Efficiency of λ-calculi with explicit substitutions*

Fairouz Kamareddine and Alejandro Ríos †

August 30, 1996

Abstract

We introduce a criterion of efficiency to simulate β-reduction in calculi of explicit substitutions and we apply it to several calculi: $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\upsilon$, λs, λt and λu. The latter is presented here for the first time and may be considered as an efficient variant of λs. The results of this paper imply that calculi à la λs are usually more efficient at simulating β-reduction than calculi in the $\lambda\sigma$-style. In fact, we prove that λt is more efficient than $\lambda\upsilon$ and that λu is more efficient than $\lambda\upsilon$, $\lambda\sigma_{\Uparrow}$ and λs. We also give counterexamples to show that all other comparisons are impossible.

1 Introduction

The classical λ-calculus (cf. [2]) deals with substitution in an implicit way. This means that the computations to perform substitution are usually described with operators which do not belong to the language of the λ-calculus. There has however been an interest in formalising substitution explicitly in order to provide a theoretical framework for the implementation of functional programming languages. Several calculi including new operators to denote substitution and new rules to handle these operators have been proposed. Amongst these calculi we mention $C\lambda\xi\phi$ (cf. [6]); the calculi of categorical combinators (cf. [4]); $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\sigma_{SP}$ (cf. [1, 5, 14]) referred to as the $\lambda\sigma$-family; $\varphi\sigma BLT$ (cf. [7]); $\lambda\upsilon$ (cf. [3]) and $\lambda\zeta$ (cf. [13]) which are descendants of the $\lambda\sigma$-family; λs (cf. [8]), λs_e (cf. [11]) and λt (cf. [10]).

This article will focus on $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\upsilon$, λs, λt and λu which is an efficient version of λs presented here for the first time. All these calculi are rewriting systems on a set of terms that contain the classical terms of the λ-calculus (*pure terms*). All of them possess a rule to start β-reduction (the only rule of the λ-calculus) and a set of rules to compute the substitution generated by this starting rule.

Since calculi with explicit substitutions are intended to extend the classical λ-calculus, it is expected that β-reduction could be recovered in some way within these calculi, for instance, if $\lambda\xi$ is an explicit substitution calculus, we may have for pure terms a, b:

1. *one step simulation*: if $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda\xi} b$.
2. *big step simulation*: if $a \twoheadrightarrow_\beta b$ and b is in β-normal form then $a \twoheadrightarrow_{\lambda\xi} b$.

The calculi $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\upsilon$, λs, λt, λu have the property of *one step simulation* and we concentrate in this paper on the efficiency of this simulation which implies the big step one,

*This work was carried out under EPSRC grant GR/K25014.

†Department of Computing Science, 17 Lilybank Gardens, University of Glasgow, Glasgow G12 8QQ, Scotland, fax: +44 41 330 4913, *email*: fairouz@dcs.gla.ac.uk and rios@dcs.gla.ac.uk

leaving the study of the efficiency of the latter for future work. Our criterion of efficiency is essentially the following: we say that the calculus $\lambda\xi_1$ is more efficient than the calculus $\lambda\xi_2$ if for every simulation of a classical β-step in $\lambda\xi_2$ there is a shorter simulation in $\lambda\xi_1$.

There are reasons why we do not consider the other calculi. For example, $\lambda\zeta$ (the only calculus that) simulates just a big step β-reduction (and hence it does not make sense to study its efficiency in our sense), whereas λs_e, $\varphi\sigma BLT$ and $\lambda\sigma_{SP}$ are less interesting because they are less behaved calculi of explicit substitutions.

In section 2 we introduce the notation, recall the calculi, present the λu-calculus and give the formal statement of the criterion of efficiency to simulate β-reduction.

In section 3 we use our criterion to compare several of the above mentioned calculi. We conclude that λt is more efficient than λv, and that λu is more efficient than λs, λv and $\lambda\sigma_{\Uparrow}$.

In section 4 we give counterexamples to show the calculi that are incomparable according to our criterion, namely: λt cannot be compared with λu, λs, $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$; λu cannot be compared with $\lambda\sigma$ and λt; λs cannot be compared with λt, λv, $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$. We show also that, surprisingly, no comparison is possible between any two calculi in the $\lambda\sigma$-style.

We conclude with a summary of the results obtained.

2 Preliminaries

We introduce the notation we shall use concerning rewriting, recall the essential properties of reduction systems, present the various calculi that will be the subject of our study of efficiency and introduce the criterion of comparison.

Definition 1 *Let A be a set and R a binary relation on A, i.e. a subset of $A \times A$. We denote the fact $(a,b) \in R$ by $a \to_R b$ or $a \to b$ when the context is clear enough. We call* reduction *this relation and* reduction system, *the pair (A,R). We denote with $\stackrel{=}{\to}_R$ the reflexive closure of R, with $\twoheadrightarrow_R$ or just $\twoheadrightarrow$ the reflexive and transitive closure of R and with $\twoheadrightarrow_R^+$ or just $\twoheadrightarrow^+$ the transitive closure of R. When $a \twoheadrightarrow b$ we say there exists a* derivation *from a to b. By $a \twoheadrightarrow^n b$, we mean that the derivation consists of n steps of reduction and call n the* length *of the derivation.*

Definition 2 *Let R be a reduction on A. We define (local) confluence or (W)CR ((weakly) Church Rosser) as follows:*

1. *R is WCR when $\forall a,b,c \in A\ \exists d \in A\ ((a \to b \wedge a \to c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d))$.*
2. *R is CR when $\forall a,b,c \in A\ \exists d \in A\ ((a \twoheadrightarrow b \wedge a \twoheadrightarrow c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d))$.*

Definition 3 *Let R be a reduction on A. We say that $a \in A$ is an R-*normal form *(R-nf for short) if there exists no $b \in A$ such that $a \to b$ and we say that b* has a normal form *if there exists a nf a such that $b \twoheadrightarrow a$. R is* strongly normalising *or SN if there is no infinite sequence $(a_i)_{i\geq 0}$ in A such that $a_i \to a_{i+1}$ for all $i \geq 0$.*

Remark 1 *Confluence of R guarantees unicity of R-normal forms and SN ensures their existence. When there exists a unique R-normal form of a term a, it is denoted by $R(a)$.*

2.1 Calculi à la $\lambda\sigma$

In this section, we introduce the $\lambda\xi$-calculi (for $\xi \in \{\sigma, \sigma_{DB}, \sigma_{\Uparrow}, \upsilon\}$) which work on 2-sorted terms: *(proper) terms* and *substitutions*. The $\lambda\sigma$-calculus was introduced in [1] and the version

presented there uses only the de Bruijn index 1 and the other de Bruijn indices are coded. We introduce here another version, denoted $\lambda\sigma_{DB}$, which uses all the de Bruijn indices and hence is at the same level with the other calculi studied in this paper. We introduce $\lambda\sigma_{DB}$ because it could be argued that the coding of the de Bruijn indices could change the status of $\lambda\sigma$ with respect to efficiency results. However, we show that $\lambda\sigma$ and $\lambda\sigma_{DB}$ have the same behaviour as far as comparison of efficiency with the other calculi studied here is concerned. The $\lambda\sigma_{\Uparrow}$-calculus is a variation of the $\lambda\sigma$-calculus that is confluent on open terms (terms with variables of sort *term* and *substitution*). As all the calculi in the $\lambda\sigma$-family were shown in [12] not to possess the Preservation of Strong Normalisation property (PSN), the $\lambda\upsilon$-calculus (cf. [3]) removes the composition of substitutions to guarantee PSN.

For every ξ, we use $a, b, c, \ldots$ to range over the set of terms $\Lambda\xi^t$, and $s, t, \ldots$ to range over the set of substitutions $\Lambda\xi^s$. We use $\lambda\xi$ to denote the set of rules of the $\lambda\xi$-calculus (which contains a rule *(Beta)*) and take the ξ-calculus to be the calculus whose rules are $\lambda\xi$–$\{(Beta)\}$. The $\lambda\xi$-calculus is the reduction system $(\Lambda\xi, \rightarrow_{\lambda\xi})$, where $\rightarrow_{\lambda\xi}$ is the least compatible (with the corresponding operators) reduction on $\Lambda\xi$ generated by the set of rules $\lambda\xi$.

For every $\xi \in \{\sigma, \sigma_{\Uparrow}, \upsilon\}$, we have that (see [1, 5, 3]) the ξ-calculus is SN and the $\lambda\xi$-calculus is confluent on closed terms. Moreover, only the $\lambda\sigma_{\Uparrow}$-calculus is confluent on open terms and only the $\lambda\upsilon$-calculus satisfies PSN.

