mu^{(2)})(1\lambda)1]\!] & \equiv \\
[\![\mathcal{F}_{\geq 2}; x_1; \updownarrow; (\mu^{(1)})(1\lambda)1]\!] & \equiv \\
[\![\mathcal{F}_{\geq 2} + x' + x_1; \updownarrow_{\geq 2}; (1\lambda)1]\!]_e & \equiv \\
([\![\mathcal{F}_{\geq 2} + x' + x_1; \updownarrow_{\geq 2}; 1]\!]_e)\lambda_{x''})[\![\mathcal{F}_{\geq 2} + x' + x_1 + x''; \updownarrow_{\geq 3}; 1]\!]_e & \equiv (x_1\lambda_{x''})x''
\end{array}
$$

Note that $[\![(\mu^{(1)})(1\lambda)1]\!]$ is not allowed, since 1 refers to the free variable 1 in $(1\lambda)1$.

Lem 7.3 *Let t be an $\Omega_{\lambda\delta}$-term. If λ° does not bind any variable in $(\lambda^\circ)(\lambda^1)(\lambda^2)\ldots(\lambda^k)t$, then $\forall \overline{v} \in \mathcal{L}^{-1}(\Theta), \overline{v''} \in \mathcal{L}(\Theta), \overline{v'} \in \mathcal{L}_\infty(\updownarrow), \theta, \theta' \in \Theta$, such that $(\overline{v'} + \overline{v''}) \cap \overline{v'} = \emptyset, \theta, \theta' \notin \overline{v} \cup \overline{v'} \cup \overline{v''}, |\overline{v''}| = k$, we have: $[\![\overline{v} + \theta + \overline{v''}; \overline{v'}; t]\!]_e \equiv [\![\overline{v} + \theta' + \overline{v''}; \overline{v'}; t]\!]_e$*

Proof: *By induction on t using lemmas 5.29 and 6.2.* $\square$

Lem 7.4 *Let* $(t_1\delta)(t_2\lambda)$ *be void in* $(t_1\delta)(t_2\lambda)t$, $i = 1 + nl(t_1)$ *and* $j = 1 + nl(t_2)$. $\forall \overline{v} \in \mathcal{L}^{-1}(\Theta)$, $\overline{v'} \in \mathcal{L}_\infty(\updownarrow)$, $\overline{v} \cap \overline{v'} = \emptyset \wedge X = hd^{i+j-1}(\overline{v'}) \Rightarrow ([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)$ *is void in* $[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)t]\!]_e$.

Proof: *By induction on* $\Omega_{\lambda\delta}$*-terms* t. □

Lem 7.5 μ*-generation is* $\alpha\beta$*-sound. I.e.,* $\forall t_1, t_2, t$ $\Omega_{\lambda\delta}$*-terms,* $\forall \overline{v} \in \mathcal{L}^{-1}(\Theta), \overline{v'} \in \mathcal{L}_\infty(\updownarrow)$ *such that* $\overline{v} \cap \overline{v'} = \emptyset$, *if* $(t_1\delta)(t_2\lambda)$ *is void in* t *then:* $[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)t]\!]_e = [\![\overline{v}; \overline{v'}; (\mu^{(1)})t]\!]_e$

Proof: *By induction on* t. *Let* $i = 1 + nl(t_1), j = 1 + nl(t_2), X = hd^i(\overline{v'}_{\geq j}) = hd^{i+j-1}(\overline{v'})$.