Definition 4 *(The $\lambda\sigma$-calculus) Terms and substitutions of the* $\lambda\sigma$-calculus *are given by:*

$\Lambda\sigma^t ::= \mathtt{1} \mid \Lambda\sigma^t\Lambda\sigma^t \mid \lambda\Lambda\sigma^t \mid \Lambda\sigma^t[\Lambda\sigma^s] \qquad \Lambda\sigma^s ::= id \mid \uparrow \mid \Lambda\sigma^t \cdot \Lambda\sigma^s \mid \Lambda\sigma^s \circ \Lambda\sigma^s$

For $s \in \Lambda\sigma$, s^n is defined by: $s^1 = s$, $s^{n+1} = s \circ s^n$. The index $\mathtt{n}$ *is coded as* $\mathtt{1}[\uparrow^{n-1}]$.
The set of rules $\lambda\sigma$ is given as follows:

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(VarId)	$\mathtt{1}\,[id]$	$\longrightarrow$	$\mathtt{1}$
(VarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\mathtt{1} \cdot (s \circ \uparrow)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(IdL)	$id \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ id$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$

Definition 5 *(The $\lambda\sigma_{DB}$-calculus) The syntax of $\lambda\sigma_{DB}$ is exactly that of the $\lambda\sigma$-calculus except that 1 is replaced by* $\mathbb{N}$. *The set, $\lambda\sigma_{DB}$, of rules of the* $\lambda\sigma_{DB}$-calculus *is $\lambda\sigma$ where (VarId) is replaced by $a[id] \rightarrow a$ plus the three extra rules:* $\mathtt{n}+\mathtt{1}[a \cdot s] \rightarrow \mathtt{n}[s]$, $\mathtt{n}[\uparrow] \rightarrow \mathtt{n}+\mathtt{1}$ *and* $\mathtt{n}[\uparrow \circ s] \rightarrow \mathtt{n}+\mathtt{1}[s]$.

Definition 6 *(The $\lambda\sigma_{\Uparrow}$-calculus) Terms and substitutions of the $\lambda\sigma_{\Uparrow}$-calculus are given by:*

$\Lambda\sigma^t_{\Uparrow} ::= \mathbb{N} \mid \Lambda\sigma^t_{\Uparrow}\Lambda\sigma^t_{\Uparrow} \mid \lambda\Lambda\sigma^t_{\Uparrow} \mid \Lambda\sigma^t_{\Uparrow}[\Lambda\sigma^s_{\Uparrow}] \qquad \Lambda\sigma^s_{\Uparrow} ::= id \mid \uparrow \mid \Uparrow(\Lambda\sigma^s_{\Uparrow}) \mid \Lambda\sigma^t_{\Uparrow} \cdot \Lambda\sigma^s_{\Uparrow} \mid \Lambda\sigma^s_{\Uparrow} \circ \Lambda\sigma^s_{\Uparrow}$

For $s \in \Lambda\sigma^s_{\Uparrow}$, s^n is given by: $s^1 = s$, $s^{n+1} = s \circ s^n$ and $\Uparrow^n(s)$ by: $\Uparrow^0(s) = s$, $\Uparrow^{n+1}(s) = \Uparrow(\Uparrow^n(s))$.
The set of rules $\lambda\sigma_{\Uparrow}$ is given as follows:

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b \cdot id]$
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(Clos)	$(a\,[s])[t]$	$\longrightarrow$	$a\,[s \circ t]$
(Varshift1)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n+1}$
(Varshift2)	$\mathtt{n}\,[\uparrow \circ s]$	$\longrightarrow$	$\mathtt{n+1}\,[s]$
(FVarCons)	$\mathtt{1}\,[a \cdot s]$	$\longrightarrow$	a
(RVarCons)	$\mathtt{n+1}\,[a \cdot s]$	$\longrightarrow$	$\mathtt{n}\,[s]$
(FVarLift1)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(FVarLift2)	$\mathtt{1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{1}\,[t]$
(RVarLift1)	$\mathtt{n+1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}[s \circ \uparrow]$
(RVarLift2)	$\mathtt{n+1}\,[\Uparrow(s) \circ t]$	$\longrightarrow$	$\mathtt{n}[s \circ (\uparrow \circ t)]$
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a\,[t] \cdot (s \circ t)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(ShiftLift1)	$\uparrow \circ \Uparrow(s)$	$\longrightarrow$	$s \circ \uparrow$
(ShiftLift2)	$\uparrow \circ (\Uparrow(s) \circ t)$	$\longrightarrow$	$s \circ (\uparrow \circ t)$
(Lift1)	$\Uparrow(s) \circ \Uparrow(t)$	$\longrightarrow$	$\Uparrow(s \circ t)$
(Lift2)	$\Uparrow(s) \circ (\Uparrow(t) \circ u)$	$\longrightarrow$	$\Uparrow(s \circ t) \circ u$
(LiftEnv)	$\Uparrow(s) \circ (a \cdot t)$	$\longrightarrow$	$a \cdot (s \circ t)$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(LiftId)	$\Uparrow(id)$	$\longrightarrow$	id
(Id)	$a\,[id]$	$\longrightarrow$	a

Definition 7 *(The λv-calculus) Terms and substitutions of the λv-calculus are given by:*

$\Lambda v^t ::= \mathbb{N} \mid \Lambda v^t \Lambda v^t \mid \lambda \Lambda v^t \mid \Lambda v^t[\Lambda v^s] \qquad \Lambda v^s ::= \uparrow \mid \Uparrow(\Lambda v^s) \mid \Lambda v^t$

For $a \in \Lambda v^t$, $s \in \Lambda v^s$, $\Uparrow^n(s)$ is given by: $\Uparrow^0(s) = s$, $\Uparrow^{n+1}(s) = \Uparrow(\Uparrow^n(s))$ and $a[s]^i$ by: $a[s]^0 = a$, $a[s]^{n+1} = (a[s]^n)[s]$. The set of rules λv is given as follows:

(Beta)	$(\lambda a)\, b$	$\longrightarrow$	$a\,[b/]$
(App)	$(a\, b)[s]$	$\longrightarrow$	$(a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a\,[\Uparrow(s)])$
(FVar)	$\mathtt{1}\,[a/]$	$\longrightarrow$	a
(RVar)	$\mathtt{n+1}\,[a/]$	$\longrightarrow$	$\mathtt{n}$
(FVarLift)	$\mathtt{1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{1}$
(RVarLift)	$\mathtt{n+1}\,[\Uparrow(s)]$	$\longrightarrow$	$\mathtt{n}\,[s]\,[\uparrow]$
(VarShift)	$\mathtt{n}\,[\uparrow]$	$\longrightarrow$	$\mathtt{n+1}$

2.2 Calculi à la λs

Calculi à la λs avoid introducing two different sets of entities and insist on remaining close to the syntax of the λ-calculus. Next to λ and abstraction, they introduce substitution (σ, ς) and

updating (φ, θ) operators. We shall introduce three such calculi: λs, λt and λu. We let a, b, c, etc. range over the sets of terms Λs, Λt and Λu. A term containing neither substitution nor updating operators is called a *pure term*. For $\xi \in \{s, t, u\}$, the $\lambda\xi$- and ξ-calculi are defined as in the previous section (take σ- or ς-*generation* instead of *Beta*) from a set of rules $\lambda\xi$ or ξ.

The λs-calculus was introduced in [8] with the aim of providing a calculus that preserves strong normalisation and has a confluent extension on open terms [11]. The λt-calculus is a variant of λs that updates partially, as the $\lambda\sigma$-calculi do. The λu-calculus is introduced here for the first time and is only a slight (yet more efficient) variation of λs.

For $\xi \in \{s, t, u\}$, we have that (see [8, 10, 9]) the ξ-calculus is SN, the $\lambda\xi$-calculus is confluent on closed terms and satisfies PSN. Moreover, the $\lambda\xi$-calculus for $\xi \in \{s, u\}$ has a confluent extension on open terms.

Definition 8 *(The λs-calculus) Terms of the λs-calculus are given by:*

$\Lambda s ::= \mathbb{N} \mid \Lambda s \Lambda s \mid \lambda \Lambda s \mid \Lambda s \, \sigma^i \Lambda s \mid \varphi^i_k \Lambda s \quad$ *where* $\quad i \geq 1\,, \;\; k \geq 0\,.$

and the set of rules λs is given as follows:

σ-*generation*	$(\lambda a)\, b$	$\longrightarrow$	$a \, \sigma^1 \, b$
σ-λ-*transition*	$(\lambda a)\, \sigma^i b$	$\longrightarrow$	$\lambda (a \, \sigma^{i+1} \, b)$
σ-*app-transition*	$(a_1 \, a_2)\, \sigma^i b$	$\longrightarrow$	$(a_1 \, \sigma^i b)\, (a_2 \, \sigma^i b)$
σ-*destruction*	$\mathtt{n}\, \sigma^i b$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } \; n > i \\ \varphi^i_0 \, b & \text{if } \; n = i \\ \mathtt{n} & \text{if } \; n < i \end{cases}$
φ-λ-*transition*	$\varphi^i_k (\lambda a)$	$\longrightarrow$	$\lambda (\varphi^i_{k+1} \, a)$
φ-*app-transition*	$\varphi^i_k (a_1 \, a_2)$	$\longrightarrow$	$(\varphi^i_k \, a_1)\, (\varphi^i_k \, a_2)$
φ-*destruction*	$\varphi^i_k \, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \text{if } \; n > k \\ \mathtt{n} & \text{if } \; n \leq k \end{cases}$

Definition 9 *(The λt-calculus) Terms of the λt-calculus are given by:*

$\Lambda t ::= \mathbb{N} \mid \Lambda t \Lambda t \mid \lambda \Lambda t \mid \Lambda t \, \varsigma^i \Lambda t \mid \theta_k \Lambda t \quad$ *where* $\quad i \geq 1\,, \;\; k \geq 0\,.$

For $a \in \Lambda t$, $\theta^i_k a$ is given by: $\theta^0_k a = a$, $\theta^{i+1}_k (a) = \theta_k (\theta^i_k (a))$.
The set of rules λt is given as follows:

ς-*generation*	$(\lambda a)\, b$	$\longrightarrow$	$a \, \varsigma^1 \, b$
ς-λ-*transition*	$(\lambda a)\, \varsigma^i b$	$\longrightarrow$	$\lambda (a \, \varsigma^{i+1} \, \theta_0 (b))$
ς-*app-transition*	$(a_1 \, a_2)\, \varsigma^i b$	$\longrightarrow$	$(a_1 \, \varsigma^i b)\, (a_2 \, \varsigma^i b)$
ς-*destruction*	$\mathtt{n}\, \varsigma^i b$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } \; n > i \\ b & \text{if } \; n = i \\ \mathtt{n} & \text{if } \; n < i \end{cases}$
θ-λ-*transition*	$\theta_k (\lambda a)$	$\longrightarrow$	$\lambda (\theta_{k+1} \, a)$
θ-*app-transition*	$\theta_k (a_1 \, a_2)$	$\longrightarrow$	$(\theta_k \, a_1)\, (\theta_k \, a_2)$
θ-*destruction*	$\theta_k \, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+1} & \text{if } \; n > k \\ \mathtt{n} & \text{if } \; n \leq k \end{cases}$

The main difference between λt and λs can be summarized as follows: the λt-calculus generates a partial updating when a substitution is evaluated on an abstraction (i.e. introduces an operator θ_0 in the ς-λ-*transition* rule) whereas the λs-calculus produces a global updating when performing substitutions (i.e. introduces a φ_0^i operator in the σ-*destruction* rule, case $n = i$). The λt-calculus shares this mechanism of partial updatings with the $\lambda\sigma$-caculi, λv and $\lambda\zeta$ since all of them introduce an updating operator in their *(Abs)*-rule.

Definition 10 *(The λu-calculus) Terms of the λu-calculus are given by:*

$\Lambda u ::= \mathbb{N} \mid \Lambda u \Lambda u \mid \lambda \Lambda u \mid \Lambda u \, \sigma^j \Lambda u \mid \varphi_k^i \Lambda u \quad$ *where* $i \geq 2,\ j \geq 1,\ k \geq 0$.

and the set of rules λu is given as follows:

σ-*generation*	$(\lambda a)\, b$	$\longrightarrow$	$a \, \sigma^1 \, b$
σ-λ-*transition*	$(\lambda a)\, \sigma^i b$	$\longrightarrow$	$\lambda (a \, \sigma^{i+1} \, b)$
σ-*app-transition*	$(a_1 \, a_2)\, \sigma^i b$	$\longrightarrow$	$(a_1 \, \sigma^i b)\, (a_2 \, \sigma^i b)$
σ-*destruction*	$\mathtt{n}\, \sigma^i b$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } n > i \\ \varphi_0^i \, b & \text{if } n = i > 1 \\ b & \text{if } n = i = 1 \\ \mathtt{n} & \text{if } n < i \end{cases}$
φ-λ-*transition*	$\varphi_k^i (\lambda a)$	$\longrightarrow$	$\lambda (\varphi_{k+1}^i \, a)$
φ-*app-transition*	$\varphi_k^i (a_1 \, a_2)$	$\longrightarrow$	$(\varphi_k^i \, a_1)\, (\varphi_k^i \, a_2)$
φ-*destruction*	$\varphi_k^i \, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k \end{cases}$

The only difference between λs and λu is that in the σ-*destruction* rule the case $n = i = 1$ is treated in a more efficient way by λu, which does not introduce the operator φ_0^1 since the computation $\varphi_0^1(b)$ will finally evaluate to b.

2.3 The criterion

We give now a formal presentation of the criterion we use to compare the different calculi.

Definition 11 *Let $a, b \in \Lambda$ such that $a \to_\beta b$, a* simulation *of this β-reduction in $\lambda\xi$ for $\xi \in \{\sigma, \sigma_\Uparrow, v, s, t, u\}$ is a $\lambda\xi$-derivation $a \to_r c \twoheadrightarrow_\xi \xi(c) = b$ where r is the rule starting β ((Beta) for the calculi in the $\lambda\sigma$-style and σ- or ς-generation for the calculi in the λs-style) applied to the same redex as the redex in $a \to_\beta b$. We say that the $\lambda\xi$-calculus* simulates β-reduction *if every β-reduction $a \to_\beta b$ has a simulation in $\lambda\xi$.*

The following was shown for each of the calculi we consider (see the relevant articles):

Lemma 1 *For $\xi \in \{\sigma, \sigma_\Uparrow, v, s, t, u\}$, $\lambda\xi$ simulates β-reduction.*

Definition 12 *Let $\xi_1, \xi_2 \in \{\sigma, \sigma_\Uparrow, v, s, t, u\}$. The $\lambda\xi_1$-calculus is* more efficient (in simulating one step β-reductions) than *the $\lambda\xi_2$-calculus, denoted $\lambda\xi_1 \prec \lambda\xi_2$ if*

1. *for every classical β-reduction $a \to_\beta b$ and every $\lambda\xi_2$-simulation $a \twoheadrightarrow^n_{\lambda\xi_2} b$ there exists a $\lambda\xi_1$-simulation $a \twoheadrightarrow^m_{\lambda\xi_1} b$ such that $m \leq n$.*
2. *there exist a classical β-reduction $a \to_\beta b$ and a $\lambda\xi_1$-simulation $a \twoheadrightarrow^m_{\lambda\xi_1} b$ such that for every $\lambda\xi_2$-simulation $a \twoheadrightarrow^n_{\lambda\xi_2} b$ we have $m < n$.*

It is easy to verify that $\prec$ is transitive and asymetric.

3 Establishing efficiency

In this section we put the criterion at work. The main idea is to define functions (denoted with Q) which evaluate the length of the derivations of certain families of terms that contain the contracta of the *(Beta)*- rules (eg. $a[b/]$ in λv). For λv it is possible to prove that all these derivations have the same length, whereas for $\lambda\sigma_{\Uparrow}$ our functions compute just the length of the shortest derivation. To define these Q-functions we need to define another functions (denoted with M) which evalute the length of the derivations of updatings. For the scope of this section, only the M-functions are needed for λt and λu.

3.1 λt is more efficient than λv

We introduce a set of terms $\Lambda_\theta \subset \Lambda t$ on which induction will be used to define M^t (a function that computes the length of derivations of updatings in λt). We are mainly interested in pure terms, which are contained in Λ_θ, but the introduction of Λ_θ is necessary since it provides a strong induction hypothesis to prove the auxiliary results needed.

Definition 13 $\Lambda_\theta ::= \mathbb{N} \mid \Lambda_\theta\Lambda_\theta \mid \lambda\Lambda_\theta \mid \theta_k\Lambda_\theta$, *where* $k \geq 0$. *The* length *of terms in* Λ_θ *is defined by:* $L_\theta(\mathtt{n}) = 1$, $L_\theta(ab) = L_\theta(a) + L_\theta(b) + 1$, $L_\theta(\lambda a) = L_\theta(\theta_k a) = L_\theta(a) + 1$.
By induction on $a \in \Lambda_\theta$ *we mean induction on* $L_\theta(a)$.

Remark 2 *Let* $a \in \Lambda_\theta$ *and* $k \geq 0$, *then* $L_\theta(a) \geq L_\theta(t(\theta_k a))$.