- *If* $t \equiv \varepsilon$ *then obvious.*
- *If* $t \equiv m$ *then* $m > 1$. *Moreover,* $([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)[\![\overline{v} + x; \overline{v'}_{\geq i+j}; m]\!]_e \equiv$
  $([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)[\![\overline{v}; \overline{v'}_{\geq i+j}; m-1]\!]_e =_\beta^{Lem\ 7.4}$
  $[\![\overline{v}; \overline{v'}_{\geq i+j}; m-1]\!]_e \equiv^{lemmas\ 5.29\ and\ 6.2} [\![\overline{v} + hd(\overline{v'}); \overline{v'}_{\geq 2}; m]\!]_e \equiv [\![\overline{v}; \overline{v'}; (\mu^{(1)})m]\!]_e$.
- *If* $t \equiv (t'_1\lambda)t'_2$ *then:* $[\![\overline{v}; \overline{v'}; (t_1\delta)(t_2\lambda)(t'_1\lambda)t'_2]\!]_e \equiv^{k=1+nl(t'_1), X'=hd^k(\overline{v'}_{\geq i+j})}$
  $([\![\overline{v}; \overline{v'}; t_1]\!]_e\delta)([\![\overline{v}; \overline{v'}_{\geq i}; t_2]\!]_e\lambda_X)([\![\overline{v} + x; \overline{v'}_{\geq i+j}; t'_1]\!]_e\lambda_{X'})[\![\overline{v} + x + x'; (\overline{v'}_{\geq i+j})_{\geq k+1}; t'_2]\!]_e =_\beta^{Lem\ 7.4}$
  $[\![\overline{v} + X; \overline{v'}_{\geq i+j}; (t'_1\lambda)t'_2]\!]_e =_\alpha^{Lem\ 6.3} [\![\overline{v} + X; \overline{v'}_{\geq 2}; (t'_1\lambda)t'_2]\!]_e \equiv^{Lem\ 7.3}$
  $[\![\overline{v} + hd(\overline{v'}); \overline{v'}_{\geq 2}; (t'_1\lambda)t'_2]\!]_e \equiv [\![\overline{v}; \overline{v'}; (\mu^{(1)})(t'_1\lambda)t'_2]\!]_e$
- *If* $t \equiv (t'_1\delta)t'_2$ *then similar.* □

Rem 7.6 Note that μ-generation is not sound. In particular, $[\![\mathcal{F}; \updownarrow; (4\delta)(\lambda)2]\!]_e \equiv (x_4\delta)(\lambda_{x'})x_1$ and $[\![\mathcal{F}; \updownarrow; (\mu^{(1)})2]\!]_e \equiv [\![\mathcal{F} + x'; \updownarrow_{\geq 2}; 2]\!]_e \equiv x_1$. Now $(x_4\delta)(\lambda_{x'})x_1 =_\beta x_1$ and $(x_4\delta)(\lambda_{x'})x_1 \not\equiv x_1$.

Lem 7.7 μ*-transition is sound:* $\forall \Omega_{\lambda\delta}$*-terms* t_1, t_2, $\overline{v} \in \mathcal{L}^{-1}(\Theta)$, $\overline{v'''} \in \mathcal{L}_\infty(\updownarrow)$ *such that* $\overline{v} \cap \overline{v'''} = \emptyset$, $\forall i \in I\!\!P$, *if* $i \notin FV((t_1\lambda)t_2)$, $k = 1 + nl(t_1)$, $X = hd^k(\overline{v'''})$ *then:*

$$
\begin{array}{lrcl}
1. & [\![\overline{v}; \overline{v'''}; (\mu^{(i)})(t_1\lambda)t_2]\!]_e & \equiv & ([\![\overline{v}; \overline{v'''}; (\mu^{(i)})t_1]\!]_e\lambda_X)[\![\overline{v} + x; \overline{v'''}_{\geq k+1}(\mu^{(i+1)})t_2]\!]_e \\
2. & [\![\overline{v}; \overline{v'''}; (\mu^{(i)})(t_1\delta)t_2]\!]_e & \equiv & ([\![\overline{v}; \overline{v'''}; (\mu^{(i)})t_1]\!]_e\delta)[\![\overline{v}; \overline{v'''}_{\geq k}; (\mu^{(i+1)})t_2]\!]_e
\end{array}
$$

Proof: *We show 1 only as 2 is similar. Let* $\overline{v} = \overline{v'} + \overline{v''}$ *such that* $|\overline{v''}| = i - 1$:

$$
\begin{array}{ll}
([\![\overline{v}; \overline{v'''}; (\mu^{(i)})t_1]\!]_e\lambda_X)[\![\overline{v} + x; \overline{v'''}_{\geq k+1}; (\mu^{(i+1)})t_2]\!]_e & \equiv \\
([\![\overline{v'} + hd(\overline{v'''}) + \overline{v''}; \overline{v'''}_{\geq 2}; t_1]\!]_e\lambda_X)[\![\overline{v'} + hd(\overline{v'''}_{\geq k+1}) + \overline{v''} + x; \overline{v'''}_{\geq k+2}; t_2]\!]_e & \equiv^{Lem\ 7.3} \\
[\![\overline{v'} + hd(\overline{v'''}) + \overline{v''}; \overline{v'''}_{\geq 2}; (t_1\lambda)t_2]\!]_e & \equiv \\
[\![\overline{v}; \overline{v'''}; (\mu^{(i)})(t_1\lambda)t_2]\!]_e & \square
\end{array}
$$