Proof: By induction on a. The interesting case is when $a = \theta_m b$. By IH we have $L_\theta(b) \geq L_\theta(t(\theta_m b))$ and since $L_\theta(a) > L_\theta(b)$, we apply again the IH (now to $t(\theta_m b)$) to obtain $L_\theta(t(\theta_m b)) \geq L_\theta(t(\theta_k(t(\theta_m b)))) = L_\theta(t(\theta_k(\theta_m b)))$. Hence, $L_\theta(a) \geq L_\theta(t(\theta_k a))$. □

The next remark will be used frequently without explicit mention.

Remark 3 *If* $a \in \Lambda_\theta$ *and* $a \to_t b$ *then* $b \in \Lambda_\theta$.

Proof: Easy induction on a. □

Definition 14 *We define* $M^t : \Lambda_\theta \to \mathbb{N}$ *by induction as follows:*
$M^t(\mathtt{n}) = 1 \quad M^t(ab) = M^t(a) + M^t(b) + 1 \quad M^t(\lambda a) = M^t(a) + 1 \quad M^t(\theta_k a) = M^t(t(\theta_k a)) + M^t(a)$

Remark that the definition is correct thanks to remark 2.

Lemma 2 *For* $a \in \Lambda_\theta$, *every* t*-derivation of* $\theta_k a$ *to its* t*-normal form has length* $M^t(a)$.

Proof: By induction on the weight $P(a)$ used to prove SN for the t-calculus (see [10]). The basic case ($a = \mathtt{n}$) is immediate, since all the derivations of $\theta_k\mathtt{n}$ to its nf have length 1. We proceed now by a case analysis. We just treat the case $a = bc$ since the argument is similar for the other cases.

Let us consider a derivation $\mathcal{D}$ of $\theta_k(bc)$ to its nf.

If the first step is internal, say $b \to b'$, we know by IH ($P(b'c) < P(bc)$) that every derivation of $\theta_k(b'c)$ to its nf has length $M^t(b'c) = M^t(b') + M^t(c) + 1$. But IH (now applied to b ($P(b) < P(bc)$) and b' ($P(b') < P(bc)$) and the fact that $\theta_k b \to \theta_k b'$) also gives $M^t(b') = M^t(b) - 1$. Hence $M^t(b'c) = M^t(b) + M^t(c) = M^t(bc) - 1$. Therefore, the length of $\mathcal{D}$ is $M^t(bc)$.

If the first step is $\theta_k(bc) \to \theta_k(b)\theta_k(c)$, since there are no rules in t which contract an application, every derivation of $\theta_k(b)\theta_k(c)$ to its nf, has length (IH applied to b and c) $M^t(b) + M^t(c) = M^t(bc) - 1$. Therefore, the length of $\mathcal{D}$ is again $M^t(bc)$. □

Corollary 1 *For $a \in \Lambda_\theta$, all the t-derivations of $\theta_k^i a$ to its t-normal form have the same length, namely $(i-1)M^t(t(a)) + M^t(a)$.*

Proof: Prove first by induction on $a \in \Lambda_\theta$, using Remark 2, that $M^t(t(a)) = M^t(t(\theta_k a))$, then use this result to prove, by induction on $j \geq 1$ that $M^t(t(a)) = M^t(t(\theta_k^j a))$. Use now Definition 14 and the two previous results to show, by induction on $l \geq 1$, that $M^t(\theta_k^l(a)) = lM^t(t(a)) + M^t(a)$. Finally, use Lemma 2 and the last result with $l = i-1$ to prove the corollary. Remark that it is in this proof that the hypothesis $a \in \Lambda_\theta$ is essential and hence the necessity of Definition 13. □

Now we are going to prove the corresponding results for $\lambda\upsilon$. Since the proofs are analogous, we just state the results.

Definition 15 $\Lambda_\uparrow ::= \mathbb{N} \mid \Lambda_\uparrow\Lambda_\uparrow \mid \lambda\Lambda_\uparrow \mid \Lambda_\uparrow[\Uparrow^k(\uparrow)]$, *where $k \geq 0$. The* length *of terms in $\Lambda_\uparrow$ is given by:* $L_\uparrow(\mathtt{n}) = 1 \quad L_\uparrow(ab) = L_\uparrow(a) + L_\uparrow(b) + 1 \quad L_\uparrow(\lambda a) = L_\uparrow(a[\Uparrow^k(\uparrow)]) = L_\uparrow(a) + 1$.

Remark 4 *Let $a \in \Lambda_\uparrow$ and $k \geq 0$, then $L_\uparrow(a) \geq L_\uparrow(\upsilon(a[\Uparrow^k(\uparrow)]))$.*

Remark 5 *If $a \in \Lambda_\uparrow$ and $a \to_\upsilon b$ then $b \in \Lambda_\uparrow$.*

Definition 16 *For $k \geq 0$, we define $M_k^\upsilon : \Lambda_\theta \to \mathbb{N}$ as follows:*

$$M_k^\upsilon(\mathtt{n}) = \begin{cases} 2k+1 & \text{if } n > k \\ 2n-1 & \text{if } n \leq k \end{cases} \qquad \begin{array}{l} M_k^\upsilon(ab) = M_k^\upsilon(a) + M_k^\upsilon(b) + 1 \qquad M_k^\upsilon(\lambda a) = M_{k+1}^\upsilon(a) + 1 \\ M_k^\upsilon(a[\Uparrow^p(\uparrow)]) = M_k^\upsilon(\upsilon(a[\Uparrow^p(\uparrow)])) + M_p^\upsilon(a) \end{array}$$

Lemma 3 *For $a \in \Lambda_\uparrow$, all the υ-derivations of $a[\Uparrow^k(\uparrow)]$ to its υ-nf have length $M_k^\upsilon(a)$.*

Proof: By induction on the weight used to show SN for the υ-calculus (cf. [3]) and case analysis. □

Corollary 2 *For $a \in \Lambda_\uparrow$, all the υ-derivations of $a[\Uparrow^k(\uparrow)]^i$ to its υ-normal form have the same length, namely $(i-1)M_k^\upsilon(\upsilon(a)) + M_k^\upsilon(a)$.*

Definition 17 *Let $a, b \in \Lambda$ and $i \geq 0$, we define $Q_i^\upsilon(a,b)$ by induction on a:*

$$Q_i^\upsilon(\mathtt{n}, b) = \begin{cases} 2i+1 & \text{if } n > i+1 \\ 2n-1 & \text{if } n < i+1 \\ i(1+M_0^\upsilon(b))+1 & \text{if } n = i+1 \end{cases} \qquad \begin{array}{l} Q_i^\upsilon(cd,b) = Q_i^\upsilon(c,b) + Q_i^\upsilon(d,b) + 1 \\ Q_i^\upsilon(\lambda c, b) = Q_{i+1}^\upsilon(c,b) + 1 \end{array}$$

Lemma 4 *Let $a, b \in \Lambda$ and $i \geq 0$, all the υ-derivations of $a[\Uparrow^i(b/)]$ to its υ-nf have the same length, namely $Q_i^\upsilon(a,b)$.*

Proof: Easy induction on $a \in \Lambda$. Remark that for $a = \mathtt{n}$ there is only one derivation whose length is easy to compute. When $n = i+1$, use Corollary 2. □

Lemma 5 *Let $a, b \in \Lambda$ and $i \geq 0$, there exists a derivation of $a\varsigma^{i+1}(\theta_0^i b)$ to its t-nf whose length is less than or equal to $Q_i^\upsilon(a,b)$.*

Proof: By induction on a reducing always at the root. For the case $a = \mathtt{i+1}$ use the fact that $M_0^\upsilon(b) \geq M^t(b)$ (induction on $b \in \Lambda$) and Corollary 1. □

Theorem 1 *λt is more efficient than λv.*

Proof: We prove that for every $a \in \Lambda$ and every λv-derivation $a \to_B b \twoheadrightarrow_v^m v(b)$ there exists $n \leq m$ such that $a \to_{\varsigma-gen} c \twoheadrightarrow_t^n t(c)$ by induction on a.

The interesting case is $a = (\lambda d)e \to_B d[e/] \twoheadrightarrow^m v(d[e/])$. By Lemma 4 we know that $m = Q_0^v(d,e)$ and Lemma 5 gives a derivation $d\,\varsigma^1 e \twoheadrightarrow_t^n t(d\,\varsigma^1 e)$ such that $n \leq Q_0^v(d,e)$.

To check the second condition in Definition 12 remark that there are an infinity of cases for which the inequality is strict. For instance, let us consider the term $(\lambda\lambda\ldots\lambda\mathtt{n})a$ with m λ's and $n > m > 1$. It is easy to check, using the function Q_{m-1}^v defined above that $3m-2$ reductions are needed to simulate β-reduction in λv, whereas only $m+1$ reductions are sufficient in λt. Remark that for $m > n$ the number of reductions needed in λv is also strictly greater than the number needed in λt. □

3.2 λu is more efficient than $\lambda\sigma_{\Uparrow}$

Definition 18 *For $k \geq 0$ and $i \geq 1$, we define $M_{ki}^{\Uparrow} : \Lambda \to \mathbb{N}$ by induction as follows:*

$$M_{ki}^{\Uparrow}(\mathtt{n}) = \begin{cases} 2n-1 & \text{if } n < k+1 \\ 2(k+i)-1 & \text{if } n \geq k+1 \end{cases} \qquad \begin{array}{l} M_{ki}^{\Uparrow}(ab) = M_{ki}^{\Uparrow}(a) + M_{ki}^{\Uparrow}(b) + 1 \\ M_{ki}^{\Uparrow}(\lambda a) = M_{k+1\,i}^{\Uparrow}(a) + 1 \end{array}$$

Lemma 6 *For $a \in \Lambda$, every $\sigma_{\Uparrow}$-derivation of $a[\Uparrow^k (\uparrow^i)]$ to its $\sigma_{\Uparrow}$-nf has length $M_{ki}^{\Uparrow}(a)$.*

Proof: By induction on a controlling all the possible $\sigma_{\Uparrow}$-derivations. □

Definition 19 *For $k \geq 0$ and $i \geq 1$, we define $Q_k^{\Uparrow} : \Lambda \times \Lambda \to \mathbb{N}$ by induction as follows:*

$$Q_k^{\Uparrow}(\mathtt{n}, c) = \begin{cases} 2n-1 & \text{if } n < k+1 \\ M_{0\,n-1}^{\Uparrow}(c)+n+1 & \text{if } n = k+1,\ k > 0 \\ 1 & \text{if } n = 1,\ k = 0 \\ 2k+3 & \text{if } n > k+1 \end{cases} \qquad \begin{array}{l} Q_k^{\Uparrow}(ab, c) = Q_k^{\Uparrow}(a, c) + Q_k^{\Uparrow}(b, c) + 1 \\ Q_k^{\Uparrow}(\lambda a, c) = Q_{k+1}^{\Uparrow}(a, c) + 1 \end{array}$$

Lemma 7 *If $a, b \in \Lambda$, the shortest $\sigma_{\Uparrow}$-derivation of $a[\Uparrow^k (b \cdot id)]$ to its $\sigma_{\Uparrow}$-nf has length $Q_k^{\Uparrow}(a,b)$.*

Proof: By induction on a controlling all the possible $\sigma_{\Uparrow}$-derivations. □

Definition 20 *For $k \geq 0$ and $i \geq 2$, we define $M^u : \Lambda \to \mathbb{N}$ by induction as follows:*

$$M^u(\mathtt{n}) = 1 \qquad M^u(ab) = M^u(a) + M^u(b) + 1 \qquad M^u(\lambda a) = M^u(a) + 1$$

Lemma 8 *For $a \in \Lambda$, every u-derivation of $\varphi_k^i a$ to its u-normal form has length $M^u(a)$.*

Proof: By induction on a. Remark that every derivation of $\varphi_k^i a$ must begin with a reduction at the root since $a \in \Lambda$. □

Lemma 9 *For every $a, b \in \Lambda$, $k \geq 0$ there exists a u-derivation of $a\sigma^{k+1}b$ to its u-nf whose length is less than or equal to $Q_k^{\Uparrow}(a,b)$.*

Proof: By induction on a. The interesting case is $a = \mathtt{k+1}$ and the result follows from Lemmas 6, 8 and the fact $M^u(b) \leq M_{0i}^{\Uparrow}(b)$, which is easily proved by induction on b. □

Theorem 2 *λu is more efficient than $\lambda\sigma_{\Uparrow}$.*

Proof: We prove that for every $a \in \Lambda$ and every $\lambda\sigma_{\Uparrow}$-derivation $a \to_{Beta} b \twoheadrightarrow^m_{\sigma_{\Uparrow}} \sigma_{\Uparrow}(b)$ there exists $n \leq m$ such that $a \to_{\sigma-gen} c \twoheadrightarrow^n_u u(c)$ by induction on a.

The interesting case is $a = (\lambda d)e \to_{Beta} d[e \cdot id] \twoheadrightarrow^m \sigma_{\Uparrow}(d[e \cdot id])$. By Lemma 7 we know that $m \geq Q^{\Uparrow}_0(d, e)$ and Lemma 9 gives a derivation $d\,\sigma^1 e \twoheadrightarrow^n_u u(d\,\sigma^1 e)$ such that $n \leq Q^{\Uparrow}_0(d, e)$.

Now, to check the second condition in Definition 12, it is easy to compute to 6 the length of the shortest simulation in $\lambda\sigma_{\Uparrow}$ (there are only 2 such simulations) of the β-reduction $(\lambda\lambda\mathtt{2})\mathtt{1} \to \lambda\mathtt{2}$, whereas the only simulation of this reduction in λu has length 4. □

3.3 λu is more efficient than λv

In this section we use the functions defined in the two previous sections to prove that λu is more efficient than λv.

Lemma 10 *For every $a, b \in \Lambda$, $i \geq 0$ there exists a u-derivation of $a\sigma^{i+1}b$ to its u-nf whose length is less than or equal to $Q^v_i(a, b)$.*

Proof: By induction on a. The interesting case is $a = \mathtt{i} + \mathtt{1}$ and the result follows from Corollary 2, Lemma 8 and the fact $M^u(b) \leq i(1 + M^v_0(b))$, proved by induction on b. □

Theorem 3 *λu is more efficient than λv.*

Proof: Analogous to the proof of Theorem 2. Just check that the only simulation of $(\lambda\lambda\mathtt{2})\mathtt{1} \to \lambda\mathtt{2}$ in λv has length 5. □

3.4 λu is more efficient than λs

The proof of efficiency in this section is simpler than the previous ones since λu and λs are closely related. We need first an easy lemma:

Lemma 11 *For $i \geq 2$ and $b \in \Lambda$ every s-derivation of $\varphi^i_0(b)$ to its s-nf is also a u-derivation.*

Proof: Easy induction on b. □

Lemma 12 *For every $a, b \in \Lambda$, $i \geq 1$ and s-derivation of $a\,\sigma^i b$ to its s-nf of length m, there exists an u-derivation of $a\,\sigma^i b$ to its u-nf whose length is less than or equal to m.*

Proof: By induction on a. The interesting case is $i > 1$ and $a = \mathtt{i}$. The result follows from Lemma 11 which gives a u-derivation of the same length. Remark that we have a strict inequality when $i = 1$ and $a = \mathtt{i}$. □

Theorem 4 *λu is more efficient than λs.*

Proof: Show, as in Thm. 2, that for every $a \in \Lambda$ and every λs-derivation $a \to_{\sigma-gen} b \twoheadrightarrow^m_s s(b)$ there exists $n \leq m$ such that $a \to_{\sigma-gen} b \twoheadrightarrow^n_u u(c)$ by induction on a.

To check the second condition, consider the β-reduction $(\lambda\mathtt{1})\mathtt{1} \to \mathtt{1}$. There is only one simulation in λs with length 4 and there is only one simulation in λu with length 3. □

4 Non-comparable calculi

To show that two calculi, say $\lambda\xi_1$ and $\lambda\xi_2$ cannot be compared with our criterion it is enough to find two classical β-reductions $a \to_\beta b$ and $c \to_\beta d$ such that

1. There is a shorter simulation $a \twoheadrightarrow_{\lambda\xi_1} b$ than the shortest simulation $a \twoheadrightarrow_{\lambda\xi_2} b$.
2. There is a shorter simulation $c \twoheadrightarrow_{\lambda\xi_2} d$ than the shortest simulation $c \twoheadrightarrow_{\lambda\xi_1} d$.

If this is the case we say that $\lambda\xi_1$ and $\lambda\xi_2$ are *incomparable*, and we write $\lambda\xi_1 \not\leftrightarrow \lambda\xi_2$.

Since $\lambda\sigma$ works in a more "atomized" way (the $\Uparrow$-operator of $\lambda\sigma_\Uparrow$ and $\lambda\upsilon$ may be decomposed in $\lambda\sigma$ as $\Uparrow(s) = 1 \cdot (s \circ \uparrow)$ and the /-operator of $\lambda\upsilon$ may be decomposed in $\lambda\sigma$ as $a/ = a \cdot id$) it is tempting to assume that $\lambda\sigma$, even its version with uncoded de Bruijn indices, would be less efficient than $\lambda\upsilon$ and $\lambda\sigma_\Uparrow$. However this is not the case. As a matter of fact there is an infinite family of terms for which $\lambda\sigma$ performs better than $\lambda\upsilon$ and $\lambda\sigma_\Uparrow$, and furthermore, for these terms, $\lambda\sigma$ also performs better than λs and λu.