Lem 7.8 μ*-destruction is sound:* $\forall \overline{v} \in \mathcal{L}^{-1}(\Theta)$, $\overline{v'''} \in \mathcal{L}_\infty(\updownarrow)$ *such that* $\overline{v} \cap \overline{v'''} = \emptyset$, $\forall i, m \in I\!\!P$:

- $[\![\overline{v}; \overline{v'''}; (\mu^{(i)})\varepsilon]\!]_e \equiv \varepsilon$.
- $[\![\overline{v}; \overline{v'''}; (\mu^{(i)})m]\!]_e \equiv [\![\overline{v'} + \overline{v''}; \overline{v'''}; m]\!]_e$ *if* $m < i$.
- $[\![\overline{v}; \overline{v'''}; (\mu^{(i)})m]\!]_e \equiv [\![\overline{v'} + \overline{v''}; \overline{v'''}; m-1]\!]_e$ *if* $m > i$.

Proof: $[\![\overline{v}; \overline{v'''}; (\mu^{(i)})\varepsilon]\!]_e \equiv \varepsilon$, *easy.* $[\![\overline{v}; \overline{v'''}; (\mu^{(i)})m]\!]_e \equiv [\![\overline{v'} + hd(\overline{v'''}) + \overline{v''}; \overline{v'''}_{\geq 2}; m]\!]_e \equiv t$ *where* $\overline{v} = \overline{v'} + \overline{v''}$ *and* $|\overline{v''}| = i - 1$. *If* $m < i$ *then* $m \leq i - 1$ *and* $t \equiv [\![\overline{v'} + \overline{v''}; \overline{v'''}; m]\!]_e$. *If* $m > i$ *then* $m \geq i + 1$ *and* $t \equiv [\![\overline{v'} + \overline{v''}; \overline{v'''}; m-1]\!]_e$. □

8 Conclusions and comparison

In order to show the soundness of our calculus we provided a translation from $\mathcal{B}$ into $\overline{\Lambda}$, a variant of Λ where bound variables are taken from a particular ordered list. Our translation functions are important on their own. First, it is nice to have a mechanical procedure which takes terms written with variable names and returns terms with de Bruijn's indices. Second, it is equally important and interesting to go the other way. For instance, when translating a term (with de Bruijn indices) that represents some mathematical theory/proof to a term with named variables, we want particular names to be used. In fact, one of the advantages of de Bruijn's indices is that α-conversion is no longer needed. Now, terms written with de Bruijn's indices are difficult to understand even for those who are familiar with them. Variable names on the other hand, clarify the term in hand but cause a lot of complications when applying reduction and substitution. If however, we order our lists of free and bound variables, then we can avoid the difficulty caused by variable names. In fact, this is what we do in this paper. We take our lists of variables to be ordered and we translate $\mathcal{B}$ into $\overline{\Lambda}$ (i.e. using variable names) in a unique way via $[\![\cdot]\!]$. When in $\overline{\Lambda}$, it is up to us to equate terms modulo α-conversion rather than being forced to do it in the translation (see Appendix B).

In order to make substitution explicit and to discuss β-reduction, we had to add three kinds of reduction rules: the φ-, σ- and μ-reductions. φ updates variables, σ substitutes terms for variables and μ decreases the indices as a result of a β-conversion which removes a λ from a term. Each kind of reduction has three rules: generation, transition and destruction. Now, substitution and reduction in $\overline{\Lambda}$ are given similarly to that of the classical calculus; i.e. implicit and global. Therefore, we show that our reduction rules actually do represent reduction and substitution in $\overline{\Lambda}$ and are hence sound. In particular, we show that σ-, μ- φ-destruction and φ-, μ-transition are sound in that if $t \to_r t'$ where r is one of these rules, then $[\![t]\!] \equiv [\![t']\!]$. This is very nice because the corresponding reductions in $\overline{\Lambda}$ also return equivalent rather than α-equivalent terms. Furthermore, we show that σ-transition is α-sound in that if $t \to_{\sigma-transition} t'$ then $[\![t]\!] =_{\overline{\alpha}} [\![t']\!]$. We also show that σ- and μ-generation are $\alpha\beta$-sound in that if $t \to_r t'$ where r is one of these two rules, then $[\![t]\!] =_{\overline{\alpha\beta}} [\![t']\!]$. Now, we are satisfied with the result concerning β-conversion. In fact, σ- and μ-generation do actually represent β-conversion in $\mathcal{B}$. Note moreover that in the soundness proof of σ-transition and σ- and μ-generation, α-conversion appears despite the fact that we avoided it in our translation function. Look for example at the proof of Lem 7.5. When $t \equiv (t'_1\lambda)t'_2$, we had to apply Lem 6.3 to obtain an α-equivalent term. We have hence singled out the steps in which α must be used: σ- and μ-generation and in σ-transition. Finally, note that we did not discuss completeness because this becomes here a trivial matter. In fact, everything that can be shown in the classical λ-calculus can be shown in our own. Even better, our calculus is more expressive in that it accommodates explicit substitution whereas the classical one does not.