The terms we are going to consider are $(\lambda\lambda(2\,2))1^n$, where a^n is defined by induction on n as $a^1 = a$, $a^{n+1} = a\,a^n$. There is only one β-redex at the root and $(\lambda\lambda(2\,2))1^n \to_\beta \lambda(2^n 2^n)$. We study now the simulation of this β-reduction in the different calculi.

Lemma 13 *There is a $\lambda\sigma$-derivation of $(\lambda\lambda(2\,2))1^n$ to its $\lambda\sigma$-nf whose length is $n+9$ and a $\lambda\sigma_{DB}$-derivation whose length is $2n+7$.*

Proof: Here is the derivation in $\lambda\sigma$:

$(\lambda\lambda(2\,2))1^n = (\lambda\lambda(1[\uparrow]\,1[\uparrow]))1^n \to (\lambda(1[\uparrow]\,1[\uparrow]))[1^n \cdot id] \to \lambda((1[\uparrow]\,1[\uparrow])[1 \cdot ((1^n \cdot id) \circ \uparrow)]) \to$
$\lambda((1[\uparrow]\,1[\uparrow])[1 \cdot (1^n[\uparrow] \cdot (id \circ \uparrow))]) \twoheadrightarrow^{n-1} \lambda((1[\uparrow]\,1[\uparrow])[1 \cdot ((1[\uparrow])^n \cdot (id \circ \uparrow))]) \to$
$\lambda((1[\uparrow][1 \cdot (1[\uparrow])^n \cdot (id \circ \uparrow)])\,(1[\uparrow][1 \cdot (1[\uparrow])^n \cdot (id \circ \uparrow)])) \to \lambda((1[\uparrow \circ (1 \cdot (1[\uparrow])^n \cdot (id \circ \uparrow))])\,(1[\uparrow][1 \cdot (1[\uparrow])^n \cdot (id \circ \uparrow)])) \to$
$\lambda((1[(1[\uparrow])^n \cdot (id \circ \uparrow)])\,(1[\uparrow][1 \cdot ((1[\uparrow])^n \cdot (id \circ \uparrow))])) \to \lambda((1[\uparrow])^n\,(1[\uparrow][1 \cdot ((1[\uparrow])^n \cdot (id \circ \uparrow))])) \twoheadrightarrow^3$
$\lambda((1[\uparrow])^n (1[\uparrow])^n) = \lambda(2^n 2^n)$

Here is the derivation in $\lambda\sigma_{DB}$:

$(\lambda\lambda(2\,2))1^n \to (\lambda(2\,2))[1^n \cdot id] \to \lambda((2\,2)[1 \cdot ((1^n \cdot id) \circ \uparrow)]) \to \lambda((2\,2)[1 \cdot (1^n[\uparrow] \cdot (id \circ \uparrow))]) \twoheadrightarrow^{n-1}$
$\lambda((2\,2)[1 \cdot ((1[\uparrow])^n \cdot (id \circ \uparrow))]) \twoheadrightarrow^n \lambda((2\,2)[1 \cdot (2^n \cdot (id \circ \uparrow))]) \to \lambda((2[1 \cdot (2^n \cdot (id \circ \uparrow))])\,(2[1 \cdot (2^n \cdot (id \circ \uparrow))])) \to$
$\lambda((1[2^n \cdot (id \circ \uparrow)])\,(2[1 \cdot (2^n \cdot (id \circ \uparrow))])) \to \lambda(2^n\,(2[1 \cdot (2^n \cdot (id \circ \uparrow))])) \twoheadrightarrow^2 \lambda(2^n 2^n)$ □

Lemma 14 *Every $\lambda\upsilon$-derivation of $(\lambda\lambda(2\,2))1^n$ to its $\lambda\upsilon$-nf has length $4n+5$.*

Proof: Every derivation of $(\lambda\lambda(2\,2))1^n$ must begin as follows:

$(\lambda\lambda(2\,2))1^n \to (\lambda(2\,2))[1^n/] \to \lambda((2\,2)[\Uparrow(1^n/)]) \to \lambda((2[\Uparrow(1^n/)])\,(2[\Uparrow(1^n/)]))$

Now, the two occurrences of $2[\Uparrow(1^n/)]$ cannot interact since no abstraction will ever appear in the first occurrence. Therefore, it is enough to show that every derivation of $2[\Uparrow(1^n/)]$ has length $2n+1$. But this is a consequence of Lemma 4 and the fact that $M_0^\upsilon(1^n) = 2n-1$, which is easily shown by induction on n. □

Lemma 15 *Every λu-derivation of $(\lambda\lambda(2\,2))1^n$ to its λu-nf has length $4n+3$.*

Proof: Every λu-derivation of $(\lambda\lambda(2\,2))1^n$ must begin as follows:

$(\lambda\lambda(2\,2))1^n \to (\lambda(2\,2))\sigma^1 1^n \to \lambda((2\,2)\sigma^2 1^n) \to \lambda((2\sigma^2 1^n)\,(2\sigma^2 1^n))$

Now, the two occurrences of $2\sigma^2 1^n$ cannot interact and therefore, it is enough to show that every derivation of $2\sigma^2 1^n$ has length $2n$. There is only one redex in $2\sigma^2 1^n$, whose contraction gives $\varphi_0^2(1^n)$ and by Lemma 8 every derivation of $\varphi_0^2(1^n)$ has length $M^u(1^n)$ which is easily computable to $2n-1$ by induction on n. □

Lemma 16 *For $a \in \Lambda$, every s-derivation of $\varphi^i_k a$ to its s-normal form has length $M^u(a)$.*

Proof: By induction on a. Identical to the proof of Lemma 8. □

Lemma 17 *Every λs-derivation of $(\lambda\lambda(2\,2))1^n$ to its λs-nf has length $4n+3$.*

Proof: Analogous to the proof of Lemma 15, using Lemma 16. □

Lemma 18 *There is a λt-derivation of $(\lambda\lambda(2\,2))1^n$ to its λt-nf whose length is $2n+4$.*

Proof: Here is the derivation in λt: $(\lambda\lambda(2\,2))1^n \to (\lambda(2\,2))\sigma^1 1^n \to \lambda((2\,2)\varsigma^2\theta_0(1^n)) \twoheadrightarrow^{n-1} \lambda((2\,2)\varsigma^2(\theta_0 1)^n) \twoheadrightarrow^n \lambda((2\,2)\varsigma^2 2^n) \to \lambda((2\varsigma^2 2^n)\,(2\varsigma^2 2^n)) \twoheadrightarrow^2 \lambda(2^n 2^n)$ □

Lemma 19 *The shortest $\lambda\sigma_{\Uparrow}$-derivation of $(\lambda\lambda(2\,2))1^n$ to its $\lambda\sigma_{\Uparrow}$-nf has length $4n+7$.*

Proof: Every $\lambda\sigma_{\Uparrow}$-derivation of $(\lambda\lambda(2\,2))1^n$ must begin as follows:

$(\lambda\lambda(2\,2))1^n \to (\lambda(2\,2))[1^n \cdot id] \to \lambda((2\,2)[\Uparrow (1^n \cdot id)]) \to \lambda((2[\Uparrow (1^n \cdot id)])\,(2[\Uparrow (1^n \cdot id)]))$

Now, the two occurrences of $2[\Uparrow (1^n \cdot id)]$ cannot interact and therefore, it is enough to verify that the shortest derivation of $2[\Uparrow (1^n \cdot id)]$ to its $\lambda\sigma_{\Uparrow}$-nf has length $2n+2$. This is easily done using Lemma 7 and the fact that $M^{\Uparrow}_{01}(1^n) = 2n-1$, proved by induction on n. □

4.1 λu and λt are incomparable

Lemmas 15 and 18 prove that the reductions $(\lambda\lambda(2\,2))1^n \to \lambda(2^n 2^n)$ with $n \geq 1$ show $\lambda u \not\prec \lambda t$.

On the other hand, $(\lambda\lambda\lambda 3)1 \to \lambda\lambda 3$ shows that $\lambda t \not\prec \lambda u$. In fact, it is easy to check that every simulation (there are 5) in λt of $(\lambda\lambda\lambda 3)1 \to \lambda\lambda 3$ has length 6, whereas in λu the unique simulation of this β-reduction has length 5.

4.2 λu and $\lambda\sigma$ are incomparable

Lemmas 15 and 13 prove that the reductions $(\lambda\lambda(2\,2))1^n \to \lambda(2^n 2^n)$ with $n \geq 3$ show $\lambda u \not\prec \lambda\sigma$ and $\lambda u \not\prec \lambda\sigma_{DB}$.

On the other hand, it is immediate to verify that $(\lambda 2)1 \to 1$ has unique simulations in λu, $\lambda\sigma$ and $\lambda\sigma_{DB}$ with respective lengths 2, 4 and 3. Therefore, $\lambda\sigma \not\prec \lambda u$ and $\lambda\sigma_{DB} \not\prec \lambda u$.