Work on explicit substitution with de Bruijn indices has been first done in depth by Curien (in his PhD thesis, 1983) and was based on categorical combinators. Curien's original work was pursued by applications such as the categorical abstract machine of [CCM 87]. [ACCL 91] provides an algebraic syntax and semantics for explicit substitution where de Bruijn's indices are used. The connection with the classical λ-calculus is not investigated. [HL 89] proposes confluent systems of substitution based on the study of categorical combinators and [Field 90] provides an account of explicit substitution similar to that of [ACCL 91]. Our approach in this paper follows de Bruijn rather than Curien in using concepts which belong to the λ-

calculus rather than to Category Theory. In fact, we believe that as λ and δ are operators of the λ-calculus whose behaviour is well-understood, σ, φ and μ should also be treated similarly. This approach of treating the λ-calculus via items has proven advantageous in our various extensions as in [BKN 95], [KN 95] and [KN 96b]. [KN 93] provides an account of explicit substitution which is used to discuss local and global substitution and reduction. No semantics is provided for that account and the precision of this paper is not assumed there. The reduction rules however of the present paper are based on [KN 93] even though there, there was no μ-reduction and α-reduction was assumed. We believe that we have in this paper presented the most extensive approach of variable manipulation, substitution and reduction. Our approach can be easily and in a straightforward fashion implemented because we have carried out all the difficult work related to variables. Furthermore, as [KN 93] has shown that [ACCL 91] can be interpreted in [KN 93] and as $\mathcal{B}$ is an extension of [KN 93], our work here also applies to [ACCL 91]. [Kra 93] provides a semantics of the explicit substitution of an extension of [KN 93]. The work of [Kra 93] originated from our function e of this paper but ignores to order the list of bound variables which we call $\updownarrow$ imposing α-conversion. In Appendix B, we provide a semantics where all α-equivalent terms are identifiable.

In [KR 95], λs, the subsystem of $\mathcal{B}$ where σ-generation does not preserve the $\delta\lambda$-couple, has been studied. λs along with the system of [BBLR 95] are the first calculi of explicit substitution which enjoy confluence on closed terms and preserve strong normalisation. In [KR 9x], it was shown that in the simply typed version of λs, well-typed terms are strongly normalising. In [KR 9y], it was shown that λs extended with open terms is confluent. At the moment, we are extending the work of [KR 95], [KR 9x] and [KR 9y] to study the properties of λs where σ-generation preserves the $\delta\lambda$-couple, hence resulting in the system $\mathcal{B}$ of this paper. Finally, Daniel Briaud noted our attention that adding intersection types to [BBLR 95] is problematic as there will be terms that are strongly normalising but not typable. This is not the case when intersection types are added to λs. This could be seen as an advantage to our framework of remaining close to the λ-calculus rather than using combinators as in [ACCL 91] and [BBLR 95].

A Making i negative in $(\varphi^{(k,i)})$

Up to now, the i-superscript in $(\varphi^{(k,i)})$ has been considered an element of $\mathbb{P}$. If however, we allow in $(\varphi^{(k,i)})$, i to be negative, we could include the following rule:

Def A.1 *($\delta\lambda$-destruction rule) For all t_1, t_2 $\Omega_{\lambda\delta}$-terms, we have: $(t_1\delta)(t_2\lambda) \to_\emptyset (\varphi^{(0,-1)})$ provided that the λ in $(t_2\lambda)$ does not bind any variable in the term following $(t_1\delta)(t_2\lambda)$, i.e. provided that $(t_1\delta)(t_2\lambda)$ is void. Sometimes we denote $\to_\emptyset$ by* **void β-reduction**.

Unfortunately, negative superscripts identify *different* variables as in: $(\varphi^{(1,-1)})(2\delta)1 \twoheadrightarrow_\varphi (1\delta)1$. Hence, updating is no longer an injection, which can be highly undesirable. This unpleasant effect however, does not occur in the setting presented above: a φ-item with a negative exponent only occurs after the clean-up of a void $\delta\lambda$-segment, hence with a λ that does not bind any variable. Therefore, the injective property of updating is not threatened. Now the σ-rules together with the $\delta\lambda$-destruction rule, enable us to accomplish β-reduction:

Def A.2 *(one-step β-reduction $\to_{\beta''}$) One-step β-reduction of an $\Omega_{\lambda\delta}$-term is the combination of one σ-generation from a $\delta\lambda$-segment $\overline{s}$, the transition of the generated σ-item through the appropriate subterm in a global manner, followed by a number of σ-destructions, and updated* *by φ-items until again an $\Omega_{\lambda\delta}$-term is obtained. Finally, there follows one void β-reduction for the disposal of $\overline{s}$, and we use the φ-rules to dispose completely of the φ-items.*

Ex A.3 $(1\delta)(2\lambda)(4\delta)1 \to_{\beta''} (3\delta)1$ as follows:

$$\begin{array}{lll}
(1\delta)(2\lambda)(4\delta)1 & \to_{\sigma} & (1\delta)(2\lambda)((\varphi)1\sigma^{(1)})(4\delta)1 \\
& \to_{\sigma\varphi} & (1\delta)(2\lambda)((2\sigma^{(1)})4\delta)(2\sigma^{(1)})1 \\
& \twoheadrightarrow_{\sigma} & (1\delta)(2\lambda)(4\delta)2 \\
& \to_{\emptyset} & (\varphi^{(0,-1)})(4\delta)2 \\
& \to_{\varphi} & ((\varphi^{(0,-1)})4\delta)(\varphi^{(0,-1)})2 \\
& \twoheadrightarrow_{\varphi} & (3\delta)1.
\end{array}$$

We used in this paper μ instead of negative superscripts for φ in order to make a clear distinction between the harmless *positive* updating and the potentially dangerous *negative* updating (see our remark after Def A.1). To be precise: $(\mu^{(i)})$ is equivalent to $(\varphi^{(i-1,-1)})$; but in the case of void reductions, $(\varphi^{(i-1,-1)})$ has the same effect as $(\varphi^{(i,-1)})$.

B An alternative semantics

In the definition of the semantic function from $\mathcal{B}$ to $\overline{\Lambda}$, we took $\mathcal{F}$ and $\updownarrow$ which were both ordered (see Def 6.1). This enabled us to translate every term t of $\mathcal{B}$ to a unique term t' of $\overline{\Lambda}$ rather than to t'' where $t' =_\alpha t''$. In this appendix, we define the semantic function which returns any element of the α-equivalence class. This is not the approach we use in the paper because implementation cannot rely on α-conversion. Of course we pay a price (which is not high compared with the advantages) in that we had to manipulate not only the list of free variables but also the list of bound ones.

Def B.1 *(λ- and δ-semantics) For all $t_1, t_2 \in \mathcal{B}^{\lambda\delta}, \overline{v} \in \mathcal{L}(\updownarrow), n \in I\!P \cup \{\varepsilon\}$,*

$$\begin{array}{lll}
[\![\overline{v}; (t_1\lambda)t_2]\!] & =_{df} & ([\![\overline{v}; t_1]\!]\lambda_v)[\![\overline{v} + v; t_2]\!] \textit{ where } v \in \updownarrow \setminus \overline{v} \\
[\![\overline{v}; (t_1\delta)t_2]\!] & =_{df} & ([\![\overline{v}; t_1]\!]\delta)[\![\overline{v}; t_2]\!] \\
[\![\overline{v}; n]\!] & =_{df} & \begin{cases} comp_n(\overline{v}) & \textit{if } n \leq |\overline{v}| \\ x_{n-|\overline{v}|} & n > |\overline{v}| \\ \varepsilon & \textit{if } n = \varepsilon \end{cases}
\end{array}$$