4.3 λt and λs are incomparable

Lemmas 17 and 18 prove that the reductions $(\lambda\lambda(2\,2))1^n \to \lambda(2^n 2^n)$ with $n \geq 1$ show $\lambda s \not\prec \lambda t$.

On the other hand, $(\lambda\lambda\lambda 3)1 \to \lambda\lambda 3$ shows that $\lambda t \not\prec \lambda s$. In fact, as in Section 4.1 it is easy to check that every simulation of this β-reduction in λs has length 5.

4.4 λt and $\lambda\sigma$ are incomparable

The simulation in λt of $(\lambda 2)1 \to 1$ requires only 2 steps and hence (see Section 4.2) $\lambda\sigma \not\prec \lambda t$ and $\lambda\sigma_{DB} \not\prec \lambda t$.

To show $\lambda t \not\prec \lambda\sigma_{DB}$, consider the β-reduction at the root of $(\lambda\lambda\lambda 4)((\lambda 1)(\lambda 1))$. It is possible to achieve the simulation in 19 steps in $\lambda\sigma_{DB}$ (let $s = ((\lambda 1)(\lambda 1))\cdot id$):

$(\lambda\lambda\lambda 4)((\lambda 1)(\lambda 1)) \to (\lambda\lambda 4)[s] \twoheadrightarrow^3 \lambda\lambda\lambda(4[1\cdot((1\cdot((1\cdot(s\circ\uparrow))\circ\uparrow))\circ\uparrow)\circ\uparrow]) \to \lambda\lambda\lambda(3[(1\cdot((1\cdot(s\circ\uparrow))\circ\uparrow))\circ\uparrow]) \to \lambda\lambda\lambda(3[1[\uparrow]\cdot(((1\cdot(s\circ\uparrow))\circ\uparrow)\circ\uparrow)]) \to \lambda\lambda\lambda(2[((1\cdot(s\circ\uparrow))\circ\uparrow)\circ\uparrow]) \twoheadrightarrow^2 \lambda\lambda\lambda(2[1[\uparrow][\uparrow]\cdot(((s\circ\uparrow)\circ\uparrow)\circ\uparrow)]) \to$

$\lambda\lambda\lambda(\mathtt{1}[((s\circ\uparrow)\circ\uparrow)\circ\uparrow]) \twoheadrightarrow^2 \lambda\lambda\lambda(\mathtt{1}[s\circ\uparrow^3]) \to \lambda\lambda\lambda(\mathtt{1}[((\lambda\mathtt{1})(\lambda\mathtt{1}))[\uparrow^3]\cdot(id\circ\uparrow^3)]) \to \lambda\lambda\lambda(((\lambda\mathtt{1})(\lambda\mathtt{1}))[\uparrow^3]) \to$
$\lambda\lambda\lambda(((\lambda\mathtt{1})[\uparrow^3])((\lambda\mathtt{1})[\uparrow^3])) \twoheadrightarrow^2 \lambda\lambda\lambda((\lambda(\mathtt{1}[\mathtt{1}\cdot(\uparrow^3\circ\uparrow)]))\,(\lambda(\mathtt{1}[\mathtt{1}\cdot(\uparrow^3\circ\uparrow)]))) \twoheadrightarrow^2 \lambda\lambda\lambda((\lambda\mathtt{1})(\lambda\mathtt{1}))$

We must prove now that no simulation in λt of this β-reduction can be achieved in less than 19 steps. To do this we are going to prove a general result about λt. In Section 3.1 we have begun to study λt in order to compare it with λv. Remark the analogy between Lemma 2 and Lemma 3 we aim now to a lemma which should correspond to Lemma 4, i.e. a result which will enable us to calculate the length of the t-derivations of $a\,\varsigma^i b$. Unfortunately, not all the derivations have the same length as for λv. Furthermore, there is no easy way to compute the length of the shortest derivation as for $\lambda\sigma_{\Uparrow}$ (see Lemma 7). Hence, it does not seem easy to obtain such a general result. However, the shortest derivation of $a\,\varsigma^i b$ can always be calculated when a does not contain applications (like our example) and we proceed now to show it. The notions used here were introduced in Section 3.1.

Definition 21 *We define $N : \Lambda_\theta \to \mathbb{N}$ recursively as follows:*

$N(\mathtt{n}) = 0 \qquad N(ab) = N(a) + N(b) \qquad N(\lambda a) = N(a) \qquad N(\theta_k a) = M^t(a)$

Lemma 20 *For $a \in \Lambda_\theta$, every t-derivation of a to its t-nf has length $N(a)$.*

Proof: By induction on the weight $P(b)$ used to prove SN for the t-calculus and case analysis. The proof is analogous to the proof of Lemma 2. □

Definition 22 *Let $\Lambda^- ::= \mathbb{N} \mid \lambda\Lambda^-$, i.e. Λ^- is the set of λ-terms which do not contain applications. For $i \geq 1$, we define $Q_i^t : \Lambda^- \times \Lambda_\theta \to \mathbb{N}$ by induction as follows:*

$$Q_i^t(\mathtt{n}, b) = \begin{cases} 1 & \text{if } n \neq i \\ N(b)+1 & \text{if } n = i \end{cases} \qquad Q_i^t(\lambda a, b) = Q_{i+1}^t(a, \theta_0 b) + 1$$

Lemma 21 *For $a \in \Lambda^-$, $b \in \Lambda_\theta$ and $i \geq 1$ the shortest derivation of $a\,\varsigma^i b$ to its t-nf has length $Q_i^t(a,b)$.*

Proof: Analogous to the proof of Lemma 2 using Lemma 20 for the case $a = \mathtt{i}$. □

Now, since our simulation starts as $(\lambda\lambda\lambda\mathtt{4})((\lambda\mathtt{1})(\lambda\mathtt{1})) \to (\lambda\lambda\lambda\mathtt{4})\varsigma^1((\lambda\mathtt{1})(\lambda\mathtt{1}))$, we use the previous lemma to conclude that every simulation of the β-reduction at the root has length 20. Therefore, $\lambda t \not\preceq \lambda\sigma_{DB}$.

4.5 λt and $\lambda\sigma_{\Uparrow}$ are incomparable

The simulation in $\lambda\sigma_{\Uparrow}$ of $(\lambda\mathtt{2})\mathtt{1} \to \mathtt{1}$ requires 4 steps and hence (see Section 4.4) $\lambda\sigma_{\Uparrow} \not\preceq \lambda t$.

To show $\lambda t \not\preceq \lambda\sigma_{\Uparrow}$ we use the results of the previous subsection and the fact that there is a simulation in $\lambda\sigma_{\Uparrow}$ of the β-reduction at the root in $(\lambda\lambda\lambda\lambda\mathtt{4})((\lambda\mathtt{1})(\lambda\mathtt{1}))$ whose length is 14. Here it is (we denote again $s = ((\lambda\mathtt{1})(\lambda\mathtt{1}))\cdot id$):

$(\lambda\lambda\lambda\lambda\mathtt{4})((\lambda\mathtt{1})(\lambda\mathtt{1})) \to (\lambda\lambda\lambda\mathtt{4})[s] \twoheadrightarrow^3 \lambda\lambda\lambda(\mathtt{4}[\Uparrow^3(s)] \twoheadrightarrow^3 \lambda\lambda\lambda(\mathtt{1}[s\circ\uparrow^3]) \to$
$\lambda\lambda\lambda(\mathtt{1}[((\lambda\mathtt{1})(\lambda\mathtt{1}))[\uparrow^3]\cdot(id\circ\uparrow^3)]) \to \lambda\lambda\lambda(((\lambda\mathtt{1})(\lambda\mathtt{1}))[\uparrow^3]) \to$
$\lambda\lambda\lambda(((\lambda\mathtt{1})[\uparrow^3])((\lambda\mathtt{1})[\uparrow^3])) \twoheadrightarrow^2 \lambda\lambda\lambda((\lambda(\mathtt{1}[\Uparrow(\uparrow^3)]))\,(\lambda(\mathtt{1}[\Uparrow(\uparrow^3)]))) \twoheadrightarrow^2 \lambda\lambda\lambda((\lambda\mathtt{1})(\lambda\mathtt{1}))$

4.6 λs and $\lambda\sigma$ are incomparable

Lemmas 17 and 13 prove that the reductions $(\lambda\lambda(\mathtt{2}\,\mathtt{2}))\mathtt{1}^n \to \lambda(\mathtt{2}^n\mathtt{2}^n)$ with $n \geq 3$ show $\lambda s \not\preceq \lambda\sigma$ and $\lambda s \not\preceq \lambda\sigma_{DB}$.