Ex B.2

$$\begin{array}{ll}
[\![\emptyset; (\lambda)(1\lambda)(1\delta)3]\!] & \equiv^{X_1 \in \updownarrow, X_1 \textit{ is arbitrary}} \\
([\![\emptyset; \varepsilon]\!]\lambda_{X_1})[\![X_1; (1\lambda)(1\delta)3]\!] & \equiv \\
(\varepsilon\lambda_{X_1})([\![X_1; 1]\!]\lambda_{X_2})[\![X_1X_2; (1\delta)3]\!] & \equiv^{X_2 \in \updownarrow, X_2 \textit{ is arbitrary}, X_2 \not\equiv X_1} \\
(\varepsilon\lambda_{X_1})(comp_1(X_1)\lambda_{X_2})([\![X_1X_2; 1]\!]\delta)[\![X_1X_2; 3]\!] & \equiv \\
(\varepsilon\lambda_{X_1})(X_1\lambda_{X_2})(comp_1(X_1X_2)\delta)x_{3-|X_1X_2|} & \equiv \\
(\varepsilon\lambda_{X_1})(X_1\lambda_{X_2})(X_2\delta)x_1 &
\end{array}$$

We need the following which defines variable substitution of lists of variables.

Def B.3 *(Substitution in lists) If $\overline{v}$ is a list of variables of $\overline{\Lambda}$, then we define $\overline{v}[v := v']'$ to be the list $\overline{v}$ but where all occurrences of v have been replaced by v'.*

Now the following lemmas are needed to show that $[\![\cdot;\cdot]\!]$ is well defined.

Lem B.4 *For any* $\overline{v}, t$, $FV([\![\overline{v}; t]\!]) \subseteq \overline{v} \cup \mathcal{F}$.

Proof: *By induction on* t*, recalling that* ε *is neither free nor bound.* $\square$

Lem B.5 *For* $X' \in \updownarrow \setminus \overline{v}$, $X \in \overline{v}$, $\overline{v} \in \mathcal{L}(\updownarrow)$ *and* $t \in \mathcal{B}^{\lambda\delta}$: $[\![\overline{v}; t]\!][X := X']' =_{\overline{\alpha}} [\![\overline{v}[X := X']'; t]\!]$.

Proof: *By induction on* $t \in \mathcal{B}^{\lambda\delta}$.

1. $[\![\overline{v}; n]\!][X := X']' \equiv [\![\overline{v}[X := X']'; n]\!]$ *for* $n \in I\!\!P \cup \{\varepsilon\}$.
2. $[\![\overline{v}; (t_1\delta)t_2]\!][X := X']' \equiv (([\![\overline{v}; t_1]\!]\delta)[\![\overline{v}; t_2]\!])[X := X']' \equiv$
   $([\![\overline{v}; t_1]\!][X := X']'\delta)[\![\overline{v}; t_2]\!][X := X']' =^{IH}_{\overline{\alpha}}$
   $([\![\overline{v}[X := X']'; t_1]\!]\delta)[\![\overline{v}[X := X']'; t_2]\!] \equiv [\![\overline{v}[X := X']'; (t_1\delta)t_2]\!]$.
3. $[\![\overline{v}; (t_1\lambda)t_2]\!][X := X']' \equiv^{X_1 \in \updownarrow\setminus\overline{v}, X_1 \not\equiv X'} (([\![\overline{v}; t_1]\!]\lambda_{X_1})[\![\overline{v} +\!\!+ x_1; t_2]\!])[X := X']' \equiv$
   $([\![\overline{v}; t_1]\!][X := X']'\lambda_{X_1})[\![\overline{v} +\!\!+ x_1; t_2]\!][X := X']' \equiv^{IH}$
   $([\![\overline{v}[X := X']'; t_1]\!]\lambda_{X_1})[\![(\overline{v} +\!\!+ x_1)[X := X']'; t_2]\!] \equiv$
   $([\![\overline{v}[X := X']'; t_1]\!]\lambda_{X_1})[\![\overline{v}[X := X']' +\!\!+ x_1; t_2]\!] \equiv [\![\overline{v}[X := X']'; (t_1\lambda)t_2]\!]$.
4. $[\![\overline{v}; (t_1\lambda)t_2]\!][X := X']' \equiv^{X' \in \updownarrow\setminus\overline{v}} (([\![\overline{v}; t_1]\!]\lambda_{X'})[\![\overline{v} +\!\!+ x'; t_2]\!])[X := X']' \equiv^{X'' \notin FV([\![\overline{v} +\!\!+ x'; t_2]\!])}$
   $(([\![\overline{v}; t_1]\!]\lambda_{X''})[\![\overline{v} +\!\!+ x'; t_2]\!][X' := X'']')[X := X']' =^{Lem\ B.4, IH}_{\overline{\alpha}}$
   $(([\![\overline{v}; t_1]\!]\lambda_{X''})[\![\overline{v} +\!\!+ x'[X' := X'']'; t_2]\!])[X := X']' \equiv (([\![\overline{v}; t_1]\!]\lambda_{X''})[\![\overline{v} +\!\!+ x''; t_2]\!])[X := X']'$
   Now, refer to case 3 above. $\square$

Lem B.6 $([\![\overline{v}; t_1]\!]\lambda_{X_1})[\![\overline{v} +\!\!+ X_1; t_2]\!] =_{\overline{\alpha}} ([\![\overline{v}; t_1]\!]\lambda_{X_2})[\![\overline{v} +\!\!+ X_2; t_2]\!]$ *for* $X_1, X_2 \in \updownarrow \setminus \overline{v}$.

Proof: *If* $X_1 = X_2$*, then nothing to prove. If* $X_1 \neq X_2$*, then:*

$$
\begin{array}{ll}
([\![\overline{v}; t_1]\!]\lambda_{X_1})[\![\overline{v} +\!\!+ X_1; t_2]\!] & \equiv^{X_2 \notin FV([\![\overline{v} +\!\!+ X_1; t_2]\!]), Lem\ B.4} \\
([\![\overline{v}; t_1]\!]\lambda_{X_2})[\![\overline{v} +\!\!+ X_1; t_2]\!][X_1 := X_2]' & =^{Lem\ B.5}_{\overline{\alpha}} \\
([\![\overline{v}; t_1]\!]\lambda_{X_2})[\![(\overline{v} +\!\!+ X_1)[X_1 := X_2]'; t_2]\!] & \equiv^{X_1, X_2 \notin \overline{v}} \\
([\![\overline{v}; t_1]\!]\lambda_{X_2})[\![\overline{v} +\!\!+ X_2; t_2]\!] & \equiv [\![\overline{v}; (t_1\lambda)t_2]\!]
\end{array}
$$
$\square$

Lem B.7 $[\![\cdot; \cdot]\!]$ *as defined in Def B.1 is well defined:* $\forall \overline{v}, t$, $[\![\overline{v}; t]\!]$ *is unique up to* α*-conversion, (I.e. does not depend on the choice of* v *in clause 1 of Def B.1).*

Proof: *By induction on* $t \in \mathcal{B}^{\lambda\delta}$ *using Lem B.6 for the interesting case* $t \equiv (t_1\lambda)t_2$. $\square$

Lem B.8 $\forall t \in \mathcal{B}^{\lambda\delta}$, $c(t, \overline{s}, \updownarrow \setminus sl(\overline{s})) =_{\overline{\alpha}} [\![sl(\overline{s}); t]\!]$. *(Hence* $e(t) =_{\overline{\alpha}} [\![\emptyset; t]\!]$*.)*

Proof: *By induction on* t. $\square$

Now the definition which replaces Def 6.1 is the following:

Def B.9 *(Semantics of* $\mathcal{B}^{\lambda\delta\sigma\varphi}$*)* $\forall t, t_1, t_2 \in \mathcal{B}^{\lambda\delta\sigma\varphi}$, $\overline{v} \in \mathcal{L}_{sp}$, $\overline{v'} \in \mathcal{L}(\Theta)$, $\theta \in \Theta$, $i, n \in I\!\!P$, $k \in I\!\!N$:

$$
\begin{array}{lll}
M1. & [\![t]\!] & =_{df}\ [\![\mathcal{F}; t]\!]
\end{array}
$$

$$
\begin{array}{lll}
M2. & [\![\overline{v}; \varepsilon]\!] & =_{df}\ \varepsilon \\
M3. & [\![\overline{v}; n]\!] & =_{df}\ comp_n(\overline{v}) \\
M4. & [\![\overline{v}; (t_1\lambda)t_2]\!] & =_{df}\ ([\![\overline{v}; t_1]\!]\lambda_X)[\![\overline{v} +\!\!+ X; t_2]\!] \text{ where } X \in \updownarrow \setminus \overline{v} \\
M5. & [\![\overline{v}; (t_1\delta)t_2]\!] & =_{df}\ ([\![\overline{v}; t_1]\!]\delta)[\![\overline{v}; t_2]\!] \\
M6. & [\![\overline{v}; (t_1\sigma^{(i)})t_2]\!] & =_{df}\ [\![\overline{v}; t_2]\!][[\![\overline{v}; i]\!] := [\![\overline{v}; t_1]\!]]' \\
M7. & [\![\overline{v}; (\varphi^{(k,i)})t]\!] & =_{df}\ [\![\overline{v}; \emptyset; (\varphi^{(k,i)})t]\!]
\end{array}
$$