On the other hand, it is immediate to verify that $(\lambda\mathtt{2})\mathtt{1} \to \mathtt{1}$ has a unique simulation in λs of length 2 and hence (see Section 4.2) $\lambda\sigma \not\preceq \lambda s$ and $\lambda\sigma_{DB} \not\preceq \lambda s$.

4.7 λs and $\lambda\sigma_{\Uparrow}$ are incomparable

It is immediate to verify that $(\lambda\mathtt{1})\mathtt{1} \to \mathtt{1}$ has unique simulations in λs and $\lambda\sigma_{\Uparrow}$ of respective lengths 3 and 2. Therefore, $\lambda s \not< \lambda\sigma_{\Uparrow}$.

On the other hand, the simulations in λs and $\lambda\sigma_{\Uparrow}$ of $(\lambda\mathtt{2})\mathtt{1} \to \mathtt{1}$ (see Sections 4.5 and 4.6) show that $\lambda\sigma_{\Uparrow} \not< \lambda s$.

4.8 λs and λv are incomparable

The reduction $(\lambda\lambda\mathtt{2})\mathtt{1} \to \lambda\mathtt{2}$ has unique simulations in λs and λv of respective lengths 4 and 5. Therefore, $\lambda v \not< \lambda s$.

On the other hand, $(\lambda\mathtt{1})\mathtt{1} \to \mathtt{1}$ has a unique simulation in λv of length 2 and hence (see Section 4.7) $\lambda s \not< \lambda v$.

4.9 $\lambda\sigma$ and λv are incomparable

Lemmas 14 and 13 prove that the reductions $(\lambda\lambda(\mathtt{2}\,\mathtt{2}))\mathtt{1}^n \to \lambda(\mathtt{2}^n\mathtt{2}^n)$ with $n \geq 2$ show $\lambda v \not< \lambda\sigma$ and $\lambda v \not< \lambda\sigma_{DB}$.

On the other hand, it is easy to verify that the shortest simulation in $\lambda\sigma$ (there are only 9), resp. $\lambda\sigma_{DB}$ (there are only 5), of $(\lambda\lambda\mathtt{2})\mathtt{1} \to \lambda\mathtt{2}$ has length 7, resp. 6, and hence (see Section 4.8) $\lambda\sigma \not< \lambda v$ and $\lambda\sigma_{DB} \not< \lambda v$.

4.10 $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$ are incomparable

Lemmas 19 and 13 prove that the reductions $(\lambda\lambda(\mathtt{2}\,\mathtt{2}))\mathtt{1}^n \to \lambda(\mathtt{2}^n\mathtt{2}^n)$ with $n \geq 1$ show $\lambda\sigma_{\Uparrow} \not< \lambda\sigma$ and $\lambda\sigma_{\Uparrow} \not< \lambda\sigma_{DB}$.

On the other hand, there is a simulation in $\lambda\sigma_{\Uparrow}$ of $(\lambda\lambda\mathtt{3})\mathtt{1} \to \lambda\mathtt{2}$ of length 7:

$(\lambda\lambda\mathtt{3})\mathtt{1} \to (\lambda\mathtt{3})[\mathtt{1}\cdot id] \to \lambda(\mathtt{3}[\Uparrow(\mathtt{1}\cdot id)]) \to \lambda(\mathtt{2}[(\mathtt{1}\cdot id)\circ\uparrow]) \to \lambda(\mathtt{2}[\mathtt{1}[\uparrow]\cdot(id\circ\uparrow)]) \to \lambda(\mathtt{1}[id\circ\uparrow]) \to \lambda(\mathtt{1}[\uparrow]) \to \lambda\mathtt{2}$

whereas it is easy to check that every simulation (there are only 14) in $\lambda\sigma$ of this β-reduction has length 8. Therefore, $\lambda\sigma \not< \lambda\sigma_{\Uparrow}$.

Unfortunately, the previous example does not work to show $\lambda\sigma_{DB} \not< \lambda\sigma_{\Uparrow}$. It is easy to find a simulation in $\lambda\sigma_{\Uparrow}$ of $(\lambda\lambda\lambda\mathtt{3})\mathtt{1} \to \lambda\lambda\mathtt{3}$ of length 9. However, in $\lambda\sigma_{DB}$ every simulation of this β-reduction has length at least 11. This can be checked by hand (even if there are thousands of derivations there is a lot of redundancy) or a simple program can do the work.

4.11 $\lambda\sigma_{\Uparrow}$ and λv are incomparable

The shortest simulation (there are only 2) in $\lambda\sigma_{\Uparrow}$ of $(\lambda\lambda\mathtt{2})\mathtt{1} \to \lambda\mathtt{2}$ has length 6 and hence (see Section 4.8) $\lambda\sigma_{\Uparrow} \not< \lambda v$.

On the other hand, there is a $\lambda\sigma_{\Uparrow}$-simulation of $(\lambda\lambda\lambda\lambda\mathtt{4})(\mathtt{1}\,\mathtt{1}) \to \lambda\lambda\lambda(\mathtt{4}\,\mathtt{4})$ of length 16:

$(\lambda\lambda\lambda\lambda\mathtt{4})(\mathtt{1}\,\mathtt{1}) \to (\lambda\lambda\lambda\mathtt{4})[(\mathtt{1}\,\mathtt{1})\cdot id] \twoheadrightarrow^3 \lambda\lambda\lambda(\mathtt{4}[\Uparrow^3((\mathtt{1}\,\mathtt{1})\cdot id)]) \twoheadrightarrow^3 \lambda\lambda\lambda(\mathtt{1}[((\mathtt{1}\,\mathtt{1})\cdot id)\circ\uparrow^3]) \to$
$\lambda\lambda\lambda(\mathtt{1}[(\mathtt{1}\,\mathtt{1})[\uparrow^3]\cdot(id\circ\uparrow^3)]) \to \lambda\lambda\lambda((\mathtt{1}\,\mathtt{1})[\uparrow^3]) \to \lambda\lambda\lambda(\mathtt{1}[\uparrow^3]\,\mathtt{1}[\uparrow^3]) \twoheadrightarrow^6 \lambda\lambda\lambda(\mathtt{4}\,\mathtt{4})$

whereas the length of every simulation in λv can be easily evaluated to 17: in fact, every derivation must start as: $(\lambda\lambda\lambda\lambda\mathtt{4})(\mathtt{1}\,\mathtt{1}) \to (\lambda\lambda\lambda\mathtt{4})[(\mathtt{1}\,\mathtt{1})/]$ and then apply Lemma 4 with $i = 0$. Therefore, $\lambda v \not< \lambda\sigma_{\Uparrow}$.

5 Conclusion

We summarize in the following table the results obtained so far. The table must be entered from the left, thus the information given, for instance, in position $(1,3)$ is to be read as $\lambda u \prec \lambda s$, whereas the information in position $(3,1)$ is $\lambda s \succ \lambda u$.

	λu	λt	λs	$\lambda \upsilon$	$\lambda \sigma$	$\lambda \sigma_{\Uparrow}$
λu	$=$	$\not\asymp$	$\prec$	$\prec$	$\not\asymp$	$\prec$
λt	$\not\asymp$	$=$	$\not\asymp$	$\prec$	$\not\asymp$	$\not\asymp$
λs	$\succ$	$\not\asymp$	$=$	$\not\asymp$	$\not\asymp$	$\not\asymp$
$\lambda \upsilon$	$\succ$	$\succ$	$\not\asymp$	$=$	$\not\asymp$	$\not\asymp$
$\lambda \sigma$	$\not\asymp$	$\not\asymp$	$\not\asymp$	$\not\asymp$	$=$	$\not\asymp$
$\lambda \sigma_{\Uparrow}$	$\succ$	$\not\asymp$	$\not\asymp$	$\not\asymp$	$\not\asymp$	$=$

References

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

[2] H. Barendregt. *The Lambda Calculus : Its Syntax and Semantics*. North Holland, 1984.

[3] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda \upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 1995.

[4] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Pitman, 1986. Revised edition : Birkhäuser (1993).

[5] P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical Report RR 1617, INRIA, Rocquencourt, 1992.

[6] N. G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report TH-Report 78-WSK-03, Department of Mathematics, Eindhoven University of Technology, 1978.

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

[8] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. Proceedings of PLILP'95. *Lecture Notes in Computer Science*, 982:45–62, 1995.

[9] F. Kamareddine and A. Ríos. An efficient calculus of substitutions. Technical report, Glasgow University, 1996.

[10] F. Kamareddine and A. Ríos. Bridging de Bruijn indices and variable names in explicit substitutions calculi. Technical report, Glasgow University, 1996.

[11] F. Kamareddine and A. Ríos. The confluence of the λs_e-calculus via a generalized interpretation method. Technical report, Glasgow University, 1996.

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

[13] C. A. Muñoz Hurtado. Confluence and preservation of strong normalisation in an explicit substitutions calculus. *LICS'96*, 1996.

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