$$
\begin{array}{lll}
M8. & [\![\overline{v}; \overline{v'}; (\varphi^{(0,i)})t_1]\!] & =_{df}\ [\![\overline{v} +\!\!+ \psi^i +\!\!+ \overline{v'}; t]\!] \\
M9. & [\![\overline{v} +\!\!+ \theta; \overline{v'}; (\varphi^{(k+1,i)})t_1]\!] & =_{df}\ [\![\overline{v}; \theta +\!\!+ \overline{v'}; (\varphi^{(k,i)})t]\!] \\
M10. & [\![\overline{v} +\!\!+ \theta +\!\!+ \psi^{k+1}; \overline{v'}; t]\!] & =_{df}\ [\![\overline{v} +\!\!+ \psi^k; \overline{v'}; t]\!]
\end{array}
$$

Soundness of the reduction rules with respect to this definition is left to the reader.

References

[ACCL 91] Abadi, M., Cardelli, L., Curien, P.-L. and Lévy, J.-J., (1991) Explicit substitutions, *Functional Programming 1 (4)*, 375-416.

[Bar 84] Barendregt, H., (1984) *Lambda Calculus: its Syntax and Semantics,* North-Holland.

[Bar 92] Barendregt, H., (1992) Lambda calculi with types, *Handbook of Logic in Computer Science*, volume II, ed. Abramsky S., Gabbay D.M., Maibaum T.S.E., Oxford University Press.

[BBLR 95] Benaissa Z., Briaud D., Lescanne P. and Rouyer-Degli J., (1995), $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation, Personal communication.

[BKN 95] Bloo, R., Kamareddine, F., Nederpelt, R., The Barendregt Cube with Definitions and Generalised Reduction, to appear in *Information and Computation.*

[Con 86] Constable, R.L. et al., (1986) *Implementing Mathematics with the Nuprl proof development system*, Prentice Hall.

[CCM 87] Cousineau G., Curien P.L. and Mauny M., (1987) The Categorical Abstract Machine, *Science of Computer Programming 8*, 173-202.

[Field 90] Field, J., (1990) On laziness and optimality in λ-interpreters: tools for specification and analysis, 17^{th} *Annual Symposium on Principles of Programming Languages*, San Fransisco, 1-15.

[GM 93] Gordon M.J.C. and Melham, T.F. (eds), (1993) *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*, Cambridge University Press.

[HL 89] Hardin, Th. and Lévy, J.-J., (1989) A confluent calculus of substitutions, Lecture notes of the INRIA-ICOT symposium, Izu, Japan, November.

[KN 93] Kamareddine, F., and Nederpelt, R.P., (1993) On stepwise explicit substitution, *International Journal of Foundations of Computer Science 4 (3)*, 197-240, 1993.

[KN 94] Kamareddine, F., and Nederpelt, R.P., (1994) A unified approach to type theory through a refined λ-calculus, *Theoretical Computer Science* 136, 183-216.

[KN 95] Kamareddine, F., and Nederpelt, R.P., (1995) Refining reduction in the λ-calculus, *Journal of Functional Programming 5 (4).*

[KN 96a] Kamareddine, F., and Nederpelt, R.P., (1996) *A useful λ-notation*, *Theoretical Computer Science* 155.

[KN 96b] Kamareddine, F., and Nederpelt, R.P., (1996) Canonical Typing and Π–conversion in the Barendregt Cube, *Journal of Functional Programming 6 (2).*

[KR 95] Kamareddine, F., and Rios A., (1995) λ-calculus à la de Bruijn & explicit substitution, *Proceedings of PLILP '95*, LNCS, Springer-Verlag.

[KR 9x] Kamareddine, F., and Rios A., The λs-calculus: its typed and its extended versions, submitted.

[KR 9y] Kamareddine, F., and Rios A., The confluence of the λs_e-calculus via a generalised interpretation method, submitted.

[Kra 93] Krabbendam, J., (1993) On the soundness of explicit substitution, Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology.

[Mel 95] Melliès, P.-A., (1995) Typed λ-calculi with explicit substitutions may not terminate, *Proceedings of TLCA'95, Lecture Notes in Computer Science 902*, Springer-Verlag.

[NGdV 94] Nederpelt, R.P., Geuvers, J.H., and de Vrijer, R.C., eds, (1994) *Selected papers on Automath*, Studies in Logic and The foundations of Mathematics, 133, North Holland.